Adapting Rigidity to Symmetric Cryptography: Towards ”Unswerving” Designs

Orr Dunkelman, Léo Perrin

▸ To cite this version:

Orr Dunkelman, Léo Perrin. Adapting Rigidity to Symmetric Cryptography: Towards ”Unswerving” Designs. SSR 2019 - Proceedings of the 5th ACM Workshop on Security Standardisation Research Workshop, Nov 2019, London, France. pp.69-80, 10.1145/3338500.3360335 . hal-02396695

HAL Id: hal-02396695

https://inria.hal.science/hal-02396695

Submitted on 6 Dec 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Adapting Rigidity to Symmetric Cryptography: Towards “Unswerving” Designs*

Orr Dunkelman[1] and Léo Perrin[2]

[1] Computer Science Department, University of Haifa, Haifa, Israel; orrd@cs.haifa.ac.il
[2] Inria, Paris, France; leo.perrin@inria.fr

Abstract. While designers of cryptographic algorithms are rarely considered as potential adversaries, past examples, such as the standardization of the Dual EC PRNG highlights that the story might be more complicated.

To prevent the existence of backdoors, the concept of rigidity was introduced in the specific context of curve generation. The idea is to first state a strict scope statement for the properties that the curve needs to have and then pick e.g. the one with the smallest parameters. The aim is to ensure that the designers did not have the degrees of freedom that allows the addition of a trapdoor.

In this paper, we apply this approach to symmetric algorithms. The task is challenging because the corresponding primitives are more complex: they consist of several sub-components of different types, and the properties required by these sub-components to achieve the desired security level are not as clearly defined. Furthermore, security often comes in this case from the interplay between these components rather than from their individual properties.

In this paper, we argue that it is nevertheless necessary to demand that symmetric algorithms have a similar but, due to their different nature, more complex property which we call *“unswervingness”*. We motivate this need via a study of the literature on symmetric “kleptography” and via the study of some real-world standards. We then suggest some guidelines that could be used to leverage the unswervingness of a symmetric algorithm to standardize a highly trusted and equally safe variant of it.

Keywords: Cryptography · Rigidity · Kleptography · Unswervingness

1 Introduction

Information needs to be protected. To this end, cryptographic protocols are used to ensure that it cannot be accessed by anyone but its intended recipient, and that it cannot be tempered with. Said protocols rely on lower level algorithms called *primitives*. These can be seen as the elementary bricks that must be used to construct secure protocols.

Primitives can be sorted into two broad categories. The first contains *public key* algorithms that rely on pairs of keys to provide their functionalities. In this case, a single key is intended to be broadcast (the *public* key) and one must be kept secret lest the whole security collapses (the *private* key). The second category contains algorithms that use only one key (the *secret* key) which must not be disclosed. They are called *symmetric*.

In this paper, we focus on symmetric primitives. They can be of several types.

Block Cipher. A block cipher is a family E_κ of permutations of $\{0,1\}^n$ that is indexed by a key κ of $\{0,1\}^k$. It should not be possible to recover κ, even if multiple pairs $(x, E_\kappa(x))$ are known. Encryption is but one of its many possible use: all other

*This paper will appear in the proceedings of SSR’19.

symmetric primitives can be built starting from a block cipher. Prominent examples include the now deprecated DES [DES77] and its very widely used successor, the AES [AES01].

Permutation. More recently, permutations have been introduced as an alternative elementary primitive to use instead of block cipher. These are simply bijective functions mapping $\{0,1\}^n$ to itself. A permutation is for example at the core of the sponge structure used by the latest standard hash function SHA-3 [SHA15b].

Stream Cipher. A stream cipher uses a key and (possibly) an initialization vector to produce a *keystream* of arbitrary length. This long bitstring is usually XORed with the plaintext to obtain the ciphertext. Stream ciphers, like for instance Trivium [De 06], are often intended for resource constrained devices. Another widely used example is the software oriented ChaCha20 [Ber08].

Hash Function. A hash function H maps the space of all bitstrings to $\{0,1\}^d$. Its output is a *digest* and d is its bitlength. Such a function needs to have several properties, including *collision resistance* (it is impossible in practice to find x, y such that $H(x) = H(y)$) and preimage resistance (given $z \in \{0,1\}^d$, it is impossible in practice to find x such that $H(x) = z$). For example, SHA-2 [SHA15a] is a commonly used hash function.

MAC. Message Authentication Codes are functions F_κ mapping all bitstrings to $\{0,1\}^d$ that are parameterized by a key $\kappa \in \{0,1\}^k$. When receiving a message $m||F_\kappa(m)$, the recipient can compute $F_\kappa(m)$ and see if the value obtained is the same as in what they receive. It ensures that the message was not tempered with and was indeed sent by someone who knows κ. For example, the HMAC [KBC97] construction constructs a MAC from a hash function.

Regardless of the details of their inner workings, the primitives must satisfy specific requirements in order for the protocols using them to be secure. For example, it must be impossible to deduce a private key from its corresponding public key. Protocol designers then assume that these properties are satisfied, and use them to argue about the security of their constructions.

A primitive can however fail to have the properties expected from it. For example, the hash function SHA-1 [Nat95] recently had its *collision resistance* disproved [SBK$^+$17]. As collision resistance is a crucial property for a cryptographic hash function, it proves that this primitive is not fit for its purpose. More generally, primitives are the subject of an intense scrutiny from the cryptographic community to ensure that they remain capable of withstanding all the attacks currently known.

This analysis builds upon the information provided by the designers of the primitive. What was the exact scope statement they followed? What trade-offs did they make? Is it possible that they underestimated a specific threat? Did they build counter-measures against a particular form of attack into their algorithm? It is expected that the designers answer these questions in the document specifying their primitive. The absence of such justifications significantly complicates the task of the cryptanalysts as they have for example to try and guess how the primitive prevents known attacks.

Unfortunately, it is not uncommon for primitives to be published without such explanations. In fact, several have been standardized by both national and international standardization bodies such as the SHA-1 [Nat95] hash function, the Dual EC pseudorandom number generator, or the Kuznyechik [Fed15] block cipher. Worse: as we will see below, the design explanations are not only necessary to ease a third party analysis, they are the only protection that exists against malicious designers.

1.1 Kleptography

Primitive designers are rarely assumed to be malevolent. The purpose of the third party analysis of a primitive is to verify that no mistakes were made, not to check the intentions of its designers.

Nevertheless, this assumption may be too optimistic. At Eurocrypt'97, Young and Yung [YY97] built upon previous work on subliminal channels [Sim83] to introduce the concept of *kleptography* which they defined as "the study of stealing information securely and subliminally" in the specific context of cryptography.[1] It considers a specific adversary, namely the *kleptographic attacker*, who "can steal the secrets securely, and in an exclusive and subliminal manner". In this context, the adversary designs an encryption scheme which is then made available as a black box for others to use. Said black-box implements added functionalities beyond the mere encryption of the plaintext that allow the kleptographic adversary to figure out the secret/private key of the user by observing some of its outputs. In other words, the kleptographic adversary tries to build a black-box containing a backdoor.

In the original paper [YY97] and later in [YY06], the authors focused on public key primitive. However, they considered block ciphers as well in [YY04]. In fact, over the years, multiple constructions of backdoored symmetric primitives have been proposed, so much so that all primitives are now covered.

- For block ciphers, a variant of the DES [Pat99] and an AES-like block cipher called BEA-1 [BBF16] have been constructed so as to contain a backdoor. We discuss them (and others) in more details in Section 4.1.1.
- For permutations, a significantly weakened version of SHA-3 was proposed by Morawiecki in [Mor15] (see Section 4.1.2).
- Stream ciphers and PRNGs are the only family we are aware of for which an actual standard has a backdoor, namely Dual EC. More details about how it was standardized are provided in [CCG$^+$16]. Its flaws were leveraged by an attacker to target Juniper [CMG$^+$16]. It may also have been used against TLS [CNE$^+$14].
- For hash function, on top of Morawiecki's proposal, backdoored variants of SHA-1 and Streebog were presented respectively in [AAE$^+$14] and in [AY14]. They are also discussed in Section 4.1.2.
- The issue of backdooring MACs is studied in [FJM18], where the authors discuss the ability of backdooring hash functions. Then, using these backdoored hash functions, they construct a backdoored HMAC construction (among others).

In this paper, we focus on backdoors added at the primitive level. There are of course other means of purposefully weakening a cryptosystem. An extensive survey on the more general topic of surreptitiously subverting cryptography is [SFKR15].

More precisely, our focus is on symmetric primitives. In order to exist, such a backdoor requires one of the following two situations to occur:

- either the weakness is such that that anyone could theoretically find it, but it is believed that such an occurrence will not happen because the public state of the art will not catch up to the analysis of the designer, or
- the primitive is, in fact, a public key cryptosystem.

Regardless, the insertion of such a backdoor would significantly constrain the design of the primitive.

[1] In this paper we study the academic literature for the matter of backdooring cryptography. Historical real-life examples, such as the Crytpo AG company, exist and highlight the fact that backdoored cryptography is a real-life issue.

1.2 Outline of our Contribution

Our first aim is to survey the literature on inserting backdoors in symmetric primitives. We narrow our scope down to the algorithms and their specification. In particular, we do not consider backdoors inserted in specific implementations using e.g. hardware trojans.

Building upon this knowledge, we then intend to provide a general framework to assess whether the designers of an algorithm could have introduced a flaw to their primitive. It is based on the new concept of *moldability* which takes inspiration from the *rigidity* that was introduced to provide similar guarantees for elliptic curves (a type of asymmetric primitive).

Outline. Section 2 describes the sub-components used to construct symmetric primitives and the design space of each of them. Section 3 presents the techniques that can be used to prevent well known attacks. In Section 4, we argue that rigidity is indeed needed even in the context of symmetric primitives, although it has some limitations in this context. Finally, we present in Section 5 a new notion, *unswervingness*. Related to rigidity, it emphasizes the importance of the design process over that of the specific primitive considered. Indeed, it would be harder to insert a backdoor in an explicit set of design requirements than in a fully specified algorithm. We use it as a basis for a possible improved standardization process.

2 The Sub-Components of a Symmetric Algorithm

In this section, we list the sub-components that are necessary for constructing a symmetric primitive as well as the properties they are expected to have. We will build upon this knowledge in Section 3 to describe how their properties can be used to prove that the primitive is safe from some attacks, and then in Section 4 to explain how specific subcomponent can allow the insertion of backdoors in a primitive.

A symmetric primitive is very often built like a block cipher. Hash function are either sponge-based or effectively use a block cipher in a specific mode to construct a compression function, which in turn is used to build the hash function. The same goes for MACs. Permutations are built using the same components and the same security arguments as block ciphers, although in their case the “key” is fixed.

Stream ciphers that are not block ciphers in counter mode tend to use different components. We discuss those separately in Section 2.7. We thus focus on block ciphers first.

2.1 Key Alternating Block Cipher

A block cipher is, formally, a family of permutations of $\{0,1\}^n$ parameterized by a key which is an element of $\{0,1\}^k$. In this case, n is the *block size* and k is the *key size*. Both of these quantities matter from a security standpoint: if the key is too short then an exhaustive search becomes possible. On the other hand, if the block size is too small then attacks exploiting collisions in the output of the cipher can become practical. Such an attack was shown practical when $n = 64$ in [BL16].

In practice, a block ciphers consists of the iteration of a simple function called the *round function* which is interleaved with *key additions*. This structure is represented Figure 1. The latter operations simply consists of XORing a *round key* into the encryption state. These round keys are derived from the *master key* using a *key scheduling algorithm*. Said algorithm can be trivial: if $k = n$ then it can simply set $K_i = K \oplus c_i$ for some round constants c_i. Such round constants can also be used in a more sophisticated way inside the key scheduling algorithm. We discuss their properties in Section 2.2.

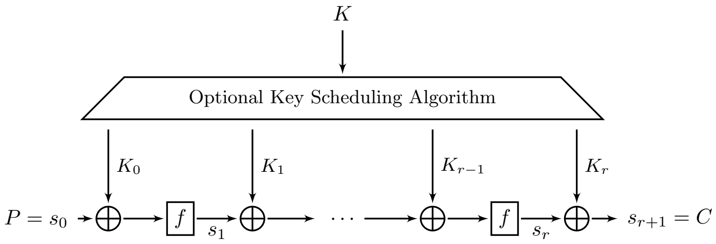

Figure 1: A key alternating block cipher: K is the master key, the K_i are the round keys, and f is the round function.

The round function itself may have different overall structures: it can be a Substitution-Permutation Network (SPN), a Feistel network, or have a structure based on a logic gates. In the case of a Feistel network, an SPN is usually used as a subfunction. An SPN round consists in the application of a small non-linear function called *S(ubstitution)-box* in parallel over the state followed by a linear permutation applied on the full state that mixes the outputs of the different S-boxes. We then need simply to consider the components of an SPN (Sections 2.3 and 2.4 for the linear layer and the S-box, respectively) and the gate-based case (Section 2.5).

2.2 Round Constants

Formally, round constants are bit-strings that are XORed or added into either the key state or the cipher state and which are different in each round. Their role is to ensure that the encryption rounds are different from one another. For example, in the AES, round constants are XORed into the key state inside the key schedule.

In order to ensure that they are different in each round, it is popular to use a simple counter or the internal state of a small LFSR as a round constant (as done e.g. in the AES).

Another design strategy consists of simply choosing random bit-strings during the design phase and then hard-coding them into the implementation. This requires a good source of random bits (possibly, a verified and tested one).

2.3 Linear Layer

The purpose of the linear layer is to ensure that, eventually, all bits of the internal state depend on all the bits from the input and from the key. Since the S-box (see below) operates in parallel on different subsets of the state, it is insufficient to achieve this goal.

While this property (called *diffusion* by Shannon [Sha49]) does not a priori impose that the component providing it is linear, it is almost always the case in practice because linear functions and their interaction with the S-box layer are easier to study. In fact, it is possible to formally quantify how much diffusion is provided by a linear layer using the so-called *branching number*. Let $n = m\ell$ be a block size and let m be the size of the S-box applied in parallel over the state. Then the branching number of a linear permutation $L : (\{0,1\}^m)^\ell \rightarrow (\{0,1\}^m)^\ell$ in this context is

$$\min_{x \neq 0} \left(\mathrm{wt}(x) + \mathrm{wt}\left(L(x)\right)\right) \ ,$$

where $\mathrm{wt}(x)$ is the number of non-zero m-bit words in $x \in (\{0,1\}^m)^\ell$. The higher the branching number, the better the diffusion provided by the linear layer. In the AES, a linear layer operating on a quarter of the state that has the maximum branching number is used.

However, while a high branching number is a desirable property, it is not necessary for the cipher to be good. In particular, another popular choice of linear layer is a simple bit permutation—which always has the worse possible branching number: 2. It is for example how the linear layer of the lightweight block cipher PRESENT [BKL+07] works.

2.4 S-Boxes

S-boxes are the only non-linear components of the ciphers using them. Formally, they are functions

$$S : \{0,1\}^m \rightarrow \{0,1\}^n$$

where m is small enough that S can be described by its lookup table, i.e. by the sequence of n-bit values $\{S(0), ..., S(2^m - 1)\}$. Typically, $m = n \in \{4, 8\}$ but other possibilities exist, e.g. $m = 6$ and $n = 4$ for the DES.

Informally, S-boxes are supposed to increase the complexity of the relation of the internal state bits with one another and with the key bits—a property called *confusion* by Shannon [Sha49]. This complexity can be quantified in different ways that are each related to a proof of security against some attacks: the *differential uniformity* matters for differential attacks (see Sections 3.3), and the linearity is used in linear attacks (see Section 3.4). Overall, the mathematical properties of the S-box play a crucial role in the security of the cipher.

Another important aspect to take into account when choosing an S-box is its implementation efficiency. It can be implemented with a simple table lookup, but this technique is not well-suited for a hardware implementation and it can lead to timing attacks when used in software. S-boxes are then chosen that have a specific structure to ensure the possibility of an efficient and secure implementation. For example, the S-box of PRESENT was chosen so as to minimize the number of gates needed to implement it in hardware. Similarly, the S-boxes of the block cipher SERPENT [BAK98] were chosen so as to minimize the number of instructions needed to implement them in a bit-sliced fashion. Finally, the S-box of Keccak [BDPA13] (which became SHA-3) is implemented via a simple cellular automata which implies the existence of efficient software and hardware implementations.

A wide variety of design choices are possible, as illustrated in Figure 2. It shows the structures used to build the 8-bit S-boxes of 53 different algorithms from the literature. “AES S-box”, “Inverse”, “Exponential”, and “Math (other)” correspond to structures obtained via simple mathematical formulas, usually over the finite field. SPN, Misty, Feistel and Lai-Massey correspond to block cipher-like structures, Misty and Lai-Massey being closely related to the Feistel network. The last three categories correspond to structures that do not yield any simple description.

The last category corresponds to S-boxes whose design structure was not disclosed by its authors.

2.5 Gate-Based Operations

There are ciphers with round functions that do not really correspond to the SPN or Feistel structure (or any of their variants). Instead, they are described as a network of various gates taking one or several w-bit words as input and outputting one such word. A popular choice in this case is for the designers to restrict themselves to gates corresponding to addition modulo 2^w, w-bit word rotations and w-bit XOR for $w \in \{8, 16, 32, 64\}$. The resulting design paradigm is nicknamed ARX (Addition, Rotation, XOR). For example, the stream cipher ChaCha20 [Ber08] is built in this fashion. The choice of the gates can be less restricted, such as in SHA-256 where the ARX gates are supplemented with the bitwise AND and OR of two 32-bit words.

The boundary between gate-based and SPN can be blurry. Indeed, the round function of Keccak is an SPN round with a 5-bit S-box but it can be completely specified in a

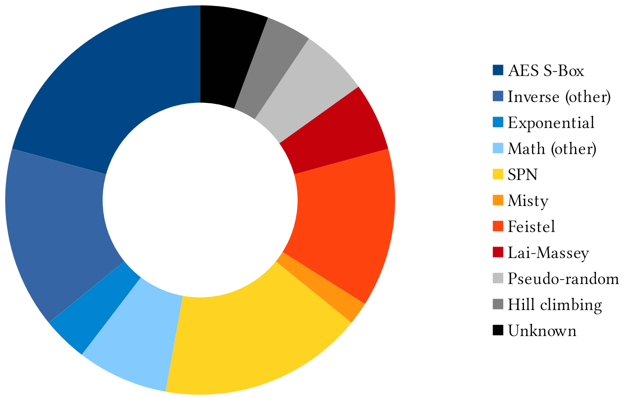

Figure 2: The structures of the 8-bit S-boxes used by 53 different algorithms (taken from [Per17]).

gate-based fashion. Similarly, the block cipher SPARX [DPU+16] is an ARX-based SPN. On the other hand, an algorithm like SIMON [BSS+13] that relies on AND, rotations and XORs does not have any component with a clear SPN structure.

2.6 Key Schedule

Most of the block ciphers transform a single key into multiple subkeys using a key schedule algorithm (a few others just repeat the key words, possibly in a different order). Some key schedules are linear (either by picking different bits for different rounds as in DES [DES77] or IDEA [LM91], or by using an LFSR as in KATAN [DDK09]), but some others employ nonlinear components in it, either the use of ARX operations or the use of S-boxes. This is done mainly to avoid related-key attacks (and its variants, such as slide attacks or reflection attacks). Finally, some primitives such as blowfish [Sch94] derive the S-boxes from the key (in what is called key-dependent S-boxes).

While there may be some keys for which the encryption is weaker (or even broken), it is unknown whether there is a general method to inject weaknesses (besides those that can be easily identified) into the key schedule.

Open Problem 1. *Is it possible to insert an undetectable backdoor through the key schedule algorithm?*

2.7 Stream Cipher Components

Stream ciphers can be built using a block cipher-based function, i.e. using a hash function with a counter and a key as its input (as in ChaCha20 [Ber08]) or using a plain block cipher in counter mode.

However, it is common to use dedicated structures that rely on Linear Feedback Shift Registers (LFSR), Non Linear Feedback Shift Registers (NLFSR) and Boolean functions mapping $\{0,1\}^w$ to $\{0,1\}$ instead. Software-oriented stream ciphers also use large arrays which get continuously updated during the encryption process, like RC4. In this case, an internal state that depends only on the key (and possibly an initialization vector) is initialized and updated using such functions. Then, during the production of the keystream, a filter function is used to obtain a bit or a byte of keystream from the state which is updated each time.

Some standard stream cipher have a hybrid structure like SNOW 3G [ETS06] which uses both an LFSR and some components of the AES to update its internal state.

In the rest of this paper, we will set such stream cipher components aside because, to the best of our knowledge, how such components could be used to introduce backdoors has not received any attention from the community. This was a surprise to us as one of the most prominent cases of standardized backdoored algorithm is essentially a stream cipher, Dual EC. It should be noted however that it was built using components from public key cryptography (a very unusual choice) rather than the components listed above.[2]

Open Problem 2. *Is it possible to insert a backdoor in an LFSR and NLFSR-based stream cipher?*

A possible starting point starting point for this research direction could take inspiration from the A5/2 stream cipher which was successfully attacked for instance in [GWG99]. Its structure allows a trivial guess and determine attack. Could this flaw be generalized?

3 Ensuring Security Against Known Attacks

3.1 Design Process

There are two main approaches for designing cryptographic primitives — the reductionist approach and the "engineering" approach. The reductionist approach is based on taking a well known hard problem (e.g., factoring or computing discrete logarithm), and using it to build a cryptosystem. Then, using standard reduction arguments (or sometimes, not so standard and very involved reduction arguments) it is possible to show that breaking the cryptosystem is as hard[3] as solving the hard problem. This approach is very common in public-key cryptosystems as well as in designing modes of operation for block ciphers (where the security proof suggests that breaking the encryption scheme is as hard as breaking the block cipher).

The second approach, the "engineering" one, balances performance and security. In the case of symmetric-key primitives this means that some more basic round is designed (e.g., in block ciphers and hash functions). This round is then iterated as many times as needed to ensure security (measured by applying all known attacks, such as the ones discussed below) and as few times as possible to maintain acceptable performance. Hence, it is very important to make sure how many rounds can be attacked, and what are the security margins of the scheme.

3.2 (Lack of) Avalanche

The first class of attacks, first proposed by Diffie and Hellman, takes advantage of insufficient mixing between state (i.e. plaintext or intermediate encryption values) and key bits [DH77]. Originally proposed against Double-DES, the *meet in the middle attack* tries to compute some internal value from both sides (plaintext and ciphertext). If a match is found, then further analysis is performed. Later, Chaum and Evertse showed how to use this attack in the context of attacking single-DES, by proposing the first attack faster than brute force against 7-round DES [CE86].

The underlying property is indeed the lack of mixing, or as commonly referred to, lack of avalanche. Hence, during the late 70's and the 80's, one important measure for discussing security of cryptographic schemes was the avalanche criteria [Fei73], later extended to the strict avalanche criteria [WT86].

In recent years a more subtle version of this property has been discovered and exploited to attack block ciphers — the (invariant) subspace attack [LAAZ11]. In this type of attacks, a structured subset of plaintexts is identified, such that the corresponding intermediate encryption values (or ciphertexts) belong also to a structured subset.

[2] We note that the Dual EC construction and backdoor are based on elliptic curves operations.

[3] Some of these reductions are not very tight.

3.3 Differential Attacks

Differential cryptanalysis [BS91] was published by Biham and Shamir in 1990. However, there is strong evidence that the NSA knew of the technique already in the 1970's, and that IBM's original cipher Lucifer [Sor84] was "updated" to DES [DES77], after the NSA revealed the technique to the designers of Lucifer/DES in IBM.

The technique itself studies the evolution of differences through the primitive, i.e., it studies two calls to the primitive, with some input difference (either in the plaintext or in the key), and studies the development of the differences through the rounds of the primitive. In block ciphers, differential cryptanalysis is built around finding a good *differential characteristic*, a prediction about the development of the differences throughout the encryption with as high probability as possible. This probability should differ from the probability of a similar transition from input difference to output difference for a random permutation (or an ideal cipher). Once this difference is high enough, one can collect plaintext pairs that satisfy the input difference, and check how many pairs that satisfy the differential characteristic exist in the dataset. Such pairs can later be used in attacks that recover the key.[4]

Differential cryptanalysis has been one of the two pillars of statistical cryptanalysis to this day, and thus, enjoys a great number of generalizations and extensions such as truncated differentials (working with sets of possible differences) [Knu95], higher order differentials (working with sets, rather than pairs) [Knu95], boomerang attacks [Wag99] (combining two different short differentials into an attack, rather than a long one), etc.

We note that differential cryptanalysis in the context of hash functions (which is mostly used for collision finding attacks) also aims for finding high probability differential characteristics that end in specific output differences, e.g., zero for collision. Yet, for hash functions the adversary has a greater control over the pairs that satisfy the characteristic, with an emphasis on techniques which are built on the ability to force some differences to evolve according to the adversary's gain using methods such as message modification.

To claim security against differential cryptanalysis it is very common to show that there are no "high probability" or impossible differentials. This is done by techniques based on Nyberg and Knudsen's seminal work on provable security against differential (and linear) cryptanalysis [NK95], which is the base for the wide trail strategy famously used to design the AES [AES01]. It is based on combining an S-box S for which the equation $S(x \oplus a) \oplus S(x) = b$ as a bounded number of solutions x with a linear layer that has a high branching number. It is then possible to prove bounds on the probability of differential characteristics.

One should be careful though—bounding the probability of a specific differential characteristic (or showing by various tools that no such differentials characteristic exist), does not guarantee that there are no high probability differentials. Still, it is the state of the art technique for proving resilience against differential attacks at the time of writing.

3.4 Linear Attacks

Linear cryptanalysis, proposed by Matsui [Mat94], is the second pillar of symmetric-key cryptanalysis. This attack is built upon finding linear approximations of the primitive, namely, a linear relation holding between the parity of some input and some output bits that has a probability which differs from the expected one (of 1/2). The attack offers a known plaintext attack of trying to assess whether the linear approximation holds or not (as a way to distinguish the primitive from an ideal one). Of course, using a linear

[4] We alert the informed reader that we do not discuss the important distinction between differential characteristics and differentials, as well as some other important, yet subtle points. We also do not discuss the process of finding the statistical properties, though of course it is of great importance.

distinguisher can also be used for key recovery attacks, and the best academic cryptanalytic result against DES is based on this technique.

A large volume of work deals with extensions and generalizations for linear cryptanalysis. Most notably, the use of multiple linear approximations simultaneously [HCN08], the use of zero-correlation approximations (for which the approximation holds with probability exactly 1/2) [BW12], differential-linear cryptanalysis [LH94] which combined the two methods, or partitioning cryptanalysis [HM97].

Similarly to differential cryptanalysis, some techniques allow designers to claim that there are no good (high bias) linear approximations for algorithms. Again, the methodology proposed by [NK95] (and the wide trail strategy) is to count the number of active S-boxes, and then obtain an upper bound on the bias of any linear approximation of the cipher. A linear layer with a high branching number allow a particularly simple argument in this case.

As with differential cryptanalysis this methodology may not work as expected due to the linear hull effect (multiple approximations sharing the input/output masks), and may not suggest security against multidimensional linear cryptanalysis.

3.5 Integral Attacks

A different type of statistical attacks is the "integral attack", which is also named square attack or saturation attack [DKR97, Luc02, KW02]. This type of attack is based on collecting a specially structured set of inputs (for example, a set composed of 256 plaintexts, all agreeing on all bytes but one, where the remaining byte takes all possible values). Somewhat related to higher-order differential cryptanalysis, the attack studies the propagation of some "labels" describing the set of plaintexts. Common labels include *Constant* (all intermediate encryption values in the set have the same value in a word), *Active* (all values of a word appear once in the intermediate encryption values of the set), *Balanced* (the XOR of all values of a word in the intermediate encryption values of the set is 0), or *Even* (each value that appears in the set is of an even multiplicity).

This type of attack is usually well suited for ciphers with "word"-oriented design (e.g., the AES, where a 4-round integral property is the base for many cryptanalytic results). One advantage of this approach is that as long as the S-box is bijective, the exact S-box does not affect the development of the integral.

In recent years this attack was extended to the division property (which can be roughly summed as "bit-oriented integral") [Tod15].

3.6 Algebraic Attacks

A completely different strand of cryptanalysis is based on algebraic attacks. Motivated by Shannon's quote that breaking a good cipher should require "as much work as solving a system of simultaneous equations in a large number of unknowns of a complex type" [Sha49], this type of attacks tries to model the problem as a set of multivariate equations that need solving. For example, in the context of block ciphers, this involves writing the plaintext/ciphertext/key relations, and solving the equations for the key, whereas for hash functions this may include finding preimages or collisions.

Usually, algebraic attacks require very little data, but their running time is hard to predict. In many cases, the tools that solve the system of equations (be it Gröbner-basis solvers, SAT solvers, or MILP solvers) run for a long unpredictable periods of time, require a lot of memory, etc.

The common method to address security against this line of attacks is to argue that the degree of the polynomials or that the distribution of monomials (i.e., how many monomials of a certain degree exist in the compact description of the block cipher) obtained by the cryptosystem reaches the bounds expected for random permutations/functions (depending on the primitive). At the same time, one should be very careful with these kinds of "security guarantees" as there are many dependencies that may cause the scheme to have a lower degree than expected [BCD11].

4 The Need for an Improved Rigidity

Rigidity was introduced in [Ber13] in the context of the design of an elliptic curve cryptosystem, a type of asymmetric primitive. It is an improvement of the folklore notion of *nothing-up-my-sleeve number*. We rephrase this definition in a more general context as follows.

Definition 1 (Rigidity)**.** The design space of a rigid algorithm is highly constrained so that only a few algorithms can satisfy all of its requirements.

In order to introduce a backdoor in an otherwise convincing algorithm, an attacker needs to use a design process that leaves enough degrees of freedom to choose subcomponents with the properties that their backdoor requires. The aim of a rigid design process is to take away this freedom by enforcing that the set of all possible algorithm is very small.

In this section, we argue that the problem solved by rigidity in the context of elliptic curves, namely the possibility of a backdoored algorithm, is very relevant for symmetric primitives as well. To this end, we provide an extensive survey of the literature on backdooring techniques in Section 4.1. We then look at two specific groups of algorithms and argue that they fail to meet the intuitive requirements for a trusted algorithm, namely the block ciphers SIMON and SPECK (Section 4.2) as well as Streebog and Kuznyechik (Section 4.3). We summarize the lessons learnt from these examples in Section 4.4.

4.1 Known Backdooring Techniques

Various building blocks used to construct symmetric algorithms can be used to insert backdoors. Below, we look at S-boxes and round constants. To the best of our knowledge, no method is known that relies on the properties of the linear layer.

4.1.1 S-box-based

It may seem counterintuitive to try and insert a backdoor through the S-box. Indeed, this component must satisfy a lot of properties that are easy to verify for a typical S-box size such a low linearity or a low differential uniformity which help prevent respectively differential attacks (see Sections 3.3) and linear attacks (see Section 3.4). Intuitively, these quantities measure how well the S-box mixes its input: how could an S-box satisfying such criteria yield a flaw? As we will see, this intuition is wrong.

The first backdoor proposed in the literature was based on a hidden flaw against linear attacks [RP97]. In this case, the S-box had to be big enough that computing its linear approximation table was not feasible, meaning that the best linear approximation could not be found in reasonable time without knowledge of the backdoor. However, this approach was proved flawed in [WBDY98]: the authors found a general algorithm capable of recovering such linear approximations.

Research then focused on constructing S-box-based backdoors that are of "regular" size. Since in this case the resilience against differential and linear attacks can be directly assessed, other attacks need to be considered. The first work in this direction was performed by Paterson in [Pat99]: he introduced a variant of the DES which essentially allowed partition attacks. Though this technique was introduced [HM97] just two years before the publication of [Pat99], the latter work actually predates it: it was finished already in 1994 but it remained unpublished for several years.

Much later, in 2016, the backdooring strategy of Paterson was revisited by Bannier et al. [BBF16]. They proved that, in order to introduce such a trapdoor, the S-box had to satisfy some very specific properties. Let $S : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ be the S-box: there must exist subspaces V and W of $\mathbb{F}_2^n$ such that for all $a \in \mathbb{F}_2^n$ there exists a $b \in \mathbb{F}_2^n$ such that $S(a + V) = b + W$. In other words, it must map additive cosets of a subspace to additive cosets of a (possibly different) subspace. They also proved that such requirements were not at odds with having good differential and linear properties. They were then able to design a block cipher called BEA-1 which is proved secure against linear and differential attacks using a wide trail argument but which is also vulnerable to a devastating partition-like attack with practical complexity.

Another very recent approach does not consist of allowing partition attacks but rather differential attacks where the difference does not correspond to the XOR but to another group operation [CBS19]. The trapdoor then consists in knowing this group operation.

Many standards have been published without a proper explanation for the design process of their S-box. These merely specify their S-box via their lookup table. While the process used was eventually disclosed or recovered in some cases, it often remains mysterious.

DES. The DES [DES77] (standardized by the American NBS, the NIST ancestor) uses eight S-boxes mapping 6 bits to 4. They were studied shortly after their publication and several non-random patterns were identified [HMS+76]. However, it was not before 1994 that the design criteria were published by one of the authors of the DES [Cop94]. It is now deprecated unless used 3 times in sequence (triple-DES).

Streebog/Kuznyechik. Streebog [Fed12] and Kuznyechik [Fed15] are two algorithms that are standardized in Russia as *GOST* standards. They share the same 8-bit S-box which has been the subject of an intense scrutiny that eventually lead to the recovery of its structure. We provide more details in Section 4.3.

Skipjack. Skipjack [U.S98] was an American NIST standard block cipher originally intended for use in the Clipper chip. It uses an 8-bit S-box for which the design process remains mysterious, although some advances on analyzing it were made in [BP15]. It is now deprecated.

TIA ciphersuite. The ciphersuite used by the north American standards for cell phone communication relies on two 8-bit S-boxes that were never explained [Qui09]. CMEA is a block cipher that uses an 8-bit S-box called the "CaveTable" which was successfully attacked in [WSK97]. The CaveTable is also used by a key derivation function called *Cave*. Another unexplained 8-bit S-box called "ibox" is used by the "Enhanced CMEA" (ECMEA).

SM4. The current standard block cipher in China is SM4 [Dt08]. It was shown in [LJH+07] that its S-box, though originally unexplained, had simply been constructed like that of the AES.

4.1.2 Round Constant-based

While S-boxes have been considered for backdooring block ciphers, modifications of the round constants have been shown to allow efficient attacks against hash function. Most prominently, it was shown by Albertini et al. that the constants of the hash function SHA-1 could be modified to allow collision attacks [AAE+14]. More precisely, they showed that the constants could be chosen during a first collision search so as to speed it up. Furthermore, their constant choice was plausible, i.e. they left the message expansion unchanged instead of requiring the use of independent round constants in each round. The resulting design is a hash function as convincing as SHA-1—except that its designers know a collision. Due to the Merkle-Damgård construction of SHA-1, this first collision can be used to build many others.

Later, variants of the standards Keccak and Streebog were proposed. For Keccak, Morawiecki proposed in 2015 [Mor15] to use round constants generated in a simple way using the string "`SHA3SHA3`" as a seed. The fact that the corresponding 64-bit string is symmetric means that strong symmetries are preserved throughout the whole hashing. This in turn allows a collision search in time $2^{d/4}$ where d is the digest size, while a hash function is expected to require $2^{d/2}$ operations to solve this problem.

For Streebog, AlTawy and Youssef [AY14] obtained similar results as those obtained in [AAE+14] for SHA-1. During the design phase, they chose the constant so as to allow a specific collision. Then, using the Merkle-Damgård structure of Streebog, they can use it to construct more collisions. In the case of Streebog, their search was helped by the lack of constant schedule. Indeed, unlike in SHA-1, the round constants are independent and specified separately which gives the attacker much more freedom. After the publication of [AY14], the authors of Streebog published a note explaining their design choice [Rud15].

4.2 SIMON and SPECK

SIMON and SPECK [BSS+13] are two families of lightweight block ciphers designed by the NSA in 2013. Both families target low-end devices, the SIMON family follows a relatively standard Feistel construction using an unusual gate-based round function, whereas the SPECK family uses the ARX paradigm.

The original design document [BSS+13] lists the different variants (each family supports different block and key sizes) and offers performance information on different platforms, but suggests no design rationale or criteria.

Despite that, SIMON and SPECK were submitted to standardization in ISO[5] in October 2014.[6] However, besides the design document, ISO has not received any design rationale nor any additional information that could attest to the cryptographic strength of the algorithms. This is despite the fact that, as NSA designed cryptosystems (and especially after the revelation of the Bullrun program), they received a great deal of cryptanalytic attention—mostly for the small block size variants.

The submission of SIMON and SPECK for standardization despite the big security concerns has caused a great deal of backlash. Since the beginning, different concerns with respect to the ill understood security of SIMON and SPECK were raised by members of SC27.[7] For example, when Belgium raised concerns in the preparations to the Jaipur meeting (October 2015), the NSA contacted the scientists behind the claims, and asked them to change their conclusion [Ash15].

Given the pressure to explain the design decisions, as some of them are slightly unorthodox (for example, in SIMON, the key addition is performed at the end of the round, and thus, the first round is essentially moot from security point of view), the opposition to standardization was very fierce. Moreover, the discussion about the smaller block sizes (most notably, 32-bit and 48-bit block sizes) in the face of the Sweet32 attack [BL16], led the NSA to first withdraw their support for the smaller block sizes.

To support the cause of the bigger block sizes, the NSA released a document in 2017 listing the design rationale [BSS+17] and a few design choices. For example, the report discusses the choice of the rotation constants used in SIMON, suggesting that they are

[5] It appears that the rules of ISO do not allow discrimination in favor or against a specific designer (or a group of designers). Hence, even if many in the cryptographic community did not like the idea of Simon and Speck being standardized, ISO had to process the SIMON and SPECK as US-developed ciphers.

[6] After the Snowden revelations, and the disclosure of the Dual EC incident, NIST announced that they will accept NSA-based crypto only after public vetting, e.g., in conferences and in international standardization bodies.

[7] The full name of the sub-committee is ISO/IEC JTC 1/SC27, and the relevant workgroup is WG2.

not optimal w.r.t. security, but almost optimal, in exchange for a significantly better performance on 8-bit microcontrollers. However, the amount of security lost by the transition is not mentioned (i.e., it is unclear how many more rounds can be attacked following the slightly sub-optimal choice of constants).

Another missing element from the design rationale document is the designers' analysis. While the report discusses the resistance of the ciphers to various attacks, it does not explicitly mention the amount of rounds that the designers could break using the different techniques, bur just summarizing existing works. Hence, one cannot evaluate the design strategy, as it is unclear whether all the results obtained by the academic community invalidate the design methodology or the internal security claims. Moreover, as the design rationale was published *after* many such papers, it is impossible to figure out whether the attacks found by academics simply mirror those originally found by the designers or if they in fact outperform them.

Finally, one of the most lacking feature of the design rationale document is a complete lack of claims regarding the safety margins offered by the different algorithms. We remind the reader that most block ciphers try to balance between security and performance, and one important aspect of this tradeoff is the understanding of the safety margins—what is the performance penalty that the designer picked to counter future advances in cryptanalysis. While [BSS$^+$17] contains some text on the matter of these safety margins, the text is far from being clear, and a straightforward reading of the text suggests that for some SIMON variants, the safety margins are smaller than the designers intended [Ash17].

Following the above issues, SIMON and SPECK were not standardized into the security and cryptography standards of ISO. At the same time, due to the distributed nature of ISO, some other subcommittees (specifically, SC31 which deals with RFID communication), did decide to include SIMON and SPECK into their standards despite the fact that the cryptography and security experts of SC27 voted in a landslide against the standardization of SIMON and SPECK.

4.3 The Russian S-box

In 2012, the Russian Federation standardized a new hash function called Streebog [Fed12]. It was followed in 2015 by a new block cipher, Kuznyechik [Fed15]. In parallel, these were both standardized as IETF RFC via the independent stream in 2013 (as RFC 6986) and 2016 (as RFC 7801) respectively. Streebog has also been an ISO standard since 2018 and Kuznyechik is being considered for standardization at the time of writing. Both use the same S-box nicknamed "π" which permutes the set $\{0,1\}^8$. In both specification, it is only described by its lookup table.

It was first studied by Biryukov, Perrin and Udovenko in [BPU16] where the authors managed to identify a yet unknown structure inside π, meaning that there was an algorithm capable of evaluating π that required less information than is contained in its lookup table. In a follow-up work [PU16], Perrin and Udovenko found a completely different structure in π which prompted them to conjecture the existence of a third structure. The structures described in [BPU16] and in [PU16] would then simply be side effects of this third one.

It turned out to be correct: in [Per19], Perrin showed that π could be written using very simple equations over the finite field. He also proved the previous structures from [BPU16] and [PU16] were indeed mere side-effects of the last one. More importantly, he could use this structure to identify a highly structured and previously unknown interaction between π and particular vector spaces of $\{0,1\}^8$, namely the multiplicative and additive cosets of the subfield. This discovery is worrying because, as recalled in Section 4.1.1, an S-box interacting in a non-trivial way with additive cosets of a vector space is a known backdooring technique.

The situation of π cannot be fully understood using only the framework of rigidity. Indeed, while it appeared at first to have no rigidity, it turned out in the end to come from a very small set of permutation, meaning that it is in fact a very rigid component. Surprisingly, its designers claimed that the rigidity of their design which was outlined by the first two structures found in π was a coincidence. This is evidenced by an internal ISO memo that was eventually leaked [SM18]. Astonishingly, they maintained this claim *after* the third structure was published, as can be seen in the leaked summary of the discussion that took place after the publication of this result [YH19]. Bonnetain et al. then proved that the arguments of the designers were factually incorrect [BPT19], as the probability for an S-box to satisfy such properties is very small.

4.4 Lessons Learnt

As evidenced by the literature on backdooring techniques, the threat posed by a malevolent designer is as relevant in the case of symmetric cryptography as in the case of asymmetric cryptography. We therefore need a notion close to that of rigidity: it is necessary to ensure that the subcomponents of an algorithm do not allow secret attacks.

At the same time, as recalled in Section 2, a symmetric primitive consists of more subcomponents than, say, an elliptic curve. Furthermore, we should be careful that the requirements preventing the insertion of a backdoor do not needlessly hamper the task of the designers. They need to be able to perform all the trade-offs necessary, and they also need to be able to choose some of the subcomponents depending on some of the properties of the others. As a consequence, we should not simply use the notion of rigidity directly in symmetric cryptography by defining the rigidity of an algorithm as the rigidity of each of its subcomponents taken independently.

Furthermore, the examples of the Russian algorithms (Section 4.3) and of the NSA lightweight blockciphers (Section 4.2) highlight the crucial importance of the specification and of a proper statement of the design requirements. It also shows that the publication of such a rationale must be done *before* any external cryptanalysis has taken place.

If the rationale is not published, an algorithm cannot be trusted. In the case of the Russian ciphers, it is even worse as all the evidence points to the design requirements provided being purposefully misleading.

Building upon these remarks, we propose a variant of rigidity that is better suited for symmetric primitives in the next section.

5 Unswervingness: a Better Rigidity for Symmetric Primitives

5.1 Definition

In light of the discussion in the previous paragraph, we propose the following notion.

Definition 2 (Moldability and Unswervingness)**.** The design requirements of an algorithm are *moldable* if there exists a weak algorithm that fits *all* of its explicit design requirements (including those regarding its efficiency). The opposite of moldability is *unswervingness.*

The main difference with rigidity is that unswervingness does not put the emphasize on the size of the set of the algorithm fitting all the design requirements—although we expect that it significantly reduces it (see below). Instead, it demands that no weak algorithm exist in this set.

An unswerving set of requirement does not exist for all algorithms, e.g. if all the components are picked randomly in an unreproduceable way. In this case, it is fair to say that the algorithm itself is moldable.

We have explicitly mentioned efficiency requirements as being part of the definition of moldability. Indeed, it is common to for example list specific cryptographic design criteria for an S-box and then to choose the one with the best implementation properties from the corresponding set.

Moldability can also be used to better understand some results from the literature. For example, in [KR01], Knudsen and Raddum showed the requirements presented by the designers of the block cipher Noekeon were satisfied by S-boxes that yielded a weak cipher. What they showed is not an attack against this algorithm. Instead, they have proved that the specification of Noekeon was moldable.

We claim that unswervingness is a highly desirable property for the design requirements of an algorithm.

Some techniques for preventing the exploitation of backdoors in some specific constructions have been proposed e.g. for HMAC and HKDF [FJM18]. Unswervingness is more general as it can be applied to any primitive.

5.2 A Solution to Known Problems

As discussed in Section 4, the designer of a symmetric cryptographic primitive can be an adversary. Imposing that all standardized designs have unswerving requirements would significantly cripple the capability of a malevolent designer in several ways.

1. Obviously, imposing that no algorithm vulnerable to known attack satisfies the design requirements of a given cipher imposes that said cipher is safe from said attacks.

2. An algorithm for which no design requirements are published is extremely moldable as for example its non-linear components could be replaced by linear ones, or its diffusion layers could be substituted with the identity, leading in both cases to an extremely weak cipher satisfying all the (non-existent) design requirements. As a consequence, algorithms like SIMON/SPECK or the Russian ciphers (discussed in Sections 4.2 and 4.3 respectively) have maximum moldability.

3. An unswerving set of design requirements has to be thorough, meaning that it forces a form of rigidity. Indeed, it imposes that the algorithm lives in a small set (that of the a priori secure ones).

4. As moldability is a property of the design requirements rather than of the algorithm or its subcomponents, algorithms with a rigid but secret design process (as was used for the design of the Russian S-box as explained in Section 4.3) cannot have unswerving design requirements.

5. The concepts moldability and unswervingness can be used as the foundation of a specific standardization process which we describe below in Section 5.3.

At the same time, the notion of unswervingness is flexible enough to allow the designers to make all kinds of tradeoffs and design decisions. Perhaps the S-box can be chosen somewhat freely but, once it is fixed, the linear layer is fully defined. In this case, we cannot really say that the choice of each subcomponent is rigid since their might be many valid S-box/linear layer pairs. However, it is unswerving.

5.3 Possible Directions for better Standardization Processes

We propose the following standardization process which has its roots in the concept of unswervingness. It also takes inspiration from the process used by the designers of the *million dollar curve* [BDF+16].

Suppose that, after a substantial cryptanalysis effort, a specific algorithm has gained the trust of the cryptographic community. It is then decided to standardize it. During the analysis phase, cryptographers considered not only the algorithm itself but all those satisfying all of its design requirements. In light of this, the standardizing agency could operate as follows.

1. Commit to the standardization of a variant of the algorithm.
2. Publicly commit to a component generation process such that its output will satisfy all the (appropriately unswerving) design requirements.
3. Gather entropy in a publicly verifiable way to seed the generation process.
4. Modify the algorithm to use the newly designed components.
5. Standardize the result.

This process has several advantages. First, imposing unswervingness means in practice forcing the designers to clearly state all their design criteria. This will ease the task of third party cryptographers analyzing the security of the algorithm.

Its main feature however is the prevention of backdoors. Indeed, in order to ensure the presence of the mathematical properties required by a backdoor, a malevolent designer would need to state these requirements explicitly—a disclosure which would substantially simplify the identification of the backdoor by third party cryptographers.

Nevertheless, this process has its caveats. The main one we see is that multiple standardizing bodies may agree on the same algorithm: for example, the AES is both a NIST and an ISO standard (among others). If each body were to pick new components then their standards would not be interoperable. Still, this problem is easily solved: once a set of components outside of the control of the designers has been chosen, other standardizing bodies can simply reuse them.

Another possible problem is the huge computational cost of selecting and analyzing said components. It is unclear whether different standardization bodies have the same capabilities (both in computation and/or in the expertise of writing the required search procedures). A partial solution to that problem is to allow designers to offer several options for each sub-component, e.g., a few S-boxes or a few MDS matrices. Then, the task of selecting the exact combination may be easier.

6 Conclusion

In light of our survey of the literature and of several case studies, we have argued that the design of symmetric primitives is as vulnerable to malevolent designers as that of asymmetric ones. In order to deny such designers the degrees of freedom that their endeavor would require, we have introduced the notions of moldability and unswervingness. We claim that it is the correct framework for assessing the security level of a symmetric primitive when the designers can be considered adversarial. Further, we have proposed a specific standardization process leveraging this concept to ensure that the final algorithm is safe.

In addition to mitigating (some) risks related to backdoored cryptography, we note that the proposed approaches allow honest designers to increase the trust in their design. This follows from the fact that once unswervingness is maintained, there is less likelihood of succeeding in hiding a backdoor.

Acknowledgements

The authors thank the anonymous reviewers for their insights and suggestions and John Kelsey for shepherding this submission. The first author was supported in part by the Israel Ministry of Science and Technology, the Center for Cyber, Law, and Policy in conjunction with the Israel National Cyber Bureau in the Prime Minister’s Office and by the Israeli Science Foundation through grant No. 880/18.

References

[AAE+14] Ange Albertini, Jean-Philippe Aumasson, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Malicious hashing: Eve’s variant of SHA-1. In Antoine Joux and Amr M. Youssef, editors, *SAC 2014: 21st Annual International Workshop on Selected Areas in Cryptography*, volume 8781 of *Lecture Notes in Computer Science*, pages 1–19. Springer, Heidelberg, August 2014.

[AES01] Advanced Encryption Standard (AES). National Institute of Standards and Technology (NIST), FIPS PUB 197, U.S. Department of Commerce, November 2001.

[Ash15] Tomer Ashur. Simon: NSA-designed Cipher in the Post-snowden World, 2015. Talk at the Technion’s CRYPTODAY.

[Ash17] Tomer Ashur. Notes on “Notes on the design and analysis of SIMON and SPECK” and an Analysis of it, 2017. Rump Session Presentation at EUROCRYPT 2017.

[AY14] Riham AlTawy and Amr M. Youssef. Watch your constants: Malicious streebog. Cryptology ePrint Archive, Report 2014/879, 2014. http://eprint.iacr.org/2014/879.

[BAK98] Eli Biham, Ross J. Anderson, and Lars R. Knudsen. Serpent: A new block cipher proposal. In Serge Vaudenay, editor, *Fast Software Encryption – FSE’98*, volume 1372 of *Lecture Notes in Computer Science*, pages 222–238. Springer, Heidelberg, March 1998.

[BBF16] Arnaud Bannier, Nicolas Bodin, and Eric Filiol. Partition-based trapdoor ciphers. Cryptology ePrint Archive, Report 2016/493, 2016. http://eprint.iacr.org/2016/493.

[BCD11] Christina Boura, Anne Canteaut, and Christophe De Cannière. Higher-order differential properties of Keccak and Luffa. In Antoine Joux, editor, *Fast Software Encryption – FSE 2011*, volume 6733 of *Lecture Notes in Computer Science*, pages 252–269. Springer, Heidelberg, February 2011.

[BDF+16] Thomas Baignères, Cécile Delerablée, Matthieu Finiasz, Louis Goubin, Tancrède Lepoint, and Matthieu Rivain. Trap me if you can, February 2016.

[BDPA13] Guido Bertoni, Joan Daemen, Michael Peeters, and Gilles Van Assche. Keccak. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 313–314. Springer, Heidelberg, May 2013.

[Ber08] Daniel J Bernstein. Chacha, a variant of salsa20. In *Workshop Record of SASC*, volume 8, pages 3–5, 2008.

[Ber13] Daniel Bernstein. Safecurves: choosing safe curves for elliptic-curve cryptography, 2013.

[BKL$^+$07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems – CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, Heidelberg, September 2007.

[BL16] Karthikeyan Bhargavan and Gaëtan Leurent. On the practical (in-)security of 64-bit block ciphers: Collision attacks on HTTP over TLS and OpenVPN. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 456–467. ACM Press, October 2016.

[BP15] Alex Biryukov and Léo Perrin. On reverse-engineering S-boxes with hidden design criteria or structure. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 116–140. Springer, Heidelberg, August 2015.

[BPT19] Xavier Bonnetain, Léo Perrin, and Shizhu Tian. Anomalies and vector space search: Tools for S-box reverse-engineering. Cryptology ePrint Archive, Report 2019/528, 2019.

[BPU16] Alex Biryukov, Léo Perrin, and Aleksei Udovenko. Reverse-engineering the S-box of streebog, kuznyechik and STRIBOBr1. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part I*, volume 9665 of *Lecture Notes in Computer Science*, pages 372–402. Springer, Heidelberg, May 2016.

[BS91] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. In Alfred J. Menezes and Scott A. Vanstone, editors, *Advances in Cryptology – CRYPTO'90*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer, Heidelberg, August 1991.

[BSS$^+$13] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK families of lightweight block ciphers. Cryptology ePrint Archive, Report 2013/404, 2013. http://eprint.iacr.org/2013/404.

[BSS$^+$17] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. Notes on the design and analysis of SIMON and SPECK. Cryptology ePrint Archive, Report 2017/560, 2017. http://eprint.iacr.org/2017/560.

[BW12] Andrey Bogdanov and Meiqin Wang. Zero correlation linear cryptanalysis with reduced data complexity. In Anne Canteaut, editor, *Fast Software Encryption – FSE 2012*, volume 7549 of *Lecture Notes in Computer Science*, pages 29–48. Springer, Heidelberg, March 2012.

[CBS19] Roberto Civino, Céline Blondeau, and Massimiliano Sala. Differential attacks: using alternative operations. *Designs, Codes and Cryptography*, 87(2):225–247, Mar 2019.

[CCG$^+$16] Stephen Checkoway, Shaanan Cohney, Christina Garman, Matthew Green, Nadia Heninger, Jacob Maskiewicz, Eric Rescorla, Hovav Shacham, and Ralf-Philipp Weinmann. A systematic analysis of the juniper dual EC incident.

Cryptology ePrint Archive, Report 2016/376, 2016. http://eprint.iacr.org/2016/376.

[CE86] David Chaum and Jan-Hendrik Evertse. Crytanalysis of DES with a reduced number of rounds: Sequences of linear factors in block ciphers. In Hugh C. Williams, editor, *Advances in Cryptology – CRYPTO'85*, volume 218 of *Lecture Notes in Computer Science*, pages 192–211. Springer, Heidelberg, August 1986.

[CMG+16] Stephen Checkoway, Jacob Maskiewicz, Christina Garman, Joshua Fried, Shaanan Cohney, Matthew Green, Nadia Heninger, Ralf-Philipp Weinmann, Eric Rescorla, and Hovav Shacham. A systematic analysis of the juniper dual EC incident. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016: 23rd Conference on Computer and Communications Security*, pages 468–479. ACM Press, October 2016.

[CNE+14] Stephen Checkoway, Ruben Niederhagen, Adam Everspaugh, Matthew Green, Tanja Lange, Thomas Ristenpart, Daniel J. Bernstein, Jake Maskiewicz, Hovav Shacham, and Matthew Fredrikson. On the practical exploitability of dual EC in TLS implementations. In Kevin Fu and Jaeyeon Jung, editors, *USENIX Security 2014: 23rd USENIX Security Symposium*, pages 319–335. USENIX Association, August 2014.

[Cop94] Don Coppersmith. The data encryption standard (DES) and its strength against attacks. *IBM journal of research and development*, 38(3):243–250, 1994.

[DDK09] Christophe De Cannière, Orr Dunkelman, and Miroslav Knežević. KATAN and KTANTAN - a family of small and efficient hardware-oriented block ciphers. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems – CHES 2009*, volume 5747 of *Lecture Notes in Computer Science*, pages 272–288. Springer, Heidelberg, September 2009.

[De 06] Christophe De Cannière. Trivium: A stream cipher construction inspired by block cipher design principles. In Sokratis K. Katsikas, Javier Lopez, Michael Backes, Stefanos Gritzalis, and Bart Preneel, editors, *ISC 2006: 9th International Conference on Information Security*, volume 4176 of *Lecture Notes in Computer Science*, pages 171–186. Springer, Heidelberg, August / September 2006.

[DES77] Data encryption standard. National Bureau of Standards, NBS FIPS PUB 46, U.S. Department of Commerce, January 1977.

[DH77] Whitfield Diffie and Martin E. Hellman. Special Feature Exhaustive Cryptanalysis of the NBS Data Encryption Standard. *IEEE Computer*, 10(6):74–84, 1977.

[DKR97] Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. The block cipher Square. In Eli Biham, editor, *Fast Software Encryption – FSE'97*, volume 1267 of *Lecture Notes in Computer Science*, pages 149–165. Springer, Heidelberg, January 1997.

[DPU+16] Daniel Dinu, Léo Perrin, Aleksei Udovenko, Vesselin Velichkov, Johann Großschädl, and Alex Biryukov. Design strategies for ARX with provable bounds: Sparx and LAX. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part I*, volume 10031 of *Lecture*

Notes in Computer Science, pages 484–513. Springer, Heidelberg, December 2016.

[Dt08] Whitfield Diffie and George Ledin (translators). SMS4 encryption algorithm for wireless networks. Cryptology ePrint Archive, Report 2008/329, 2008. http://eprint.iacr.org/2008/329.

[ETS06] ETSI/Sage. Specification of the 3GPP Confidentiality and Integrity Algorithms UEA2 & UIA2. Document 2: SNOW 3G Specification. Technical report, ETSI/Sage, September 2006.

[Fed12] Federal Agency on Technical Regulation and Metrology. Information technology – data security: Hash function. English version available at http://wwwold.tc26.ru/en/standard/gost/GOST_R_34_11-2012_eng.pdf, 2012.

[Fed15] Federal Agency on Technical Regulation and Metrology. Information technology – data security: Block ciphers. English version available at http://wwwold.tc26.ru/en/standard/gost/GOST_R_34_12_2015_ENG.pdf, 2015.

[Fei73] Horest Feistel. Cryptography and Computer Privacy. *Scientific American*, 228(5):15–23, 1973.

[FJM18] Marc Fischlin, Christian Janson, and Sogol Mazaheri. Backdoored Hash Functions: Immunizing HMAC and HKDF. In *31st IEEE Computer Security Foundations Symposium, CSF 2018, Oxford, United Kingdom, July 9-12, 2018*, pages 105–118. IEEE Computer Society, 2018.

[GWG99] Ian Goldberg, David Wagner, and Lucky Green. The (Real-Time) Cryptanalysis of A5/2, 1999. Rump Session Presentation at CRYPTO 1999.

[HCN08] Miia Hermelin, Joo Yeon Cho, and Kaisa Nyberg. Multidimensional linear cryptanalysis of reduced round Serpent. In Yi Mu, Willy Susilo, and Jennifer Seberry, editors, *ACISP 08: 13th Australasian Conference on Information Security and Privacy*, volume 5107 of *Lecture Notes in Computer Science*, pages 203–215. Springer, Heidelberg, July 2008.

[HM97] Carlo Harpes and James L. Massey. Partitioning cryptanalysis. In Eli Biham, editor, *Fast Software Encryption – FSE'97*, volume 1267 of *Lecture Notes in Computer Science*, pages 13–27. Springer, Heidelberg, January 1997.

[HMS$^+$76] M. Hellman, R. Merkle, R. Schroeppel, L. Washington, W. Diffie, S. Pohlig, , and P. Schweitzer. Results of an initial attempt to cryptanalyze the NBS Data Encryption Standard. Technical report, Stanford University, Information Systems Laboratory, 1976. Available at https://ee.stanford.edu/~hellman/resources/1976_sel_des_report.pdf.

[KBC97] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. HMAC: Keyed-hashing for message authentication. RFC 2104, RFC Editor, February 1997. http://www.rfc-editor.org/rfc/rfc2104.txt.

[Knu95] Lars R. Knudsen. Truncated and higher order differentials. In Bart Preneel, editor, *Fast Software Encryption – FSE'94*, volume 1008 of *Lecture Notes in Computer Science*, pages 196–211. Springer, Heidelberg, December 1995.

[KR01] Lars R. Knudsen and Håvard Raddum. On noekeon, 2001.

[KW02] Lars R. Knudsen and David Wagner. Integral cryptanalysis. In Joan Daemen and Vincent Rijmen, editors, *Fast Software Encryption – FSE 2002*, volume 2365 of *Lecture Notes in Computer Science*, pages 112–127. Springer, Heidelberg, February 2002.

[LAAZ11] Gregor Leander, Mohamed Ahmed Abdelraheem, Hoda AlKhzaimi, and Erik Zenner. A cryptanalysis of PRINTcipher: The invariant subspace attack. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 206–221. Springer, Heidelberg, August 2011.

[LH94] Susan K. Langford and Martin E. Hellman. Differential-linear cryptanalysis. In Yvo Desmedt, editor, *Advances in Cryptology – CRYPTO’94*, volume 839 of *Lecture Notes in Computer Science*, pages 17–25. Springer, Heidelberg, August 1994.

[LJH$^+$07] Fen Liu, Wen Ji, Lei Hu, Jintai Ding, Shuwang Lv, Andrei Pyshkin, and Ralf-Philipp Weinmann. Analysis of the SMS4 block cipher. In Josef Pieprzyk, Hossein Ghodosi, and Ed Dawson, editors, *ACISP 07: 12th Australasian Conference on Information Security and Privacy*, volume 4586 of *Lecture Notes in Computer Science*, pages 158–170. Springer, Heidelberg, July 2007.

[LM91] Xuejia Lai and James L. Massey. A proposal for a new block encryption standard. In Ivan Damgård, editor, *Advances in Cryptology – EUROCRYPT’90*, volume 473 of *Lecture Notes in Computer Science*, pages 389–404. Springer, Heidelberg, May 1991.

[Luc02] Stefan Lucks. The saturation attack - a bait for Twofish. In Mitsuru Matsui, editor, *Fast Software Encryption – FSE 2001*, volume 2355 of *Lecture Notes in Computer Science*, pages 1–15. Springer, Heidelberg, April 2002.

[Mat94] Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In Tor Helleseth, editor, *Advances in Cryptology – EUROCRYPT’93*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer, Heidelberg, May 1994.

[Mor15] Pawel Morawiecki. Malicious Keccak. Cryptology ePrint Archive, Report 2015/1085, 2015. http://eprint.iacr.org/2015/1085.

[Nat95] National Institute of Standards and Technology. FIPS 180-1: Secure hash standard, 1995.

[NK95] Kaisa Nyberg and Lars R. Knudsen. Provable security against a differential attack. *Journal of Cryptology*, 8(1):27–37, December 1995.

[Pat99] Kenneth G. Paterson. Imprimitive permutation groups and trapdoors in iterated block ciphers. In Lars R. Knudsen, editor, *Fast Software Encryption – FSE’99*, volume 1636 of *Lecture Notes in Computer Science*, pages 201–214. Springer, Heidelberg, March 1999.

[Per17] Léo Perrin. *Cryptanalysis, Reverse-Engineering and Design of Symmetric Cryptographic Algorithms*. PhD thesis, University of Luxembourg, Belval, Luxembourg, 2017.

[Per19] Léo Perrin. Partitions in the S-box of Streebog and Kuznyechik. *IACR Transactions on Symmetric Cryptology*, 2019(1):302–329, 2019.

[PU16] Léo Perrin and Aleksei Udovenko. Exponential s-boxes: a link between the s-boxes of BelT and Kuznyechik/Streebog. *IACR Transactions on Symmetric Cryptology*, 2016(2):99–124, 2016. http://tosc.iacr.org/index.php/ToSC/article/view/567.

[Qui09] Frank Quick. Common cryptographic algorithms. Available online at https://www.3gpp2.org/Public_html/Specs/S.S0053-0_v2.0.pdf., 2009.

[RP97] Vincent Rijmen and Bart Preneel. A family of trapdoor ciphers. In Eli Biham, editor, *Fast Software Encryption – FSE'97*, volume 1267 of *Lecture Notes in Computer Science*, pages 139–148. Springer, Heidelberg, January 1997.

[Rud15] V. Rudskoy. Note on Streebog constants origin. Available online at https://tc26.ru/upload/medialibrary/efb/streebog_constants_eng%20Rudskoi.pdf, 2015.

[SBK+17] Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov. The first collision for full SHA-1. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 570–596. Springer, Heidelberg, August 2017.

[Sch94] Bruce Schneier. Description of a new variable-length key, 64-bit block cipher (Blowfish). In Ross J. Anderson, editor, *Fast Software Encryption – FSE'93*, volume 809 of *Lecture Notes in Computer Science*, pages 191–204. Springer, Heidelberg, December 1994.

[SFKR15] Bruce Schneier, Matthew Fredrikson, Tadayoshi Kohno, and Thomas Ristenpart. Surreptitiously weakening cryptographic systems. Cryptology ePrint Archive, Report 2015/097, 2015. http://eprint.iacr.org/2015/097.

[Sha49] Claude E. Shannon. Communication theory of secrecy systems. *Bell Systems Technical Journal*, 28(4):656–715, 1949.

[SHA15a] Secure hash standard (shs). National Institute of Standards and Technology (NIST), FIPS PUB 180-4 , U.S. Department of Commerce, August 2015.

[SHA15b] Sha-3 standard: Permutation-based hash and extendable-output function. National Institute of Standards and Technology (NIST), FIPS PUB 202 , U.S. Department of Commerce, August 2015.

[Sim83] Gustavus J. Simmons. The prisoners' problem and the subliminal channel. In David Chaum, editor, *Advances in Cryptology – CRYPTO'83*, pages 51–67. Plenum Press, New York, USA, 1983.

[SM18] Vasily Shishkin and Grigory Marshalko. A Memo on Kuznyechik S-Box. ISO/IEC JTC 1/SC 27/WG 2 Officer's Contribution N1804, September 2018.

[Sor84] Arthur Sorkin. Lucifer, a Cryptographic Algorithm. *Cryptologia*, 8(1):22–42, 1984.

[Tod15] Yosuke Todo. Structural evaluation by generalized integral property. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 287–314. Springer, Heidelberg, April 2015.

[U.S98] U.S. Department Of Commerce/National Institute of Standards and Technology. Skipjack and KEA algorithms specifications, v2.0, 1998.

[Wag99] David Wagner. The boomerang attack. In Lars R. Knudsen, editor, *Fast Software Encryption – FSE’99*, volume 1636 of *Lecture Notes in Computer Science*, pages 156–170. Springer, Heidelberg, March 1999.

[WBDY98] Hongjun Wu, Feng Bao, Robert H. Deng, and Qin-Zhong Ye. Cryptanalysis of Rijmen-Preneel trapdoor ciphers. In Kazuo Ohta and Dingyi Pei, editors, *Advances in Cryptology – ASIACRYPT’98*, volume 1514 of *Lecture Notes in Computer Science*, pages 126–132. Springer, Heidelberg, October 1998.

[WSK97] David Wagner, Bruce Schneier, and John Kelsey. Cryptanalysis of the cellular encryption algorithm. In Burton S. Kaliski Jr., editor, *Advances in Cryptology – CRYPTO’97*, volume 1294 of *Lecture Notes in Computer Science*, pages 526–537. Springer, Heidelberg, August 1997.

[WT86] A. F. Webster and Stafford E. Tavares. On the design of S-boxes (impromptu talk). In Hugh C. Williams, editor, *Advances in Cryptology – CRYPTO’85*, volume 218 of *Lecture Notes in Computer Science*, pages 523–534. Springer, Heidelberg, August 1986.

[YH19] Hirotaka Yoshida and Jonathan Hammell. Meeting report for the discussion on Kuznyechik and Streebog, April 2019.

[YY97] Adam Young and Moti Yung. Kleptography: Using cryptography against cryptography. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT’97*, volume 1233 of *Lecture Notes in Computer Science*, pages 62–74. Springer, Heidelberg, May 1997.

[YY04] Adam Young and Moti Yung. A subliminal channel in secret block ciphers. In Helena Handschuh and Anwar Hasan, editors, *SAC 2004: 11th Annual International Workshop on Selected Areas in Cryptography*, volume 3357 of *Lecture Notes in Computer Science*, pages 198–211. Springer, Heidelberg, August 2004.

[YY06] Adam Young and Moti Yung. A space efficient backdoor in RSA and its applications. In Bart Preneel and Stafford Tavares, editors, *SAC 2005: 12th Annual International Workshop on Selected Areas in Cryptography*, volume 3897 of *Lecture Notes in Computer Science*, pages 128–143. Springer, Heidelberg, August 2006.