Cryptanalysis of Simon et al.

cryptanalysis of lightweight symmetric ciphers

Lambooij, E.

Award date:
2017

Cryptanalysis of Simon et al.

Cryptanalysis of lightweight symmetric ciphers

E. Lambooij
Email: e.lambooij@student.tue.nl
Student-ID: 0721553

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science
in
Computer Science and Engineering

Supervisors:
dr. Orr Dunkelman (University of Haifa)
prof.dr. Tanja Lange (TU/e)
msc. Rolf Pielage (Deloitte)

July 2017

Contents

Abstract

Unlike most ciphers the lightweight symmetric cipher Simon did not have a design rationale or any cryptanalysis when it was published. To determine whether the cipher is safe to be used in real life a thorough cryptanalysis is to be conducted. The cryptanalysis of ciphers has always been a cornerstone in the design of new ciphers. This thesis consists of three methods analysing the cryptographic strength of Simon. The first method defines a generic distinguisher for k-encryption using the same key with sub full-codebook complexity. The second method describes a framework to transform Simon into an S-box based cipher. This reduces the cost of computing a Difference Distribution Table from 2^{2n} to $n \cdot \frac{2^{16}}{4}$ where n denotes the word size used. The third method introduces a related key differential attack covering 14 rounds and using 2^{16} chosen plaintexts. These results, although not breaking Simon, provide cryptanalysts with new tools to work with and provide new directions into which researchers can conduct their research.

Acknowledgements

INTENTIONALLY LEFT BLANK

1 Introduction

Cryptogaphy has had a major influence in the technical innovation of the last couple of decades. Many modern processes rely on the use of cryptography to ensure confidentiality and integrity. These processes are protected by certain assumptions made in the design of the cryptographic protocols.

Cryptographic protocols are built using standard building blocks, which are called cryptographic primitives. Examples of these cryptographic building blocks are: stream ciphers, block ciphers, cryptographic hashes and asymmetric encryption/signatures/key exchange. Cryptographic protocols can be proven to be strong, assuming the used cryptographic primitives are strong. Or in other words, if the building blocks are cryptographically weak, then no strong cryptographic protocols using them can exist.

One could say that researching cryptographic primitives is as important as developing strong cryptographic protocols. This thesis focuses on finding weak spots in cryptographic primitives, i.e. the cryptanalysis of cryptographic primitives. It mainly focuses on the cryptanalysis of block ciphers. This thesis mainly focuses on the Simon family of block ciphers by Beaulieu, Shors, Smith, and Treatman-clark [3].

In contrast to asymmetric cryptographic primitives, symmetric cryptographic primitives usually do not have an underlying mathematical hard problem. This means that the cryptographic primitive cannot be proven computationally hard by giving a reduction proof to a known hard problem. A reduction proof is a proof that maps a certain mathematical problem onto another mathematical problem. By giving a reduction proof, one can prove that a mathematical problem is at least as hard as another problem. In the case of cryptographic primitives one would want to give a reduction proof to a random self reducible hard problem, such as the discrete log problem, lattice problems such as the nearest vector problem, or multivariate polynomial equation solving.

In protocols, cryptographic primitives are often modelled as a perfect version of the primitives. To prove a security protocol a block cipher is often modelled as a Perfect Random Permutation (PRP). A PRP is a bijective function that maps an input onto an output, thus every input maps exactly one output and every output is mapped to by exactly one input. Moreover a PRP is a random mapping, which means that there exists no relation between in- and output. In Figure 1 a graphical representation of a PRP is given.

Modelling a block cipher as a PRP is convenient for protocol designers, but since a block cipher needs a concrete description it cannot be proven to be a PRP. This implies that other techniques need to be used to assure the cryptographic strength of a block cipher.
The cryptographic strength of a primitive is expressed as a security level. The

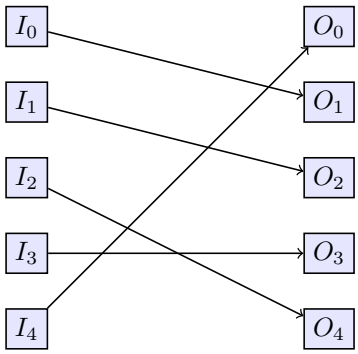

Figure 1: An instance of a PRP (Perfect Random Permutation) mapping 5 inputs to 5 outputs

unit of a security level is bits. A primitive with a security level of 80-bits means that the amount of work needed to be done to break a cipher that is in the order of 2^{80} operations.
The most fundamental way to bound the security level of a block cipher is to look at how much work it is to try out all different keys. This is called a brute force search and generally speaking for block ciphers with a k-bits key the number of different keys is 2^k. Thus the upper bound of the security level of a block cipher using a k-bits key is k.
To find better attacks than a brute force search several techniques have been proposed in the past, e.g. differential [4], linear [29] and algebraic cryptanalysis [12] . One part of the cryptanalysis of a cipher is trying to carry out known 'attacks' and observing if any of the attacks lower the security level of the cipher. In this manner a cryptanalyst can prove an upper bound on the security level of a cipher, but unlike with a PRP no lower bound of a cipher can be proven in this way.

By the nature of the process, assessing the security level of a cipher costs a significant amount of time. Another drawback is that the cryptographic strength of a symmetric cipher can only be assessed against known attacks. This makes it vital not only to cryptanalyse ciphers using known attacks, but to devise new attacks as well and to communicate the results with the public. If new attacks are not shared, a situation could occur where different researchers know different attacks. This could lead to a situation where one group of researchers has knowledge of an attack of which the other researchers (in most cases the public) is not aware. Using this information the first group of researchers could construct a cipher that is weak against the new attack and strong against all publicly known attacks. The public, not aware of the existence of this attack, will assume the cipher to be strong, since it survives all known attacks. This is an undesirable situation.

The currently used block ciphers have all received lots of public crutiny. Of all the major ciphers e.g. AES (Rijndael cipher) [14], Twofish [42], RC5 [40], PRESENT [9], etc. the security bounds under the publicly known attacks have been stable for a relatively long time. Standardised ciphers should be used for most uses, but as technology advances, cryptography is deployed on a increasingly varying set of devices. Not all devices, even today, possess the computational power, chip area, energy and/or RAM size to be able to handle the conventional ciphers.

One of the technological advances that exposed the needs of more lightweight block ciphers is the Internet of Things (IoT). The devices that exist in the IoT 'realm' have to be small, cheap and energy efficient. However, these devices should use cryptography to secure the communication. The security of the communication gets even more important when considering that most of the IoT devices communicate wirelessly with each other and on the internet. To handle the new design criteria of this emerging technological field, the cryptographic community has recently designed several lightweight block ciphers.

Two particularly interesting families of lightweight block ciphers are: Simon and Speck. Both have been designed by the same team and have been published in the same preprint by Beaulieu et al.[3]. Opposite to common scientific convention the authors did not provide any design rationale or cryptanalysis of their algorithms. This has led to lots of scrutiny by the cryptanalysis community, nevertheless no full break of either Simon or Speck has been reported yet.

This research focusses on the cryptanalysis of the Simon family of block ciphers. A family of ciphers is a set of ciphers sharing one design using different parameters. Simon deserves and receives more scrutiny by the academic community due to its peculiar design. Speck uses a more standard ARX (Add-Rotate-Xor) construction built onto a Feistel network. ARX uses modular addition as the non-linear component, which has been used in several other (lightweight) block ciphers (e.g. PRINCE [10], HIGHT [23], LEA [24], RC5 [40]). Unlike Speck, Simon uses a less used and researched non-linear component, the logical `and`. The main advantage of the construction used by Simon is that it is more efficient to implement a logical `and` in hardware than a modular addition (due to having to take care of the carry in modular addition which is quite expensive in hardware). One advantage of using modular addition as the non-linear operator instead of the logical `and`, is that modular addition diffuses unlike the logical `and`.

Organisation

This thesis consists of three main parts. The first part, starting at Section 2, provides the background knowledge needed to understand the rest of the thesis. This part also explains some basics of cryptanalysis and block cipher design and describes differential (Section 3) and related key differential attacks (Section 4).

In Section 5 a brief overview of Simon is given.

The second part consists of novel work done for this thesis. In Section 6 I research a distinguisher for k-encryptions with the same key. In Section 7 an algorithm for transforming Simon into an S-box based algorithm is proposed. Section 8 contains a related key differential attack.

The last part (Section 9) contains a discussion of the aforementioned sections. It mainly focusses on the public and academic significance of the research conducted. The discussion also provides some insight in the way the research for this thesis was done.

1.1 Related work

One of the first cryptanalysis papers on Simon was done by Abed, List, Lucks, and Wenzel [1]. This research set a baseline for the papers to come. The 13 round differential characteristic introduced (for Simon32) in that paper is shown to be optimal by an adaptation of Matsui's algorithm [30] in [27]. In [44] the researchers use Mouha et al.'s framework [33], especially the adaptation geared towards bit oriented ciphers [44] is used to compute the maximum number of active components using Mixed Integer Linear Programming. This is used to find fixed and related key differential characteristics and differentials for various lightweight block ciphers including Simon. Nevertheless, no related key differential characteristics are reported for Simon in this study. In [45] the authors mainly focus on the cryptanalysis of the 32 and 48 bit versions of Simon and find a 21 round integral distinguisher by experimental analysis for Simon32.

The work of Biryukov and Perrin [6] on hidden structures influenced significantly the direction of Section 7. While the paper by Patarin [38] on general attacks on Feistel networks influenced the work on Section 6. Although slightly different the authors in [32] find a similar result as in Section 6, while giving a bound on the advantage of the attacker. The preprint by Nandi [34] greatly simplifies the proof in comparison to [32]. In [1] a 14 round related key differential is proposed for Simon32. We found a characteristic with equal length, but with nicer properties that are described in Section 8.

2 Introduction to Cryptanalysis

What is a better way learning cryptanalysis then diving straight into it? This section describes some basic cryptanalysis and cryptography concepts. Using this knowledge a very basic encryption scheme is described, which the reader should be able to dissect and attack using the just acquired skills.

Everyone slightly interested in mathematics or computing science has heard about the Ceasar Cipher or the Vigniere cipher. And although they both have

had their importance in the past, they are nowhere near the state of the art cryptographic systems which are widely employed nowadays. Therefore, a bit breaking with traditions, these cryptographic (toy) systems will not be described in this thesis. What will be described is an easy to understand (slightly) crippled block cipher with which the basic concepts of cryptographic systems and their cryptanalysis are described.

As described earlier the cryptographic primitives can be divided in a couple of basic concepts. The two main concepts for maintaining confidentiality are: symmetric and asymmetric cryptography. This thesis focuses on symmetric cryptography, therefore this introduction will be mainly focusing on symmetric cryptography and its cryptanalysis.

Symmetric encryption can be viewed as a function

$$E : \mathbb{F}_2^n \times \mathbb{F}_2^k \to \mathbb{F}_2^m,$$

where n denotes the plain text size, k denotes the key size and m denotes the cipher text size. This function transforms a plain text, which is a human readable text, into a cipher text using a key.

Remark 1. In the literature in the field of cryptography as well as in this thesis the plaintext is often denoted by p. The ciphertext and key material are often denoted by c and κ respectively . $\mathbb{F}_2^n$ denotes the set of all n-bit words. Another way to denote n-bit words would be $\{0,1\}^n$. The $\texttt{xor}(\oplus)$ between two elements of $\mathbb{F}_2^n$ is the bitwise modular addition of the two elements. The logical $\texttt{and}(\cdot)$ can be seen as the bitwise multiplication of the two elements.

For encryption to be useful a function to unravel the encryption is needed. This function is named decryption. Symmetric decryption can be described as the following function:

$$D : \mathbb{F}_2^m \times \mathbb{F}_2^k \to \mathbb{F}_2^n$$

such that:

$$D(E(p,\kappa),\kappa) = p \text{ for } p \in \mathbb{F}_2^n \text{ and } \kappa \in \mathbb{F}_2^k.$$

Following the Kerckhoffs' principles [25] the only secret part of a cryptographic system should be the key. A very simple 'unbreakable' symmetric cipher, the One Time Pad (OTP) only uses the xor operator. The encryption and decryption functions are the same and are defined by:

$$E(p,\kappa) = D(p,\kappa) = p \oplus \kappa.$$

It is straightforward to check if this is sound (i.e. decryption is the inverse of encryption):

$$D(E(p,\kappa),\kappa) = ((p \oplus \kappa) \oplus \kappa) = p \oplus \kappa \oplus \kappa = p.$$

As Shannon described in 1949 the one time pad is information theoretically secure, which means that no matter how much computing power is used, the

message cannot be deciphered without possession of the key [43]. Nevertheless there are various problems in using OTP as a cipher. The first problem is that the key has to be as long as the message. This shifts the problem of sending the message in secrecy to sending the key in secrecy. Of course, this could be overcome by for example burning a large chunk of random data onto a (or two) cd-rom(s) and exchanging the cd-rom in person. Afterwards the two parties can communicate using this data. This may be useful for low bandwidth high risk/secrecy communication (such as the communication between the Kremlin and the White House), but using such a cipher to encrypt your e.g. YouTube video stream is unpractical and quite costly.
(it would be a funny first of april joke, "Now introducing: OTP, the new standard. One key, One time, One DVD for all your communications")

The second practical problem with OTP, as the name suggests, is that the key may only be used once. Otherwise the whole secrecy breaks due to statistical analysis.

Remark 2 (Reusing the key in OTP)**.** Say we have two messages encrypted using OTP $c_1 = m_1 \oplus \kappa$ and $c_2 = m_2 \oplus \kappa$ and assume we know that the distribution of m_1, m_2 is not uniform (e.g. the english language), then we do know the distribution of $m_1 \oplus m_2$. If we now `xor` the two ciphertexts we get: $c_1 \oplus c_2 = m_1 \oplus m_2$. Which makes it easier to break, since the probability that two `e`'s overlap each other is higher than two `q`'s.

The third problem of OTP is that it is only information theoretically secure when used with true random key. True random data is hard and expensive to come by, you need specialised hardware to generate it and even then the throughput of the random data is quite low. This leads to another scalability problem.

To tackle the problems mentioned above, current ciphers do use a combination of two concepts, diffusion and confusion to reach a certain guarantee of confidentiality [43]. These two concepts are used to hinder statistical analysis of the cipher. Diffusion ensures that there are lots of dependencies between 'blocks' in the cipher text and due to confusion these dependencies are hard to analyse statistically. One of the side effects of using these concepts is that the so called avalanche effect can be observed. This causes, given a change of one part in the plain text, a change in all parts of the ciphertext with a uniform probability.

Another concept which is employed by most contemporary ciphers is dividing the amount of work into smaller chunks. So instead of one function that performs all the work needing to be done to encrypt the plain text, designers tend to use a smaller function that does a small amount of work and iterate that function to ensure proper encryption. There are three reasons to do so, the first reason is that designing and analysing a smaller function that is iterated is often easier then designing a function that has to do all the work at once. The second reason is that by using a round based scheme proper diffusion of the key material is easier to accomplish. The third reason is that by reducing the

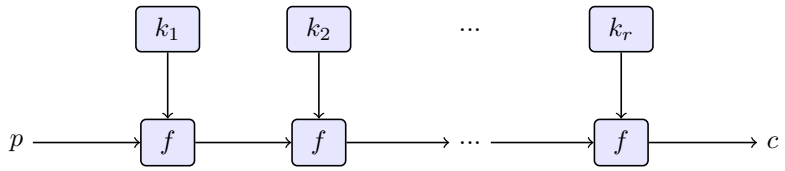

Figure 2: A round based encryption function, with and r rounds and roundkeys $k_1, k_2, ..., k_r$.

number of rounds the analysis of the cipher is easier. By breaking these round reduced versions in a certain way the strength of a cipher is measured.
There are roughly two round based schemes used in the contemporary cryptographic designs: Substitution Permutation Networks (e.g. AES) and Feistel networks (DES, Simon). In this thesis I will mostly focus on the Feistel networks, since Simon is a (traditional) Feistel network.

Definition 1 (Round based Cipher)**.** Let $f : \mathbb{F}_2^n \times \mathbb{F}_2^k \to \mathbb{F}_2^n$ be the round function, then the encryption function can be defined as (see Figure 2 for a visual representation):

$$E(p, \kappa) = f \circ f \circ ... \circ f(p, \kappa) = f(f(...f(p, \kappa), ...\kappa), \kappa)$$

Most of the time the amount of key material needed for the round keys is larger than the amount of key material in the master key. To solve this problem a key-schedule is designed that expands the key. Note that although the master key is expanded the entropy of this expanded key is still equal to or less than the entropy of the master key. The key schedule described here is not safe.

Feistel networks [28] have been used for symmetric cryptography for almost four decades. One of the first (widely used) symmetric ciphers using a Feistel network is DES [17]. Since then many ciphers have been designed using a (generalised) Feistel cipher. (One Feistel round is depicted in Figure 3)

Definition 2 (Feistel network cipher)**.** A Feistel cipher with r rounds can be described as follows: Let $p = L_0|R_0$ and let the internal state after the i-th round be described by :

$$L_{i+1} = R_i \oplus f(L_i, k_i)$$
$$R_{i+1} = L_i$$

The decryption of a Feistel cipher is computed by starting (L_r, R_r) and going back to (L_0, R_0) by computing:

$$L_i = R_{i+1}$$
$$R_i = L_{i+1} \oplus f(R_{i+1}, k_i)$$

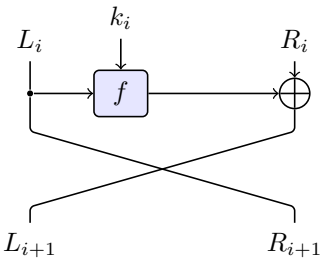

Figure 3: General Feistel network round function

2.1 Linear Toy Cipher: LTC

To practise the just obtained knowledge a cipher using a Feistel cipher is described and an analysis of its cryptographic strength is done. Note that since this is an introduction into cryptanalysis the cipher described is far from perfect, making it easy to break. Please do not see this cipher stated in this document to be any form of approval of such a cipher and therefore please do not use it other than as an attack target.

Remark 3 (Why not to use LTC, NTC or CTC)**.** I especially designed LTC, NTC and CTC such that the cryptanalysis would be near to trivial. Considerable effort has been made in choosing the constants, key schedule, and S-boxes such that the ciphers seem strong at first sight, but can be easily broken by different techniques. Although the name bears resemblance to the cipher named Toy Cipher by Courtois [13] this is merely a naming collision.
These ciphers are designed such that the reader understands Simon better as that is the target of this thesis.

As defined in Definition 2 a Feistel cipher can be described by a function f and a key schedule. The cipher defined in this section in and outputs 32-bits blocks of data, uses a 64-bits key and uses 12 rounds to encrypt the data. First the key schedule is defined:

Definition 3 (TLC Key schedule)**.** Let master key $K \in \mathbb{F}_2^{64}$ then round key k_i is defined as:

$$k_i = (K \ggg 16i) \bmod 2^{16}$$

This results in the following key schedule which is greatly influenced by the key

schedule used by RoadRunneR [2]:

$$\begin{aligned} K &= A|B|C|D \quad \text{with } A, B, C, D \in \mathbb{F}_2^{16} \\ k_0 &= A \\ k_1 &= B \\ k_2 &= C \\ k_3 &= D \\ k_4 &= A \\ &... \\ k_{11} &= D \end{aligned}$$

The round function f is described by:

$$f(x, k_i) = (x \lll 7) \oplus (x \lll 2) \oplus k_i$$

For a visual representation of the first four rounds of the cipher see Figure 4. The pseudocode of the cipher is:

Algorithm 1 TLC Encryption

roundkeys = $[(K \ggg 16i) \bmod 2^{16}$ for $0 \leq i < 12]$
$L|R$ = plaintext
for $0 \leq i < 12$ **do**
  $L' = L$
  $L = R \oplus f(L,\text{roundkeys[i]})$
  $R = L'$
end for
ciphertext = $L|R$

Exercise 1. *Try to write down the pseudo code for the decryption function of TLC.*

Remark 4 (Encoding)**.** Most ciphers act on bit strings (actually there are ciphers defined on integers which are described by Black and Rogaway [8]). Writing down, and reading, these bit strings is quite tedious. This is solved by encoding all raw in- and outputs in hexadecimal encoding. Not only is hexadecimal encoding shorter than using bit or decimal encoding, it is also easier to see structures since each character is encoded by exactly four bits. Unless stated otherwise all keys, plaintexts and ciphertexts are encoded with hexadecimal encoding (essentially all data in this thesis typefaced in a fixed width font is encoded in hexadecimal encoding).

Using this encryption algorithm we get the following plaintext/ciphertext pairs when encrypting with key `1234 5678 9012 3456`: At a first glance no

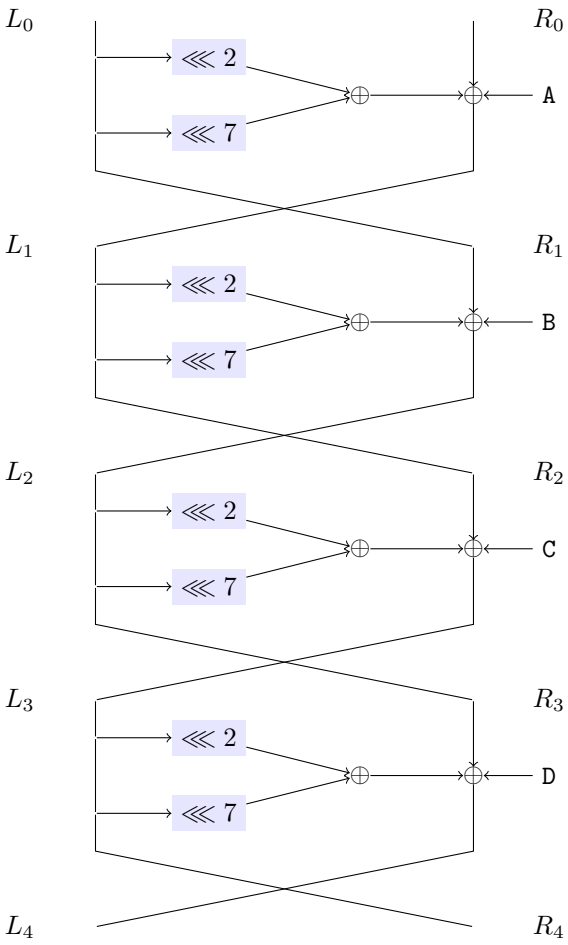

Figure 4: First four rounds of TLC with $K = \texttt{A|B|C|D}$

plain text	cipher text
0000 0000	3B5A 511B
0000 0001	5A52 514A
0000 0002	F94A 51B9
0000 0003	9842 51E8
0000 0004	BF7B 505F
0000 0005	DE73 500E
0000 0006	7D6B 50FD
0000 0007	1C63 50AC
0000 0008	3319 5393
0000 0009	5211 53C2

Table 1: 10 plaintext ciphertext pairs for the TLC with key: `1234 5678 9012 3456`

obvious structure can be found in the ciphertexts, even when the plain text is obviously structured. Can we now conclude that this a strong cipher? In the next section the cipher is broken by what is called the linearisation attack.

2.2 Breaking the Toy Linear Cipher

As can be observed from the definition of TLC the cipher consists of only linear operations.

Exercise 2. *Verify that $\lll$ and $\oplus$ are both linear operations.*

By carefully analysing the expansion of the internal state of the first couple of rounds, it is easy to see that the key material can be separated from the message.

$$
\begin{aligned}
L_1 &= (L_0 \lll 7) \oplus (L_0 \lll 2) \oplus R_0 \oplus k_0 \\
R_1 &= L_0 \\
L_2 &= ((L_0 \lll 7) \oplus (L_0 \lll 2) \oplus R_0 \oplus k_0) \lll 7) \oplus ((L_0 \lll 7) \oplus (L_0 \lll 2) \oplus R_0 \oplus k_0) \lll 2) \oplus R_1 \oplus k_1 \\
&= (L_0 \lll 14) \oplus (L_0 \lll 9) \oplus (R_0 \lll 7) \oplus (k_0 \lll 3) \oplus (L_0 \lll 9) \oplus (L_0 \lll 4) \oplus (R_0 \lll 2) \oplus (k_0 \lll 2) \oplus L_0 \oplus k_1 \\
&= ((L_0 \lll 14) \oplus (R_0 \lll 7) \oplus (L_0 \lll 4) \oplus R_0 \oplus (L_0 \lll 2)) \oplus ((k_0 \lll 7) \oplus (k_0 \lll 2) \oplus k_1)
\end{aligned}
$$

Note that in the last expression the part on the left only depends on the plain text and the right part only depends on the key. This observation can be used to break the cipher. The key contribution can be computed by encrypting **0**

$$C = TLC_{encrypt}(0, \kappa).$$

This constant C can now be used to encrypt and decrypt without knowing the key by:

$$TLC_{encrypt}(p, \kappa) = TLC_{encrypt}(p, 0) \oplus C$$

and decryption:

$$TLC_{decrypt}(c, \kappa) = TLC_{decrypt}(c \oplus C, 0)$$

Note that for both encryption and decryption one does not need the key, but one does need to obtain the encryption of one chosen plaintext (in this case `0`) for this attack to succeed.

This cipher is obviously broken due to it only consisting of linear components, there are multiple other ways (which take advantage of the same weakness to break this cipher). Nevertheless this attack is by far the easiest to understand and implement. To avoid this weakness one needs to introduce some form of non-linearity in the cipher. In the next section a cryptosystem is introduced which does employ a non-linear component.

Exercise 3. *Prove that LTC with 64 rounds is equal to the identity regardless of the key used.*

2.3 Non-linear Toy Cipher: NTC

To counteract the weaknesses of TLC this cipher contains a source of non-linearity. One of the most used, analysed and understood sources of non-linearity are Substitution-boxes (S-boxes). S-boxes are functions that substitute one value for another. The S-box used in the design of this cipher is:

$$S = [0, 2, 3, A, 6, F, E, 1, 8, B, 9, C, 4, 5, 7, D]$$

This is a 4×4-bits S-box as its in-and output is 4 bits (e.g. $S(5) = F$). This means that if we want to use S directly on a word in the cipher we need the function $\bar{S} : \mathbb{F}_2^{16} \to \mathbb{F}_2^{16}$. Let $\bar{S}$ be defined by:

$$\bar{S}(x) = S(x_1)|S(x_2)|S(x_3)|S(x_4),$$

for $x = x_1|x_2|x_3|x_4$.

Recall the Feistel network cipher of the previous section and recall that a Feistel network cipher is defined by its key-schedule and its round function. This cipher reuses the key-schedule of LTC see Definition 3. The round function f is defined as:

$$f(x, k_i) = \bar{S}(x) \oplus k_i$$

The function f applies S to every 4-bit nibble in the word and then xors the key with the result. See Figure 5 for a visual representation of the round function. The number of rounds in NTC is 12, which is the same as in LTC

The algorithm used in breaking the LTC does not work in this case since the cipher is using a non-linear component to 'hide' the key material. I.e. if we write down for the i-th nibble x and k_0 to k_r, then what happens after r rounds is:

$$S(S(...S(S(x_i) \oplus (k_0)_i)...) \oplus (k_r)_i), \text{ for } 1 \leq i \leq 4$$

Since S is non-linear the following inequality holds in general:

$$S(x \oplus y) \neq S(x) \oplus S(y)$$

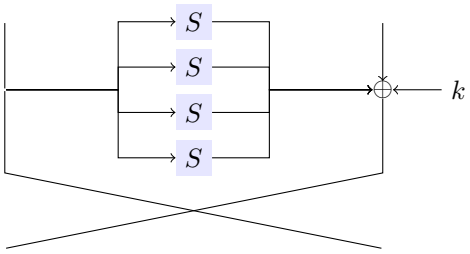

Figure 5: Round function of NTC

Thus the key bits cannot be separated from the plaintext bits as in the linear case.

One exploitable property of this cipher is that one nibble is only affected by its own bits and the key material, (i.e. a change in one nibble of the plain text will only affect the nibble with the same index in the cipher text). This allows for an attack on the cipher by brute forcing all possible sub-keys for each nibble. Each nibble only is affected by 4 bits of the round key at a time, thus for each nibble we need to try at most 48 bits of key material, assuming random round keys, (i.e. NTC has 12 rounds and since every nibble can be broken individually, 4 bits of key material need to be brute forced per round). This basic analysis assumes that there is no key schedule. Since we have 4 nibbles, this would lead to an an attack using 2^{50} computations. Note that this is smaller than the original workload of 2^{64} encryptions, but we can do better.

Recall that in the key schedule the number of key bits influencing each nibble in the encryption path is 16. This results in a brute force attack with a total of 2^{18} partial encryptions (a number that most scientific calculators can do in a matter of minutes/seconds).

2.4 The Combined Toy Cipher: CTC

As we have seen, the cipher with only linear components as well as the cipher with only non-linear components can be easily broken. As Shannon already noticed in 1949 we need both confusion and diffusion to have a cipher that is cryptographically strong (we need more, but that is for a later section). The linear layer that is used in the LTC provides diffusion, but the resulting function is easily analysed since all relations in the resulting ciphertext are linear. On the other hand the non-linear cipher, NTC, had statistically hard to analyse relations in the resulting cipher text. But since there was no diffusion and therefore the bits only had local influence, the cipher was 'easily' brute forced (and due to the 'very interesting' key-schedule it was even easier).

To solve the above problems we devise a new cipher combining a linear and a

non linear layer. For the analysis in this chapter we will use a (slightly) weaker linear layer not to complicate things too much. This results in the following round function used in our Feistel network.

$$f(x, \kappa) = \bar{S}(x) \ggg 2 \oplus \kappa$$

The key schedule used is the one used in the previous ciphers and the number of rounds is 12.

This cipher is not breakable with the techniques discussed until now. In the next section a technique is proposed that can be used to break this cipher.

3 Differential Cryptanalysis

When Eli Biham and Adi Shamir discovered differential cryptanalysis [4] and disclosed it to the public in 1990 the world was shocked. The discovery made it possible to attack ciphers that researchers were not able to break before. DES, the standard block cipher of that time was broken soon after by using differential cryptanalysis [5]. Differential cryptanalysis is still the powerful technique it was almost four decades ago. It is used in many attacks against contemporary ciphers. By using adaptations, the technique can manoeuvre itself through the defences of the modern ciphers. Nevertheless, most of the currently used ciphers do not see a full break by the use of differential cryptanalysis. Researchers attack round reduced versions of the ciphers instead, therefore most attacks only serve as a 'measure' of the strength of a cipher and cannot be used in a practice. As we have seen in the previous sections modern ciphers try to withstand attacks by using diffusion and confusion. This makes it such that the statistical relationship between the plaintext and ciphertext is hard to unravel and dependent on all input bits of the function.
Recall the S-box used in the previous section:

$$S = [0, 2, 3, A, 6, F, E, 1, 8, B, 9, C, 4, 5, 7, D].$$

If the input to the S-box is uniformly distributed the output of this S-box is uniformly distributed as well.
By analysing the difference of two inputs and the difference of the accompanying outputs we can see that two different input differences can map to the same output difference. Inputs $1, 2$ have an input difference of $1 \oplus 2 = 3$ and an output difference of $S(1) \oplus S(2) = 2 \oplus 3 = 1$. We can get the same output difference if we take as inputs 3 and 9 ($3 \oplus 9 = A$) which will result in: $S(3) \oplus S(9) = A \oplus B = 1$. This shows that the mapping between input and output differences is not bijective and this leads to a mapping whose outputs may not be uniformly distributed when the inputs are chosen uniformly.

This property can be used to attack the S-box such that a relationship between the input and output of the cipher can be found. To be able to easily compute such a relationship a Difference Distribution Table (DDT) for this S-box is computed. The DDT describes the probability of a differential transition

through the S-box. To compute the DDT every possible input pair of the S-box is input and the number of different output differences for a certain input difference are counted (see Algorithm 2).

Algorithm 2 Compute the DDT

```
Let S be the S-box to be analysed
Let DDT be a |S| × |S| dimensional array initialised to 0
for All possible input differences diff do
  for All inputs i do
    let outputDiff = S(i) ⊕ S(i ⊕ diff)
    DDT[diff][outputDiff]++
  end for
end for
return DDT
```

In the DDT the rows denote the input differences and the columns denote the output differences coupled to the input difference. The row with index 2 defines how many times each output difference occurred when the input difference 2 was fed into the S-box. Note that every row in this DDT sums up to $16 = 2^4$, since there are 16 different pairs of inputs that lead to a certain input difference. As we can see in the DDT (Table 2) when the input difference is 5 the output difference is 4 with probability $\frac{6}{16}$. The differential uniformity of a S-box is equal to the highest number in the DDT, except for the entry with input 0. Differential uniformity is used as a measure for how well the S-box performs with respect to differential cryptanalysis, the higher the differential uniformity the stronger the strongest possible attack is. The differential uniformity of the S-box used in CTC is 10.
The most interesting (highest) entries of the DDT are marked in blue. In Section 3.1 this non-uniformity is used to attack the cipher.
Note that if an input difference is 0, the output difference is always 0, since having an input difference of 0 means that the two inputs are equal and thus the outputs are equal.

Remark 5 (Bent functions and 'perfect' S-boxes)**.** The S-box in this section has been designed such that the resulting DDT contains high entries. This way finding a differential characteristic with high probability would become easier. Of course in a real cipher one should construct the S-box in such a way that the highest value in the DDT is as low as possible.
The question immediately arises if it would be possible to create an S-box such that the DDT generated by the S-box would be uniform. This topic has been studied by Nyberg [35] and many other after that. By using bent functions such S-boxes can be designed, but they are not practical since the number of input bits needs to be twice the number of output bits. In [36] Nyberg describes how to control the linearity and differential uniformity of S-boxes.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	2	2	2	0	2	0	0	0	4	2	0	0	0	0	2
2	0	2	0	4	0	0	0	2	6	0	0	0	0	0	2	0
3	0	4	4	0	2	0	0	2	0	2	2	0	0	0	0	0
4	0	2	0	0	0	0	2	0	0	0	0	2	2	4	4	0
5	0	0	2	0	6	0	0	0	0	0	0	2	0	2	0	4
6	0	0	0	2	0	4	2	0	0	2	0	0	0	2	2	2
7	0	2	0	0	0	2	0	0	2	0	0	0	10	0	0	0
8	0	0	2	0	0	0	2	0	2	4	4	0	2	0	0	0
9	0	0	0	6	0	0	2	0	0	0	2	4	0	0	0	2
A	0	4	2	0	2	0	0	0	0	2	2	2	0	0	2	0
B	0	0	2	0	0	2	0	0	4	0	0	6	2	0	0	0
C	0	0	0	0	6	0	0	6	0	0	0	0	0	2	2	0
D	0	0	2	0	0	2	2	2	2	0	0	0	0	4	2	0
E	0	0	0	2	0	0	2	4	0	0	2	0	0	0	0	6
F	0	0	0	0	0	4	4	0	0	2	2	0	0	2	2	0

Table 2: The DDT of the S-box. The values denote the number of times the output difference (column) occurred given a certain input difference (row).

3.1 Differential Characteristics of CTC

Looking at the DDT of our S-box (Table 2) a couple of high numbers can be discovered. In this particular instance they should be easy to find, since they are coloured red and blue in the table. Using these numbers so called differential characteristics can be constructed, which can be seen as a couple of differential transitions leading into each other. One such characteristic, although a pretty bad one, is given in Figure 6.

In the differential setting it is meaningful to overload S to be able to handle difference input, this is called a differential transition and is associated with a probability that the transition occurs. For example the differential transition $S(\Delta\texttt{0009}) = \Delta\texttt{0003}$ with probability $\frac{6}{16}$ and the differential transition $S(\Delta\texttt{0000}) = \Delta\texttt{0000}$ with probability one. What happens in this characteristic is that in the first round we have $S(\Delta\texttt{0007}) = \Delta\texttt{000C}$ with a probability of $\frac{10}{16}$. Then the linear function, right rotation by two, transforms $\Delta\texttt{000C} \ggg 2 = \Delta\texttt{0003}$. This is xored with the right part and the key difference (which are both 0) and swapped. This results in the left and right parts in the second round to be: `0003` and `0007`. Note that the difference behaves well under rotation.

In Figure 6 the characteristic is shown. As can be seen the probability this characteristic occurs is $\frac{10}{16} \cdot \frac{2}{16} \cdot \frac{2}{16} \approx 2^{-6.678}$ which is almost 1 in a hundred times. In the next part of this section a characteristic with better probability and better properties is created.

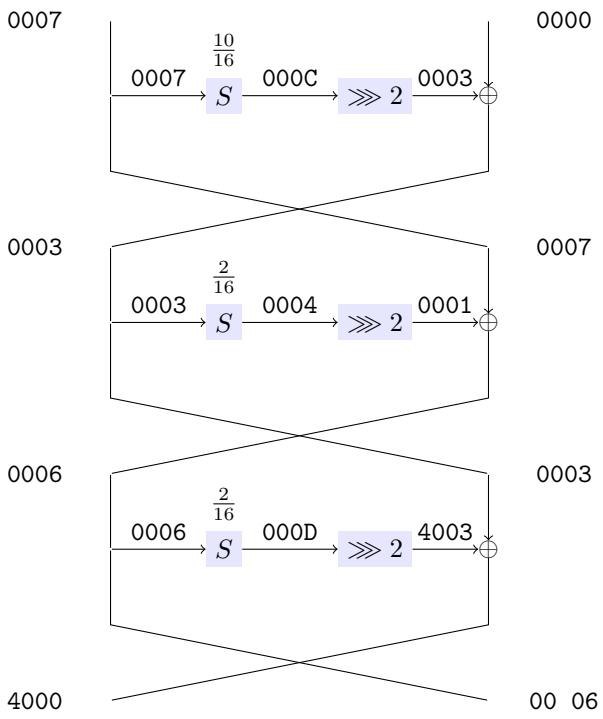

Figure 6: An example of a (bad) differential characteristic for CTC. In this figure the Δ for denoting differences is omitted for clarity reasons.

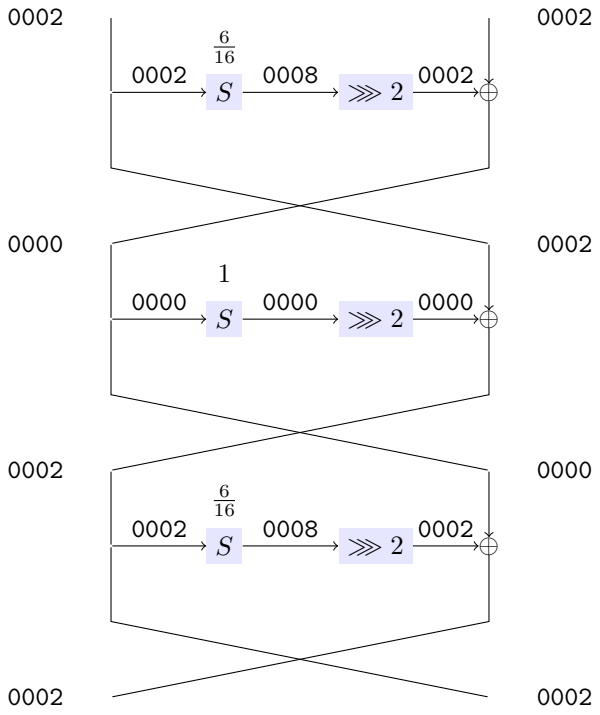

Figure 7: The differential characteristic used for the key recovery attack on CTC. In this figure the Δ for denoting differences is omitted for clarity reasons.

Due to the relative simplicity of this cipher a differential characteristic can be found which has at most 1 active S-box in each transition. This characteristic is (see Figure 7 for a visual representation of this characteristic):

$$0002\ \ 0002 \xrightarrow{\frac{6}{16}} 0000\ \ 0002 \xrightarrow{1} 0002\ \ 0000 \xrightarrow{\frac{6}{16}} 0002\ \ 0002$$

Note that the input and output of this characteristic is the same. This kind of characteristic is called an iterative characteristic. Due to the in- and output of the characteristic we can chain them together, hence the name.
The probability of this characteristic occurring is:

$$\frac{6}{16} \cdot 1 \cdot \frac{6}{16} = \left(\frac{3}{8}\right)^2 \approx 2^{-2.83}$$

There are 2^{32} different pairs of inputs giving the difference 0002 0002. Each pair follows the characteristic with probability $2^{-2.83}$. Which means that after one iteration of the characteristic there are $2^{32} \cdot 2^{-2.83} \approx 2^{29.17}$ pairs that have the correct output difference. After two iterations there will be approximately

$2^{32} \cdot 2^{-2.83} \cdot 2^{-2.83} \approx 2^{26.34}$ pairs left with the right output. The number of iterations of the characteristic we can cover until the expected number of correct pairs left is > 1 is 11.

Since the iterative characteristic spans 3 rounds, this gives us a differential characteristic of 33 rounds with a probability of:

$$(2^{-2.83})^{11} = 2^{-31.13} > 2^{-32}$$

This probability gives us an expected number of correct pairs of approximately $2^{0.87} \approx 1.87$ which is larger than 1 and therefore can be used as a distinguisher. An attentive reader would however have figured out that the cipher in question only uses twelve rounds. To cover twelve rounds only four iterations of the characteristic are needed which leads to the differential probability:

$$(2^{-2.83})^4 = 2^{-11.32}$$

To get an expected number of correct pairs of 1 for the 12 rounds characteristic we need approximately $2^{12} = 2556$ chosen plaintext ciphertext pairs.

3.2 Key recovery

Having a distinguisher for a cipher is nice and in most cases one should not use a cipher for which a distinguisher exists. Nevertheless, having a key recovery attack for a cipher is even nicer.

In this part a key recovery attack against CTC using the distinguisher from Section 3.1 is described.

The key recovery attack can be divided into three parts:

Pair collection In the pair collection phase a set of plain text pairs with the given input difference and their accompanying cipher text pairs are collected.

Pair filtering In the pair filtering phase all pairs with the wrong output difference are filtered out.

Key guessing In the key guessing phase some bits of the key are guessed and verified by using the differential characteristic.

The first two phases are quite straightforward, so this section elaborates on the last phase, key guessing. The main idea in the key guessing phase is to guess the part of the key influencing the active S-box and then checking if the differential characteristic after the S-box is still in accordance to the differential characteristic.

Note that in the case of CTC the last key is added after the S-box. The contribution of the last round key disappears in the difference, thus in the case

of CTC it is necessary to roll back two rounds. A table with keys to be guessed is initialised to 0 and for every correct pair the pair is partially decrypted. If the difference after the partial decryption is in accordance to the differential characteristic the counter for the guessed key is increased by one. The key with the highest counter is the most probable key.

In the following section the Key Guessing phase for CTC is designed. Let the ciphertext pairs be denoted as:

$$(C, C') = (L_{12}|R_{12}, L'_{12}|R'_{12}) = (L_{12}|L_{11}, L'_{12}|L'_{11})$$

and let the round keys for round 11 and 12 be denoted as: k_{11} and k_{12}.
Recall that the differential characteristic only affects the four least significant bits of each block which will only activate one S-box. This means that only the four least significant bits of the round keys can be determined.

Let K_{12} and K_{11} be the sets of possible round keys for round 11 and 12 with only the four least significant bits set. Note that each set contains 16 elements. Now for every combination of possible round keys $rk_{11} \in K_{11}$ and $rk_{12} \in K_{12}$ verify for every ciphertext plaintext pair L_{10}, L_{11} and L'_{10}, L'_{11}, where $F(x) = \bar{S}(x) \ggg 2$:

$$\begin{aligned} L_{11} &= F(R_{12}) \oplus L_{12} \oplus k_{12} \\ L'_{11} &= F(R'_{12}) \oplus L'_{12} \oplus k_{12} \\ L_{10} &= F(R_{11}) \oplus L_{11} \oplus k_{11} \\ L'_{10} &= F(R'_{11}) \oplus L'_{11} \oplus k_{11} \end{aligned}$$

and then verify that the following two equations hold:

$$\begin{aligned} \texttt{0000} &= L_{11} \oplus L'_{11} \\ \texttt{0002} &= L_{10} \oplus L'_{10} \end{aligned}$$

For every pair for which the above two equations hold increase the counter of the key by 1. After analysing all pairs and keys the key with the highest counter is the most probable key.
Once the round keys for round 11 and 12 have been determined the same tactic can be used to determine the round keys for round 9 and 10. Due to the key schedule, we have at this point recovered the four least significant bits of each block in the master key.
To recover the remaining bits we could either brute force the remainder, which would cost 2^{48} encryptions. Or we could take advantage of the fact that the following variations of the given differential characteristic exist:

```
0002 0002
0020 0020
0200 0200
2000 2000
```

Using these characteristics the full key recovery is straightforward.
So how much does this attack cost? To have enough ciphertext pairs to pull of the attack (and we are erring on the safe side here) we will need 2^{15} chosen plain text pairs, which will net an expected 12 correct pairs. Then if choosing to go for brute force attack the total cost would be 2^{15} plaintext pairs and 2^{48} computation time.
If the other characteristics are used the number of chosen plaintext pairs would be $4 \cdot 2^{15} = 2^{17}$ while reducing the computation time to $4 \cdot 2^8 \cdot 4 \cdot 12$ which is negligible (for every correct pair (12) try all possible round keys (2^8) for four rounds and do this four times).

3.3 Hardening the cipher against differential Cryptanalysis

There are a couple of ways to harden CTC against differential cryptanalysis. One is to use an S-box with a more uniform DDT, reducing the differential uniformity of the S-box. This would reduce the probabilities of the differential transitions succeeding, e.g., by using the AES S-box or a smaller S-box with a similar differential uniformity. Exchanging the S-box does, in this case, not negatively impact the cipher's complexity as long as the S-box is substituted with a similar sized S-box. Note that an S-box could be engineered in such a way that it has specific properties that increase performance.

Increasing the number of rounds is another way of hardening the cipher against differential cryptanalysis. The differential characteristic found is effective for up to 34 rounds, thus increasing the number of rounds to 60 should protect the cipher against this particular characteristic and probably more characteristics. However, increasing the number of rounds does negatively impact the performance of the cipher.

The last way to harden the cipher against differential cryptanalysis is to make the linear part in the round function more robust. This facilitates to increasing the minimal number of active S-boxes. Combined with the differential uniformity this is often given as a argument for an upper bound of the number of rounds attackable with differential cryptanalysis. This is not always true due to a cipher not being a Markov process. This is especially the case in lightweight ciphers.

Exercise 4. *Try to find a differential attack for the Feistel cipher with 12 rounds and the round function, being:*

$$F(x, \kappa) = (S(x) \lll 7) \oplus (x \lll 2)$$

round	difference	key difference	transitional prob.
1	0002 0002	0000	$\frac{6}{16}$
2	0000 0002	0002	1
3	0000 0000	0000	1
4	0000 0000	0000	1
5	0000 0000	0000	1
6	0000 0000	0002	1
7	0002 0000	0000	$\frac{6}{16}$
8	0002 0002	0000	$\frac{6}{16}$
9	0000 0000	0000	1
10	0002 0000	0002	$\frac{6}{16}$
11	0000 0002	0000	1
12	0002 0000	0000	$\frac{6}{16}$
output	0002 0002		

Table 3: The Related Key differential characteristic for CTC

4 Related Key differential cryptanalysis

Can we do better then a differential attack? In the differential attack a difference is introduced into the plaintext pairs. In certain circumstances a difference could also be introduced in the key. This can be used to gain some extra attack surface. One such circumstance would be that an oracle allows the attacker to flip bits of the key and rerun the attacks. Whether this extra attack surface leads to a better attack is mostly dependent on the key schedule used.

To analyse how CTC behaves in the related key setting the starting point is the differential characteristic described in Section 3.1. This characteristic worked with a key difference of 0000 0000 0000 0000. To enhance this characteristic one could for example try to negate the 0002 difference in the second round by assuring that the key difference in the second round is 0002. By employing the key difference 0000 0002 0000 0000 the characteristic as in Table 3 is possible: The characteristic has a probability of $(\frac{6}{16})^5 \approx 2^{-7.08}$, which is larger than the iterated characteristic described in the previous section. This characteristic would allow us to mount the same key recovery attack as in the previous section but with, only 2^{11} chosen plaintext ciphertext pairs.

4.1 Hardening CTC against Related Key differential attacks

Note that by hardening the cipher against differential attacks in the fixed key model it is often also hardened in the related key model. However, this does not provide full protection against related key differential attacks. To fully harden the cipher against related key differential attacks, the key schedule should be updated such that the round keys cannot be easily manipulated and no cycles

occur in the key schedule.

To accomplish this, several methods are employed, from adding round constants to reusing the round function for the key schedule as in Simeck [46]. The key schedule of Simon is mainly linear, which makes it easy to reason about the smallest cycles possible, see Section 8.1 for an analysis.

Exercise 5. *try attacking CTC with the following key schedule:*

$$K = k_0|k_1|k_2|k_3$$
$$k_{i+4} = k_i \lll 1 \oplus k_{i+3} \lll 2$$

Please note that this scheme is weak on purpose, so even with all the hardening explained and applied this should never be used in any system.

5 Simon

Simon [3] is a family of lightweight block ciphers defined on a variety of block and key sizes. Simon is geared towards hardware implementations and is therefore constructed from hardware friendly components. It uses rotations for diffusion and bitwise logical `and` as the non-linear component.

Simon has met some controversy by the cryptographic community due to its non-standard design and the lack of publicly available cryptanalysis or design rationale by its authors. This has led to a great amount of cryptanalysis papers targeting Simon over the last four years.

This amount of cryptanalysis has become an argument used by the authors and standardisation committees to standardise Simon. Until now [mid May 2017] all but the 128-bit versions of Simon have been retracted from standardisation, i.e. in ISO JCT1/SC27, and there exists no reason to believe the remaining version will be standardised by NIST in the near future. Nevertheless, the industry is looking for a lightweight alternative either for the advantages or to have a next best thing. This should be an incentive for the cryptographic community and standardisation organisations to find a suitable new standard. In case no suitable lightweight cipher is brought forward the industry will use the most advertised and most cryptanalysed lightweight cipher, which at this moment in time would be Simon.

5.1 Simon structure

As stated before Simon is a family of lightweight block ciphers. The different versions and their parameters are described in Table 4. Simon is a Feistel

Block size $2n$	Key size mn	Word size n	Key words m	Rounds
32	64	16	4	32
48	72	24	3	36
	96		4	36
64	96	32	3	42
	128		4	44
96	96	48	2	52
	144		3	54
128	128	64	2	68
	192		3	69
	256		4	72

Table 4: Simon versions

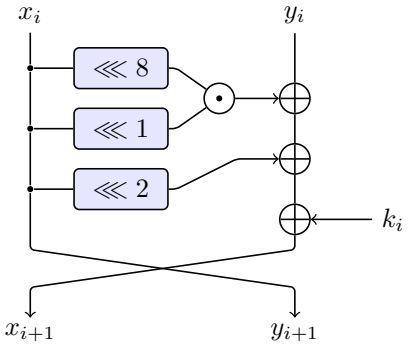

Figure 8: Simon round function

network with two branches using the following round function:

$$f(x, k_i) = (x \lll 2) \oplus (x \lll 1 \texttt{ and } x \lll 8) \oplus k_i$$

where x is a word half the block size and k_i is the round key for round i (see Figure 8 for a visual representation of Simon's round function).

Simon uses four different key schedules depending on the number of key words (m). In this thesis mainly the keyschedule for $m = 4$ is analysed. The key schedule with four key words can be described by:

$$k_{i+4} = c \oplus (z)_i \oplus k_i \oplus (k_{i+1} \ggg 1) \oplus (k_{i+1} \ggg 4)$$

Where $c = -3 = 2^n - 3$ and z is a version dependent constant to make a cryptographic separation between different versions of Simon. See [3] for the definition of z and the two other key schedules.

6 Distinguishing t-encryption

Assume the following scenario: Alice wants to send Bob a secret message. She does this by encrypting the message with a secret key which only she and Bob know. They however do not trust the cipher they are using and decide to encrypt the message twice with the same key. Does this tactic (apart from obviously wasting cycles) harm Alice and Bob?

It is clear that this tactic does not significantly increase security, but can it impact the security negatively? In this section the negative impact of such a tactic is shown in the general case. Then experimental results of double encryption on Simon32 [3] are discussed.

6.1 Background knowledge

6.1.1 Permutations

The main operation discussed in this section is t-encryption, which means encrypting t times with the same key iteratively. To be able to reason about what happens when performing t-encryption, let us first have a look at happens when a double encryption ($t = 2$) is performed. Recall that encryption can be seen as a random permutation. Let $\mathcal{P}$ be the set of all permutations and let $C : \mathbb{F}_2^n \times \mathbb{F}_2^k \to \mathbb{F}_2^n$ denote encryption of an n-bit word with a k bit key. Now C can be seen as a generator for a subset of $\mathcal{P}$, such that $C_\kappa(p) = C(p, \kappa)$ is the encryption under one key and $C_\kappa \in \mathcal{P}$ for every $\kappa \in \mathbb{F}_2^k$.
A permutation can be written down as a cycle decomposition of the permutation. The following permutation:

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 3 & 2 & 8 & 9 & 4 & 6 & 10 & 5 & 1 & 7 \end{pmatrix}$$

which maps the element 1 to 3, 3 to 8, 8 to 5, etc. can be written down as:

$$(1, 3, 8, 5, 4, 9)(2)(6)(7, 10)$$

As can be seen in the cycle representation of the permutation, the permutation has 4 cycles of which two are two fixed points (6 and 2) which are considered cycles of length one.
The squaring of a permutation P, which is equal to applying the permutation twice, is denoted as P^2. Squaring only affects the cycles with an even number of elements. This can be easily seen when the above permutation is squared, which leads to the following permutation:

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 8 & 2 & 5 & 1 & 9 & 6 & 7 & 4 & 3 & 10 \end{pmatrix}$$

The cycle decomposition of the squared permutation is:

$$(1, 8, 4)(3, 5, 9)(2)(6)(7)(10)$$

Exercise 6. *Ensure that squaring the following permutation does not change the number of cycles, nor the sizes of the cycles .*

$$(1, 8, 4)(3, 5, 9)(2)(6)(7)(10)$$

The expected number of cycles in a random permutation on N elements is the N-th Harmonic number ($\sum_{k=1}^{N}(\frac{1}{k})$) which can be (for large N) approximated by the natural logarithm. The expected number of cycles with an even number of elements is equal to the expected number of odd cycles in a permutation. This leads to the following theorem:

Lemma 1. *Let C_ℓ be a permutation cycle with length $\ell \in \mathbb{N}$ and let $t \in \mathbb{N}$ such that l is divisible by t, then the permutation $(C_\ell)^t$ consists of t cycles of length $\frac{l}{t}$.*

Proof. Let $S = (s_0, s_1, s_2, ..., s_{\ell-1})$ be a sequence of length ℓ. Since ℓ is divisible by t the t subsequences $S_0 = (s_0, s_t, s_{2t}, \dots), S_1 = (s_1, s_{t+1}, s_{2t+1}, \dots), \dots, S_i = (s_i, s_{t+i}, s_{2t+i}, \dots), \dots, S_{t-1} = (s_{t-1}, s_{tk-1}, \dots)$ are disjoint subsequences obtained by skipping t elements. The permutation cycle C_l can be mapped to S by taking s_0 to be an element of the cycle $e \in C_l$, $s_1 = P(e)$, $s_2 = P(P(e))$ etc. □

Theorem 1. *Given a permutation $P \in \mathcal{P}$ with P having N elements. The expected number of cycles in P^t is for prime t: $\frac{2t-1}{t} \cdot \ln(N)$*

Proof. The expected number of cycles with length divisible by t is $\frac{1}{t}\ln(N)$ and by Lemma 1 the cycles with length divisible by t are split into t cycles when raising the permutation to the power t. This results in an expected number of cycles of:

$$t \cdot \frac{1}{t}\ln(N) + \frac{t-1}{t}\ln(N) = \frac{2t-1}{t}\ln(N)$$

□

By Lemma 1 a random permutation P containing a cycle with a length divisible by t to the power t contains t equally sized cycles.

6.1.2 Distinguishing t-encryption

Patarin [38] shows that a Feistel network corresponds to an even permutation, that means that it can be written as a product of an even number of transpositions. Using the full codebook ($O(2^n)$) one can thus distinguish a Feistel cipher from a random permutation. We note that this also implies that the number of cycles in the permutation is even.

However, distinguishing a Feistel based cipher from a even permutation is much harder in the general case. In this section a distinguisher is explored that can distinguish double encryption with the same key from a random Feistel network based permutation in the general case.

Let $C(p, \kappa) = c$ with $\kappa \in \mathbb{F}_2^k$ and $p, c \in \mathbb{F}_2^n$ denote a cipher that given a key κ and plaintext p produces a ciphertext c. Note that this function generates a permutation for each key κ. Let double encryption be denoted by $C^2(p, \kappa) = C(C(p, \kappa), \kappa)$. Then $C^2(p, \kappa)$ has the same effect as squaring the permutation generated by the cipher. By Theorem 1 the expected number of cycles in the cycle decomposition of the permutation generated by C^2 for every key κ is $\frac{3}{2}\ln(N)$ for $N = 2^n$.
The above property can be used to create the following distinguisher: Given that the number of cycles of a random permutation follows a Poisson distribution with a mean equal to $\ln(N)$, where N is the number of elements of the permutation, the distribution of the number of cycles of a squared permutation is Poisson distributed with a mean equal to $\frac{3}{2}\ln(N)$.

To decide when the distinguisher should return "double encryption" and when it should return "random permutation" the following equality should be solved:

$$\begin{aligned}
\frac{e^{-\mu} \cdot \mu^x}{x!} &= \frac{e^{-1.5\mu} \cdot (1.5\mu)^x}{x!} \\
e^{-\mu} \cdot \mu^x &= e^{-1.5\mu} \cdot (1.5\mu)^x \\
\frac{\mu^x}{(1.5\mu)^x} &= \frac{e^{-1.5\mu}}{e^{-\mu}} \\
\ln \frac{\mu^x}{(1.5\mu)^x} &= \ln \frac{e^{-1.5\mu}}{e^{-\mu}} \\
\ln 1.5^{-x} &= -1.5\mu + \mu \\
x &= \frac{-.5\mu}{-\ln 1.5} \\
x &\approx 1.23315 \cdot \mu.
\end{aligned}$$

Thus given a permutation on N elements the distinguisher should return "random permutation" if the number of cycles in the permutation is lower then

$$\ln(N) \cdot 1.23315$$

and "double encryption" otherwise.

To extend this distinguisher to distinguish k-encryptions note that the following inequality holds for $t > 2$:

$$\frac{2k-1}{t} \ln N \geq 1.23315 \ln N$$

This implies that the distinguisher works for k encryptions with $k \geq 2$.

6.1.3 Equal cycle length distinguisher

The aforementioned distinguisher is very straightforward, but there is another property of squared permutations that can be exploited to construct a distinguisher for double encryption.

Recall that given a permutation $P \in \mathcal{P}$ the cycles with even size will be split up in two equally sized cycles in P^2. This can be used to create a distinguisher as the probability of two cycles of equal length m appearing in a random permutation is $(\frac{1}{m})^2$.

Lemma 2. *The probability of a random permutation of N elements containing two cycles with equal length m is $\frac{1}{m^2}$, for $2m \leq N$.*

Proof. To prove this lemma a counting argument is used. First, the number of ways $2m$ elements can be picked from a permutation with N elements is $\binom{N}{2m}$. The elements can be divided into two sets of size m in $\binom{2m}{m}$ ways. The sets can be permuted in $(m-1)!$ ways and the elements not in one of the cycles can be permuted in $(N-2m)!$ ways. This leads to

$$\binom{N}{2m}\binom{2m}{m}((m-1)!)^2(N-2m)! = \frac{N!}{m^2} \quad \text{for } 2m \leq N$$

permutations having at least two cycles with length m. The probability that a randomly chosen permutation has at least two cycles with size m is

$$\frac{N!}{m^2} \cdot \frac{1}{N!} = \frac{1}{m^2} \quad \text{for } 2m \leq N$$

. □

This distinguisher does not need a full code book analysis. Future work could focus on computing the expected number of oracle calls needed for this distinguisher to work (and the probability that the distinguisher is right).

6.1.4 Impossible cycle length distinguisher

To distinguish double encryption the following observation can be used: When squaring the permutation not all cycle lengths are possible. To be precise, with a permutation of size N the square permutation cannot contain cycles of even length larger than $\frac{N}{2}$. Thus when a cycle with even length is observed with a length larger than $\frac{N}{2}$ the permutation is not an even permutation.

Lemma 3. *Given a random permutation p of size N, the probability that in this permutation a cycle larger than $\frac{N}{2}$ is present, is approximately*

$$H_N - H_{\frac{N}{2}} \approx \ln 2 \approx 0.69315$$

where H_m is the m-th Harmonic number.

Proof. First we prove that the number of permutations of length N containg a cycle of length $m < \frac{N}{2}$ is $\frac{N!}{m}$. We can choose $\binom{N}{m}$ sets with m elements from N elements. Fixing the first element to account of rotational symmetry of cycles,

each set can be ordered in (m-1)! ways. The rest of the elements can be ordered in $(N-m)!$ ways. This gives

$$\binom{N}{m}(m-1)!(N-m)! == \frac{N!}{m}$$

permutations with a cycle of length m for $m > \frac{N}{2}$.
Since the sets of permutations with cycles of length $> \frac{N}{2}$ are mutually exclusive, the number of permutations containing a cycle with length $> \frac{N}{2}$ can be expressed as:

$$\sum_{m=\frac{N}{2}}^{N} \frac{N!}{m}.$$

This leads to the probability of picking a permutation with N elements, containing a cycle with length $> \frac{N}{2}$ to be:

$$\frac{1}{N!} \cdot \sum_{m=\frac{N}{2}}^{N} \frac{N!}{m} = \sum_{m=\frac{N}{2}}^{N} \frac{1}{m} = \sum_{m=1}^{N} \frac{1}{m} - \sum_{m=1}^{\frac{N}{2}} \frac{1}{m}$$

This can be expressed as:

$$H_N - H_{\frac{N}{2}} \approx \ln N - \ln \frac{N}{2} = \ln 2 \approx 0.69315$$

□

This has been described in the 100 prisoners problem which was introduced by [20].

This can be generalized such that the probability of a random permutation containing a cycle with length $> k \cdot N$, but since the sets of permutations containing cycles with a length $< \frac{N}{2}$ are not mutually exclusive, care has to be taken not to double count the number of permutations.

There exists only one cycle with length $> N$ and this cycle has with equal probability even or odd length. Using the above result the probability that an even length cycle of length $> n$ in the permutation p exists is:

$$0.69315 \cdot 0.5 = 0.346575$$

By using these probabilities an algorithm can be constructed distinguishing between a random permutation and the square of a random permutation. The attacker is given an oracle O which emulates a random permutation or the square of a random permutation with equal probability. The attacker should output 1 if O emulates a squared random permutation and 0 if the oracle O emulates a random permutation. The attacker is allowed at most q queries to

the oracle.

The probability that the attacker guesses right if the number of queries q is smaller than N (for a permutation of size $2N$) is exactly 0.5. When the number of available queries is larger than N the probability of finding a cycle with length divisible by 2 and larger than N is 0 if the permutation is square. If the permutation is random the probability of finding a cycle with even length and with a length larger than N is:

$$0.5 \cdot 0.346575 = 0.1732875$$

What rests is to design an algorithm such that the attacker gains an advantage using a maximal amount of queries q. This can be done by using the observation as discussed before. The algorithm proposed is as follows: Start at a random element s and set the counter $i = 0$. Then request the oracle for an encryption $c = O(s)$ and increase i by one. If c is equal to s then stop, otherwise compute a new $c = O(c)$ and increase the counter by one. Repeat until a cycle is found or $i > q$.
If $i > q$ and no cycle is found return 0 (random permutation). If a cycle is found with $i > N$ and i divisible by 2 return 0, return 1 otherwise.

6.2 Experimental verification on Simon32

To verify the idea experimentally the number of cycles generated by double encrypting with Simon32 is computed and compared to normal Simon32. As expected double encrypting with Simon32 generates the expected cycle length patterns as can be seen in Figure 9. The next thing to check is the number of cycles with equal length in double encrypted Simon32 and Simon32 which is depicted in Figure 10. These figures were produced using 400 random keys and calculated the number and the length of cycles by doing a full codebook analysis.

6.3 Constant memory cycle length decomposition

Finding a cycle decomposition of a permutation from an n bit cipher generally takes $O(2^n)$ memory and time. In this section a constant memory algorithm is given to compute the cycle lengths of the disjoint cycle decomposition of a permutation. The worst case running time of this algorithm given a maximum amount of memory m is $O(\sum_{i=0}^{2^n} 2^n - i) < O(2^{2n})$ for $m \ll 2^n$. The algorithm is given in Algorithm 3.

The idea behind the algorithm is that every cycle in the permutation has one element that defines the cycle. We choose the defining element to be the smallest element in the cycle. To decompose the permutation into disjoint cycles we pick an element and traverse the permutation starting at the element. If we

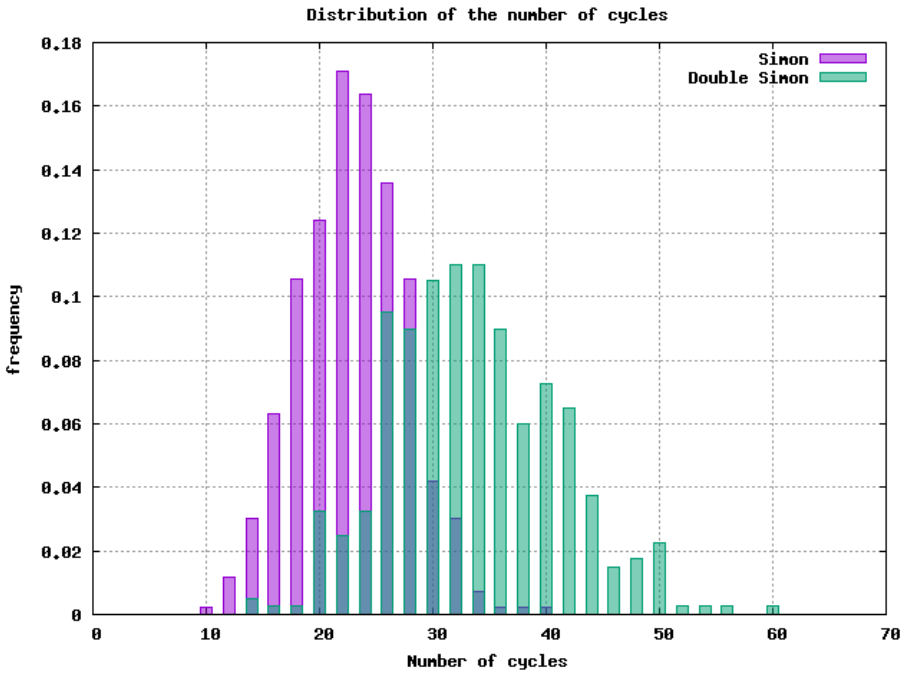

Figure 9: The number of cycles for 400 random keys using Simon32 with 12 rounds.

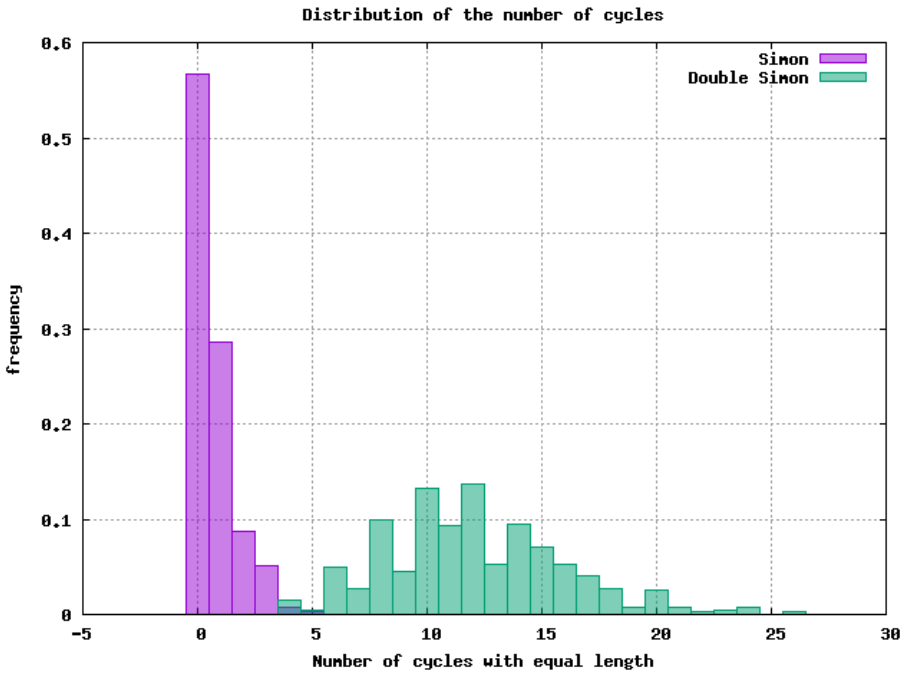

Figure 10: The number of cycles with equal length for 400 random keys using Simon32 with 12 rounds.

find an element smaller than the element we started on, the starting element is not the identifier of the cycle and we abort and pick the next element. If we reach the starting element without encountering a smaller element we are sure that the starting element uniquely defines the cycle we traversed and we increase the cycle counter by one and continue with the next element. Once we tried all elements in the permutation the counter contains the amount of disjoint cycles in the permutation.
The starting elements are picked in lexicographical order.
The above sketches the algorithm when `memsize` is 0. We can save some time by allowing the algorithm to save some of the encountered elements that still have to be tried. We can save some time by skipping those elements when choosing new starting point. If we choose `memsize` to be N, for a permutation on N elements, the algorithm runs in N operations.

Algorithm 3 DecomposeCyclesLowMemory(Permutation σ, Int memsize)

```
Let C be the number of clusters
Let p = 0 be a counter
Let F be an array of booleans of size memsize initialised to 0
Let |σ| denote the size of the permutation
while p < |σ| do
  let start vertex s = p
  while true do
    s = σ(s)
    if s < p then
      break
    else if s == p then
      C = C + 1
      break
    end if
    if s < p + memsize then
      F[s − p] = 1
    end if
  end while
  if F contains a 0 then
    Let next be the index of the first 0 in F
    p = p + next
    Shift F to the left by next bits.
  else
    Set all cells in F to 0
    p = p + memsize
  end if
end while
return C
```

6.4 Conclusion

This section describes a distinguisher to distinguish double encryption with the same key. It does also work on ciphers that can be written as a power of a permutation.

7 Rewriting Simon into ATC

This section describes a method to rewrite or restructure Simon (or any similar cipher). The transformation was used to better understand the cryptanalysis tools encountered during my studies regarding Simon. The transformation transforms Simon into a structure somewhat similar to DES.

Note that by restructuring a cipher the cryptographic properties do not change. The same attacks work and the security of the transformed cipher is equal to the original cipher. Nevertheless, it can be an interesting way to look at a cipher and it certainly is possible to find new interesting properties in the transformed cipher.

One advantage of restructuring a Simon-like cipher to contain S-boxes is that constructing a DDT for a reasonably sized S-box is easy. This allows for faster and easier differential cryptanalysis of Simon-like ciphers by significantly reducing the time and memory needed to create and store the DDT.

The basic idea is to transform a cipher C, which in this case is a Feistel cipher, to another equivalent function C' that has some kind of predefined structure. One trivial solution would be to treat the whole round function as one big substitution box and substitute the round function by this S-box.

Assume that $C : \mathbb{F}_2^n \times \mathbb{F}_2^k \to \mathbb{F}_2^n$ then for C' to be equivalent to C we need $C' : \mathbb{F}_2^n \times \mathbb{F}_2^k \to \mathbb{F}_2^n$ such that for all $\kappa \in \mathbb{F}_2^k$ and $p \in \mathbb{F}_2^n$ we have $C(p, \kappa) = C'(p, \kappa)$. Let $f : \mathbb{F}_2^n \times \mathbb{F}_2^m \to \mathbb{F}_2^n$ denote the round function, then a cipher consisting of r rounds can be described by

$$C(p, k) = f(f(...f(p, k_0), k_1), ..., k_r)$$

where $k_0, k_1, ..., k_r$ are determined by the key κ and a key schedule $K : \mathbb{F}_2^k \to (\mathbb{F}_2^m)^r$. Note that in this section only the case is considered where the round functions f and f' of C and C' are equivalent although this is not needed for C and C' to be equivalent.

There are infinitely many equivalent functions, but as noted earlier the goal of the transformation was to transform Simon into a more DES-like structure. As can be seen in Figure 11 the new structure consists of Expansion, S-box and Permutation layers. In the next paragraphs three possible and equally valid

constructions will be argued.

The first construction that springs to mind is the construction where the S-box layer consists of one $n \times n$ S-box and the Expansion and Permutation layers are both the Identity. The main disadvantage of this construction is that all structure is removed from the round function. This is a valid tactic to use for differential cryptanalysis. A better and more interesting tactic, especially for differential cryptanalysis would be a top down technique as described by Dinur, Dunkelman, Gutman, and Shamir [18].

A second possibility is to look at the structure of Simon and create an S-box for each output bit. For Simon32 this would result in 16 small, easy to analyse S-boxes. Nevertheless, the analysis of the transformed cipher would not be different from the original cipher. The expansion function of this variant will become quite large, while the permutation layer will stay equal to the Identity.

The third possibility and the one chosen to elaborate on in this section is to look at the input bits influencing each output bit. The output bits are grouped into sets such that the total number of input bits needed to compute each set of output bits is minimal. As can be seen in Table 5 each output bit is dependent on 3 input bits. When going for four S-boxes the best possible partitioning of output bits results in 8×4 S-boxes (8 input bits to 4 output bits). One such partitioning is defined in the next sections. Note in this case both the expansion and permutation layers are non trivial.

These 8×4 S-boxes allow for a more efficient differential cryptanalysis by reducing the time needed to create a DDT for Simon. The DDT of the S-boxes needed for Simon32 can be constructed in $2^{2 \cdot 8} \cdot 4 = 2^{18}$ time. Since every version of Simon can be expressed in the same fashion, and only the number of S-boxes grows, the DDT of a Simon version with n-bit words can be computed in $2^{2 \cdot 8} \cdot \frac{n}{4}$ time and memory.

7.1 Expansion layer

The expansion expands the 16 bit blocks to 32 bits to feed the four 8-bit S-boxes. The bit permutation of the expansion layer is given in Figure 12.

7.2 S-box layer

The ouput of the expansion matrix is divided into four blocks of 8 bits which are then fed into the S-boxes. $([1 \dots 8] \rightarrow S_1, [9 \dots 16] \rightarrow S_2 [17 \dots 24] \rightarrow S_3, [25 \dots 32] \rightarrow S_4)$
The S-box layer consists of four different 8 bit S-boxes with 4 bit outputs which are given by Tables 10, 11, 12, 13. The columns of the tables denote the least

I_1	I_2	I_3	Output bit
16	8	15	1
1	9	16	2
2	10	1	3
3	11	2	4
4	12	3	5
5	13	4	6
6	14	5	7
7	15	6	8
8	16	7	9
9	1	8	10
10	2	9	11
11	3	10	12
12	4	11	13
13	5	12	14
14	6	13	15
15	7	14	16

Table 5: Dependency of output bits on input bits in Simon32

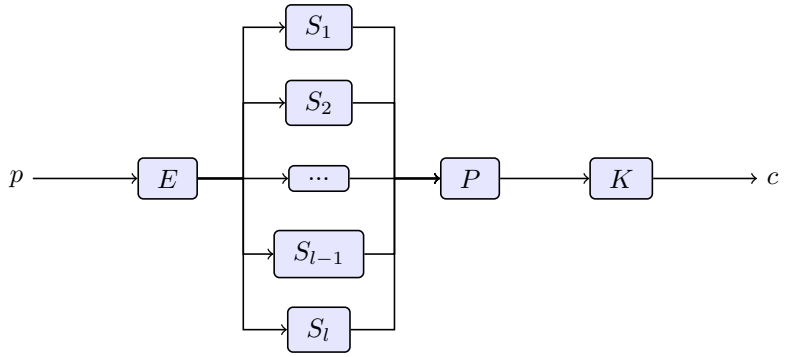

Figure 11: ATC round function

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \end{pmatrix} \mapsto$$
$$\begin{pmatrix} 1 & 2 & 8 & 9 & 10 & 11 & 15 & 16 & 2 & 3 & 4 & 5 & 6 & 12 & 13 & 15 & 3 & 4 & 5 & 9 & 10 & 11 & 12 & 14 & 1 & 6 & 7 & 8 & 13 & 14 & 15 & 16 \end{pmatrix}$$

Figure 12: Expansion layer bit expansion for E

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \end{pmatrix} \mapsto$$
$$\begin{pmatrix} 1 & 2 & 3 & 10 & 4 & 5 & 7 & 14 & 6 & 11 & 12 & 13 & 8 & 9 & 16 & 15 \end{pmatrix}$$

Figure 13: Permutation layer bit permutation for P

significant bits and the rows denote the most significant bits of the input. The elements of the table denote the output of the S-box.

7.3 Permutation layer

The permutation layer takes the 16 bit output of the S-box layer and rearranges the bits according to the bit permutation as given in Figure 13

8 Related key differential attack on Simon

As discussed in Section 4 the relations between keys can be used to increase the attack area of a cipher. This section takes advantage of a cycle found in the key schedule to construct a novel related key attack. This attack is different from other related key attacks in the sense that the Hamming weight of the transitions is high ($\frac{n}{2}$ with n the word size of the cipher) in comparison to other differential attacks.

Another difference to other differential attacks is that the transitions used in this attack have an underlying structure that can be used to enhance the attack.

8.1 Key schedule Cycles

The Simon key schedule can be seen as function that maps an internal key schedule state to the next internal key schedule state ($\mathbb{F}_2^{64} \to \mathbb{F}_2^{64}$). By looking at the `xor` of two keys this function becomes an invertible linear function and can be described by an invertible matrix M. Given an internal state $K_j = (k_j, k_{j+1}, k_{j+2}, k_{j+3})$, the matrix M can be used to compute the $j+i$-th internal state K_{j+i} by computing: $K_j \cdot M^i$.
The set of linearly independent eigenvectors of the matrix M^i denote the unit vectors of internal key states that result in a cycle of length i.

Two interesting internal states which create cycles are: `aaaa aaaa aaaa aaaa` and `5555 5555 5555 5555` which both will generate a cycle of length 1 (e.g. given that the `xor` of two master keys is `5555 5555 5555 5555` the `xor` of every round key is `5555`)

Apart from the method described above the existence of cycles with length 1 can also be proven as is shown in Lemma 4

First an expression for the difference of two round keys must be established. Let S^{-i} denote the right bitwise rotation by i bits. Then round key k_{i+4} is expressed by:

$$k_{i+4} = c \oplus (z_j)_i \oplus k_i \oplus (S^{-4} \cdot k_{i+3}) \oplus (S^{-3} \cdot k_{i+3}) \oplus (S^{-1} \cdot k_{i+1}) \oplus k_{i+1},$$

then the difference $k_{i+4} \oplus k'_{i+4} = \Delta k_{i+4}$ can be expressed as:

$$\Delta k_{i+4} = \Delta k_i \oplus (S^{-4} \cdot \Delta k_{i+3}) \oplus (S^{-3} \cdot \Delta k_{i+3}) \oplus (S^{-1} \cdot \Delta k_{i+1}) \oplus \Delta k_{i+1},$$

Lemma 4. *Given that*

$$\Delta k_{i+4} = \Delta k_i \oplus S^{-4} \cdot \Delta k_{i+3} \oplus S^{-3} \cdot \Delta k_{i+3} \oplus S^{-1} \cdot \Delta k_{i+1} \oplus \Delta k_{i+1}$$

and let

$$\Delta k_i = \Delta k_{i+1} = \Delta k_{i+2} = \Delta k_{i+3} \tag{1}$$

1-cycles exist for the key schedule.

Note that for 1-cycles to be present in the key schedule assumption (1) is essential and we need to prove that equation (2) is true.

$$\Delta k_{i+4} = \Delta k_i = \Delta k_{i+1} = \Delta k_{i+2} = \Delta k_{i+3} \tag{2}$$

Proof.

$$\begin{aligned}
\Delta k_{i+4} &= \Delta k_i \oplus S^{-4} \cdot \Delta k_{i+3} \oplus S^{-3} \cdot \Delta k_{i+3} \oplus S^{-1} \cdot \Delta k_{i+1} \oplus \Delta k_{i+1} && (3)\\
&= \Delta k_i \oplus S^{-4} \cdot \Delta k_{i+3} \oplus S^{-3} \cdot \Delta k_i \oplus S^{-1} \cdot \Delta k_i \oplus \Delta k_i && (4)\\
&= S^{-4} \cdot \Delta k_i \oplus S^{-3} \cdot \Delta k_i \oplus S^{-1} \cdot \Delta k_i && (5)
\end{aligned}$$

□

If $\Delta k_{i+4} = \Delta k_i$ then

$$\Delta k_i \oplus S^{-1}\Delta k_i = S^{-3}\Delta k_i \oplus S^{-4}\Delta k_i = S^{-3}(\Delta k_i \oplus S^{-1}\Delta k_i)$$

and

$$\Delta k_i \oplus S^{-4}\Delta k_i = S^{-1}(\Delta k_i \oplus S^{-1}\Delta k_i).$$

From these expressions we get that $k_i = S^{-1}\Delta k_i$ or $k_i = S^{-2}\Delta k_i$.

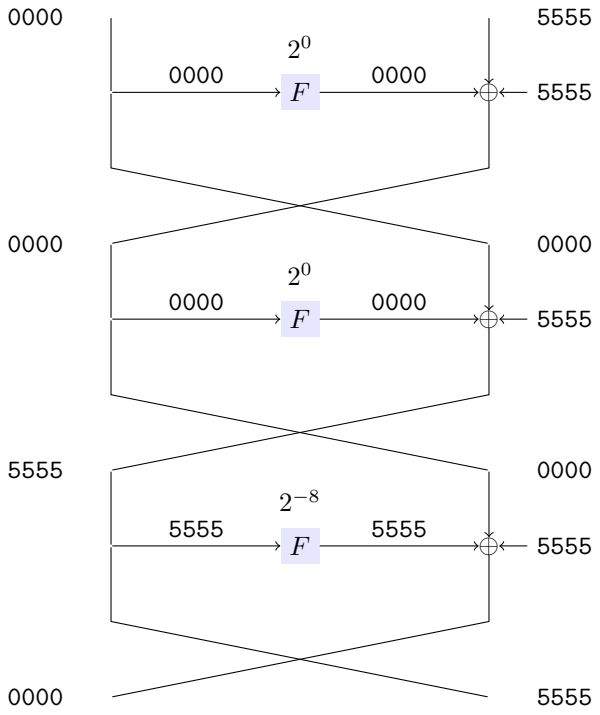

Figure 14: The related key differential characteristic used to attack Simon

8.2 Related Key attack Simon32/64

Some of the most powerful differential attacks rely on iterated characteristics [4]. These are differential characteristics where the input and output difference are equal. To find these characteristics in the related key setting the key schedule has to have a cycle with the same phase as the size of the iterated characteristic.

As described in Section 8.1 the key schedule of Simon32/64 has a few key differences that result in cycles in the round key differences. These keys can be used to look for iterated differential characteristics in the related key setting. The characteristic described in this section works for two different key differences, namely: `aaaa aaaa aaaa aaaa` and `5555 5555 5555 5555`.

The 3-round iterated characteristic for key difference `aaaa aaaa aaaa aaaa` is: `0000 aaaa` $\xrightarrow{2^0}$ `0000 0000` $\xrightarrow{2^0}$ `aaaa 0000` $\xrightarrow{2^{-8}}$ `0000 aaaa` and for the key difference `5555 5555 5555 5555`: `0000 5555` $\xrightarrow{2^0}$ `0000 0000` $\xrightarrow{2^0}$ `5555 0000` $\xrightarrow{2^{-8}}$ `0000 5555`. See Figure 14 for a visual representation of this characteristic.

Note that since the two key differences (i.e. `aaaa` and `5555` will result in a related key differential distinguisher with the same probability it means that

two key pairs can be used to increase the code book by a factor of two. To increase the codebook by a factor of 2^2 keys K_0, K_1, K_2, K_3 should be used with the following relationships:

$$K_0 \oplus K_1 = \texttt{5555 5555 5555 5555}$$
$$K_0 \oplus K_2 = \texttt{aaaa aaaa aaaa aaaa}$$
$$K_1 \oplus K_3 = \texttt{aaaa aaaa aaaa aaaa}$$
$$K_2 \oplus K_3 = \texttt{5555 5555 5555 5555}$$

Breaking one of the keys, breaks all the keys.

The probability of this characteristic occurring can be computed as follows. Assume that the round function and key schedule produce random output. The proposed characteristic has two distinct cases that influence the probability of the differential characteristic occurring. In the first case the left hand side is `0000` and it is easy to see that the right hand side difference after the round function will be `5555` with probability 1. In the second case where the left hand side difference is `5555` should result in a right hand side difference of `0000` after the round key difference addition. This means that the right hand side difference before the key addition should be equal to the left hand side difference, namely `5555`. The probability of this event occurring is calculated in the next paragraph.

Note that the round function can be split up in two parts. A linear and non-linear part, where the linear part is $L(p) = p \lll 2$ and the non-linear part is $N(p) = p \lll 1$ `and` $p \lll 8$. The linear part L will always preserve the difference, thus we have to make sure that the difference of the non-linear part $N(p_1) \oplus N(p_2) = \texttt{0000}$, otherwise the output difference after both parts would be different. This leads to the following equation with inputs $p_1, p_2 \in \mathbb{F}_2^{16}$ required to be true:

$$p_1 \lll 1 \texttt{ and } p_1 \lll 8 = p_2 \lll 1 \texttt{ and } p_2 \lll 8$$

in addition to $p_1 \oplus p_2 = \texttt{5555}$. Using this, the above equation is only true if every 0 bit in the difference is the result of two 0 bits in the plaintexts, leading to p_1 and p_2 being elements of the Moser-de Bruijn sequence [16] if the above equation holds.

Definition 4 (Moser-de Bruijn sequence)**.** A natural number $n \in \mathbb{N}$ is an element of the Moser-de Bruijn sequence if every $2 \cdot i + 1$-st bit is 0 for $i \in \mathbb{N}$, i.e., every bit with an odd index bit is 0.

The number of elements in the Moser-de Bruijn sequence up to n is $\sqrt{n}$ [16] thus the probability of $x \in \mathbb{F}_2^{16}$ to be an element of the Moser-de Bruijn sequence is 2^{-8}. In general, the probability that a random element from $\mathbb{F}_2^n$ is

an element of the Moser-de Bruijn sequence is $2^{-\frac{n}{2}}$.

Notice that if word p_1 is an element of the Moser-de Bruijn sequence and $p_1 \oplus p_2 = \texttt{5555}$, then p_2 is an element of the Moser-de Bruijn sequence, thus the probability that both p_1, p_2 are elements of the Moser-de Bruijn sequence is 2^{-8}.

The above leads to a probability of 2^{-8} that the three round characteristic succeeds.

The interesting part here is that if all inputs to the round function of Simon are elements of the Moser-de Bruijn sequence, the outputs of the round function are as well.
Note that if it could be proven or disproven that there exists a master key for which the key schedule generates an element of the Moser de Bruijn sequence for each round key, then this would be a weak key. Encryption with such a key could easily be distinguished from a random function by giving as input an element of the Moser-de Bruijn sequence, which would result in an element of the Moser-de Bruijn sequence as an output. Note that no such master key exists for Simon32 and probably no such master key exists for the other versions.

An experiment has been run to see if such a key exists for Simon32. Since there are $2^{\frac{64}{2}} = 2^{32}$ Moser-de Bruijn keys this experiment was feasible. No keys have been found to generate a Moser-de Bruijn round key for every round. Nevertheless an interesting observation is that a sizeable fraction of these keys always generates a Moser de-Bruijn element as a round key on rounds: $1, 2, 3, 4, 9, 21, 27$
Although at this moment this has not led to an attack it is something to keep in mind.

The same reasoning can be applied for the `aaaa aaaa aaaa aaaa` case.
Using the above it is interesting to notice that the differential path can only succeed if both plaintexts in the pair succeed to keep generating an Moser-de Bruijn element every third round (i.e. every time the differential transition $5555 \rightarrow 0000$ occurs the left hand input of the round function must be an element of the Moser-de Bruijn sequence). The probability of this happening if we consider the key schedule to be a random function (as often is assumed) is 2^{-8}. Experiments show that this is not the case for every key pair especially if we take care with choosing the key pairs. The average number of correct pairs found seems to stay true to the expected probability, but some of the key pairs tend to generate far more correct pairs thanothers (see Table 6 for a histogram of the number of correct pairs with Moser-de Bruijn key and plaintext.

Another interesting idea is that since the characteristic can only occur when the left hand part of the characteristic is an element of the Moser-de Bruijn sequence we can cut down the number of input pairs to be tried to 2^{24}.
By choosing the keys to be Moser-de Bruijn elements we can reduce the number of input pairs by even more because then all the inputs for the first four rounds

# keys	# correct pairs
4862	0
6	1
16	2
4	3
12	4
5	5
11	6
10	8
1	9
2	10
2	11
12	12
2	14
1	15
8	16
1	17
4	18
1	21
2	22
10	24
1	28
3	32
1	33
2	36
2	42
7	48
1	56
1	60
1	62
1	72
2	96
1	120
1	144
1	156
1	182
1	240
1	512

Table 6: The number of keys generating a certain number of correct pairs. All experiments were run with a random Moser-de Bruijn key and Moser-de Bruijn inputs on 12 round Simon32

will be elements from the Moser-de Bruijn sequence. This would mean that we essentially skipped the first 4 rounds in the differential path. Nevertheless at this moment in time I am still trying to witness this behaviour in my experiments. In essence this would lead to an extension of the differential by one round which in the current experiments is not witnessed.

8.2.1 Other versions

The characteristic is applicable to every Simon version using the key schedule with four keywords (i.e. $m = 4$). The probability for the 3-round characteristic is directly defined by the word size (i.e. $2^{-\frac{n}{2}}$ where n is the word size. For versions with word size 16, 24, 32, 64 the probability of the 3-round iterated characteristic occurring is respectively $2^{-8}, 2^{-12}, 2^{-16}, 2^{-32}$. This results in a maximum of 12 rounds for every version using $m = 4$.

Lemma 5. *When $m = 2$, the key schedule does not exhibit key difference cycles with master key differences* `5555 5555` *and* `aaaa aaaa`*.*

Proof. The key schedule with $m = 2$ is

$$k_{i+2} = k_i \oplus (S^{-3} \cdot k_{i+1}) \oplus (S^{-4} \cdot k_{i+1}).$$

To have a 1 round cycle the following needs to hold:

$$k_{i+2} = k_{i+1} = k_i$$

which gives us the following equation:

$$k_i = k_i \oplus S^{-3} \cdot k_i \oplus S^{-4} \cdot k_i$$

which holds if

$$S^{-3} \cdot k_i = S^{-4} \cdot k_i$$

Which holds if

$$k_i = S^{-1} k_i$$

For this equation to hold all bits need to be equal, thus only allowing for `0000 0000` and `ffff ffff` to be master keys giving a one round cycle in the key schedule.

□

8.2.2 Extending the differential

The differential can be extended for free by two rounds by using the differential characteristic `0000 aaaa` $\xrightarrow{2^0}$ `0000 0000` $\xrightarrow{2^0}$ `aaaa 0000` $\xrightarrow{2^{-8}}$ `0000 aaaa` $\xrightarrow{\cdots}$ `0000 aaaa` $\xrightarrow{2^0}$ `0000 0000` $\xrightarrow{2^0}$ `aaaa 0000`

metric	9 rounds	12 rounds	15 rounds
# samples	800	800	800
# fails	0	409	799
average # correct pairs	254.9	0.961	0.001
standard deviation	130.0	1.384	0.035

Table 7: Experimental results for Simon32 with 9, 12 and 15 rounds

8.3 Experimental results

An experiment was conducted by trying random key k and $k' = k \oplus$`5555 5555 5555 5555` and recording how many plaintext pairs followed the differential described in Figure 14. A summary of the results is given in Table 7. A fail is an instance where no right pairs have been found.

Another experiment was done by using only plaintext pairs consisting of Moser-de Bruijn elements. This resulted in far fewer plaintext pairs to be tried per key (2^{16} instead of 2^{32} pairs). This allowed for far more key pairs to be tried. Finding a correct pair can be modelled as a Poisson process and in this particular instance it is a Poisson process with $\lambda = 1$. The results of the experiment can be found in Table 8. Note that this does not fit a Poisson process (the probability of encountering an instance with 48 correct pairs would approximately be 2^{-205}, but was observed).

8.4 Observations

During my research I have done a lot of experiments of which not all have resulted in an attack. Nevertheless, some I have made some observations that might be interesting for others to read. This section contains a couple of interesting observations done for Simon32.

The first observation is that some key pairs generate a larger than expected number of right pairs. The first thought was that this could be caused by the key schedule and thus by looking at the keys that invoke high number of occurrences some sort of system could be found. This, however, is (at least at first sight) not the case. The keys which generate more then 18 correct pairs are summarised in Table 9.

Another observation is that the number of correct pairs, which is the number of input pairs resulting in the correct output and the number of right differentials, which are the pairs for which the differential is right, are the same. This is observed in all experiments that have a Moser-de Bruijn input.

The following observation does not solely focus on differential cryptanalysis, but it is interesting to note nevertheless. When the left hand part of the cipher is a Moser-de Bruijn element the non-linear part of the cipher is skipped and

occurrences	**Right pairs**
47232	0
28070	1
13486	2
5836	3
2686	4
1229	5
680	6
298	7
155	8
95	9
69	10
52	11
39	12
20	13
12	14
11	15
11	16
5	18
1	19
3	20
1	21
1	22
1	25
1	26
1	28
1	29
1	32
1	42
1	46
1	48

Table 8: Experiment run on 12-round Simon32.

Key 1	Key 2	Right pairs	Right differentials
051098C151775211	5045CD9404220744	18	18
1AAB75B53FB9D557	4FFE20E06AEC8002	18	18
20025C3E73E71DE0	7557096B26B248B5	18	18
3001725E5F4AEF0E	6554270B0A1FBA5B	18	18
4F1D54C14A677875	1A4801941F322D20	18	18
1089165466B96D12	45DC430133EC3847	19	19
400146753F2F7964	155413206A7A2C31	20	20
6144CB711C97505F	34119E2449C2050A	20	20
6503078D76DADAED	305652D8238F8FB8	20	20
1A9574ED436723C0	4FC021B816327695	21	21
5D96144C4B8EEA5D	08C341191EDBBF08	22	22
14C724074358FFA2	41927152160DAAF7	25	25
44929D4A050D1902	11C7C81F50584C57	26	26
7478C2CF04B3C780	212D979A51E692D5	28	28
3780600126D03FBC	62D5355473856AE9	29	29
0EC3302D35F5A132	5B96657860A0F467	32	32
2532322461C3888E	706767713496DDDB	42	42
7D57603A685210D5	2802356F3D074580	46	46
04257D107369805F	51702845263CD50A	48	48

Table 9: Interesting keys

the round is linear. This also works on Moser-de Bruijn elements rotated by 1.

8.5 Conclusion

The most interesting part in this section is to see that this related key differential characteristic of Simon can be explained through the structure of the Simon round function. This results in the differential characteristic having concrete preconditions on when it succeeds. Nevertheless, this has not lead to any break throughs. The differential found in this round spans 14 rounds which is the same as in [1]. In contrast to the differential described by Abed et al. [1] the differential described in this section only needs 2^{16} chosen plaintexts to work.

9 Discussion

The title of this thesis is perhaps a little misleading. By reading the title one may be tempted to think that this thesis does only cover the cryptanalysis of Simon. Nevertheless, upon reading the thesis, Simon does not appear frequently. One has to keep in mind that most, if not all, parts have had Simon as a major influencer and motivator. Most of the time these sections started as an alternative method to analyse Simon. They evolved throughout the research and got

altered such that the initial goal (finding weaknesses in Simon) was often in a sense neglected.

To give an example, Section 6 started as a project to analyse the graphs generated by the round function of Simon32. This lead to a fairly weak analysis which was able to replace some encryptions in a brute force search by 1 round encryptions. Later this idea evolved into a description of the cipher describing it as a permutation. Not much later, after some experimentation, a distinguisher with respect to a random permutation, for Simon32, was found experimentally. This however, proved to be already known [38]. This whole track lead to the section on distinguishing t-Encryption. Although the section on k-Encryption does use Simon32 to verify the theory experimentally, originally it was an attempt to break Simon

All sections have had multiple transformations, dead ends and side tracks. This example does show the way I have been working the last months. It maybe is not the most fruitful way of researching and may seem at some times *ad hoc* or unguided. But it resembles the way my mind works and tries to tie everything I learn, hear and read together. This can sometimes lead to some interesting ideas.

In the past weeks/months I have been thinking a great deal about research and especially cryptanalysis in respect to how I have been working. My main concern was that I might have been working in the wrong state of mind. Too much trying to find something novel, something innovative, something that would break Simon. A state in which I have tickled and observed Simon, seen it twitch, wrinkle and roll over in despair never to fall apart. This thought made me think. Not only about the fruitlessness of my attempts, more so about the attempts (I) made on breaking Simon in the traditional way.

This traditional cryptanalysis which started with Biham and Shamir [4] giving us, the public, differential cryptanalysis. Not long after Matsui [29] invented linear cryptanalysis. A newer, but less successful analysis is Algebraic cryptanalysis by Courtois and Pieprzyk [12].

These attacks are the backbone of contemporary cryptanalysis. Most, if not all, interesting new ciphers, are designed such that these attacks have only had success against round-reduced versions. I think, that it is fair to state that new ciphers that are properly designed to withstand these types of attack cannot be broken by them. Notice that there are formal methods of proving a certain cipher to be resistant against these types of attack [26], but it is infeasible (for most ciphers) to prove resistance against all attacks possible by a computationally bounded adversary.

This leads to my belief, that to attack contemporary ciphers, which have all defensive measures in place against the well-known attacks, we need new cryptanalysis tools. Inventing these new cryptanalysis tools takes up a lot of

time and energy. Both have to be invested in an uncertain goal, invested in something more bound to fail than lead to success.

In this light I deemed my chaotic style of learning as a trait and stopped suppressing it. Especially since a thesis is a moment where one can `(1)` fall, get up, `goto 1` until one is bruised and battered and full of knowledge. All of this without major adverse effects.

9.1 Academic significance

One question often asked to you as a researcher in the midst of the corporate world is: 'What is the significance of your research to the public?'. As strange as it seems, academics (in my vicinity) never ask you the question: 'What is the significance of your research to your field of study?'. At least not straight away.

The academic significance of my work is not easy to quantify. There are a lot of bits of which the significance varies greatly. Not all my endeavours are as polished and well developed as it might have been, have I had focussed more on one topic. Nevertheless, they could be seen as a 'proof of concept', a start, as a way to pave the way for further research. It could be seen as a wandering through an exciting forest full of fruits of which only some contain the sweet scent as well as the sweet taste of academic significance.

In the next section I will go over the concepts in this thesis and try to argue their academic significance.

9.1.1 Distinguishing k-encryption

In this section I describe a distinguisher that can distinguish in a generic way a k-encryption with the same key from a random even permutation. This advocates against the use of multiple encryptions with the same encryption key. As well as against encryption schemes having symmetry in their key schedule. Both of these constructions are not seen much nowadays. This distinguisher would have had a larger impact if the attack would be targeting a specific cipher or protocol. I have, to no avail, been looking for symmetry in the Simon key schedule. I have also spent some energy and time on identifying ciphers using a weak key schedule and structure, but no ciphers have been found which are vulnerable. Note that RoadRunneR [2] uses a key schedule that would be exploitable. Nevertheless due to the use of key whitening the attack does not work.

Although the distinguisher does not target any known cipher I know of, it is an interesting distinguisher to keep in mind and one to be used to argue against weak key schedules such as in RoadRunneR [2] or DES. In C a paper is included which describes an attack on RoadRunneR making use of the weak key schedule

to mount a related key differential attack. Note that slide attacks [7] do take advantage of the same weakness of ciphers, but they do require more properties to succeed. Nevertheless when slide attacks do succeed they are more powerful.

Another use case of this technique is to find hidden structures in for example S-boxes as described by [6]. This technique can discover if and how many times a certain structure is repeated without having to know the structure.

After a discussion with Rogaway we came to the conclusion that there are some interesting open questions regarding this topic. The first distinguisher described, the cycle count distinguisher, is a static distinguisher where the attacker needs a full (or near full) codebook number of queries. The second distinguisher, the equal cycle length distinguisher, is adaptive in the sense that after each query the next query is decided upon. In the studies I conducted I did not look at the optimal tactic to choose the next query. This, however, would be very interesting to look at as then a function could be composed expressing the advantage of an attacker given the permutation size and a maximal number of queries q. The mathematical structures to handle these types of statistical problem are described by Flajolet and Sedgewick [19], but are at the moment of writing out of reach.

9.1.2 ATC

In this section a transformation of Simon to an equivalent function is described. The transformation can be used to describe bit-based ciphers as word-based ciphers. This greatly reduces the amount of memory and time needed to compute the DDT.

The transformation does, however, also spring some interesting ideas to mind. One idea is that it should be possible to design a structure such that every block cipher can be transformed into that structure. This structure could then be used for the cryptanalysis of the cipher. Rewriting ciphers into standard structures could unlock the possibility of automatic cryptanalysis, where the cryptanalyst's job is to provide the proper transformation. It could be that this structure is as simple as treating the whole round function as one S-box. At the moment I do not know whether such is possible, but I think it would be worthwhile to do a little bit more research in this direction.

Another interesting question raised is whether or not all structures used in cryptography nowadays have the same expressivity, i.e. are all structures exhibited in the design of ciphers in the same equivalence class? In other words, and to be a bit more concrete, can for example, all Substitution Permutation Networks [22] be expressed as traditional Feistel networks [28]? What about generalised Feistel networks [37]? Or a structure as used in Minalpher [41]?

The last thing this transformation could be used for is hiding structure. In the case of Simon it proved to be 'easy' to recover the structure of the original cipher as Neves showed in a discussion. Nevertheless it is not unthinkable to use such a scheme to hide a certain structure in a round function or even a whole cipher. This is much like hiding structures in S-boxes as in [6], but then on a different scale. Note that DES has a similar structure to the structure Simon has been transformed to. It would be interesting to see whether there is some hidden structure in DES.

9.1.3 Related Key differential Attack

In the section describing the related key differential attack on Simon I describe an attack using a related key differential characteristic of 14 rounds. This characteristic needs 2^{16} chosen plaintexts for a distinguisher. This characteristic has a high Hamming weight which is often advantageous for key recovery attacks.
The only other related key differential attack on Simon I encountered in my studies is an attack by Abed et al. [1], which uses a 14 round related key characteristic to achieve an 18 round key recovery attack, using $2^{54.55}$ time and $2^{30.86}$ chosen plaintexts.
The main concern with related key differentials used against lightweight block ciphers is that on the one hand they are infeasible. This is due to the tendency of devices deploying a lightweight cipher to have a fixed key and the probability of two devices having a specific key difference is small. This means that to use this attack an attacker needs to find two devices with the appropriate key difference, or mount a fault attack on one device to flip some key bits.
On the other hand these devices have a higher tendency to use the Merkle-Damgard construction [15, 31]. Which makes the related key setting more significant.

9.2 Public significance

As stated before a question often asked to me is: 'What is the significance of your research to the public?' Most of the time this question is answered with the following statement: 'Let's assume I do find an attack against Simon or AES or any contemporary cipher. And let's assume that this attack reduces the time needed for key recovery from 2^{128} to 2^{126}. Then this attack, having major academic significance, would reduce the time to attack said cipher from an eternity to one fourth of an eternity.' Although true, this does not tell the whole story. As is often remarked by academics, once a crack has formed in a cipher, the eventual practical break of the cipher is reached sooner.

Another argument often heard is that cryptanalysis nowadays often focuses on round reduced or otherwise crippled ciphers (or hashes) as a target. This, in reality, means that even if one finds a 30 round differential key recovery attack for Simon32 (which has 32 rounds), this attack would not make breaking the

full cipher faster than brute forcing the key.

In this light it is better to see cryptanalysis as a way to measure the cryptographic trust of the ciphers. In this case cryptographic trust is a subjective measure that can be attributed to a certain cipher and depending on who you ask the cryptographic trust in a cipher can greatly vary between parties. Establishing a measuring tool and plotting the distribution of cryptographic trust throughout the population, would be an interesting research topic combining many different fields of studies.

The question of public significance of the research could be answered by stating that my cryptographic trust in Simon has increased over the course of my thesis.

9.3 Simon and the design of lightweight ciphers

One major asset of the academic reviewing process is that it is a double blind process. This is one of the reasons I have high regards for the academic way of working. Nevertheless, after publication, the papers are attributed to persons which removes the anonymity of the authors (in most cases the reviewers stay anonymous after publication, which in essence is quite remarkable). Most of the time this protocol does not inhibit any drawbacks.

In the case of Simon I do not feel the same way. Simon is met with a lot of suspicion and is treated with much more caution because of the authors listed at the top of the paper. The structures and simplicity used in Simon should be receiving a lot of reviewing, but the reviewing should be done without any bias, which is in my opinion part of the scientific legacy.

The goal of lightweight ciphers is often explained by the following: small, efficient and strong, where small implies that the structure should help create a small implementation, either in hardware or in software (or both).
Efficiency touches on several different concepts which are for the most part intertwined. Lightweight ciphers often run on constrained devices, making the efficiency of encryption, i.e., the (amortized) number of bytes encrypted in each clock cycle, an important measure. Another facet of efficiency is often expressed in the energy consumption/latency of the cipher.
A lightweight cipher also needs to be strong, removing the assumption that lightweight ciphers can be less cryptographically secure with respect to 'normal' ciphers.

To this list I would like to add the necessity of a lightweight cipher to be simple. Simple to implement, analyse and audit. This simpleness becomes more important when one keeps in mind that lightweight ciphers are often deployed in speciality processors. This often leads to developers implementing their own

cryptography. And although this practice should not be encouraged and or endorsed, making a cipher overly complicated is not the way to discourage it.

Apart from having ample choices in the cryptographic strength, Simon is in my opinion a good example of a cipher that is simple. The authors show this by including (readable) pseudo code of the cipher in the body of the paper. Another characteristic of a simple cipher is that has to be easily analysable. One argument often used against Simon is that it includes a version with 32-bits blocks and a 64-bits key. Although these versions are not up to standards with regards to security they are often used as a dummy target to test new attacks against Simon as can be seen in [18], [1] and this thesis. This greatly increases the simplicity of analysis of a cipher.

9.4 Conclusion

Although some interesting concepts have been touched upon in this thesis, the main, ultimate and somewhat unreachable goal, of breaking full round Simon has not been met. By researching Simon I got a peek into cryptanalysis, cipher design and standardisation of ciphers. Since and before Simon many interesting lightweight ciphers have been published after, and I look forward to the advances in the lightweight cryptography field with enthusiasm and interest.

References

[1] Farzaneh Abed, Eik List, Stefan Lucks, and Jakob Wenzel. Differential Cryptanalysis of Round-Reduced Simon and Speck. In Carlos Cid and Christian Rechberger, editors, *Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers*, volume 8540 of *Lecture Notes in Computer Science*, pages 525–545. Springer, 2014. ISBN 978-3-662-46705-3. doi: 10.1007/978-3-662-46706-0_27. URL https://doi.org/10.1007/978-3-662-46706-0_27.

[2] Adnan Baysal and Sühap Sahin. RoadRunneR: A Small and Fast Bitslice Block Cipher for Low Cost 8-bit processors. In *Lightweight Cryptography for Security and Privacy - 4th International Workshop, LightSec 2015, Bochum, Germany, September 10-11, 2015, Revised Selected Papers*, pages 58–76, 2015.

[3] Ray Beaulieu, Douglas Shors, Jason Smith, and Stefan Treatman-clark. The Simon and Speck families of lightweight block ciphers. *Cryptology ePrint Archive*, pages 1–42, 2013. URL http://eprint.iacr.org.

[4] Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In Alfred Menezes and Scott A. Vanstone, editors, *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer, 1990. ISBN 3-540-54508-5. doi: 10.1007/3-540-38424-3_1. URL https://doi.org/10.1007/3-540-38424-3_1.

[5] Eli Biham and Adi Shamir. Differential Cryptanalysis of the Full 16-Round DES. In Ernest F. Brickell, editor, *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, volume 740 of *Lecture Notes in Computer Science*, pages 487–496. Springer, 1992. ISBN 3-540-57340-2. doi: 10.1007/3-540-48071-4_34. URL https://doi.org/10.1007/3-540-48071-4_34.

[6] Alex Biryukov and Léo Perrin. On Reverse-Engineering S-Boxes with Hidden Design Criteria or Structure. In Gennaro and Robshaw [21], pages 116–140. ISBN 978-3-662-47988-9. doi: 10.1007/978-3-662-47989-6_6. URL https://doi.org/10.1007/978-3-662-47989-6_6.

[7] Alex Biryukov and David A. Wagner. Slide Attacks. In Lars R. Knudsen, editor, *Fast Software Encryption, 6th International Workshop, FSE '99, Rome, Italy, March 24-26, 1999, Proceedings*, volume 1636 of *Lecture Notes in Computer Science*, pages 245–259. Springer, 1999. ISBN 3-540-66226-X. doi: 10.1007/3-540-48519-8_18. URL https://doi.org/10.1007/3-540-48519-8_18.

[8] John Black and Phillip Rogaway. Ciphers with Arbitrary Finite Domains. In Bart Preneel, editor, *Topics in Cryptology - CT-RSA 2002, The Cryptographer's Track at the RSA Conference, 2002, San Jose, CA, USA, February 18-22, 2002, Proceedings*, volume 2271 of *Lecture Notes in Computer Science*, pages 114–130. Springer, 2002. ISBN 3-540-43224-8. doi: 10.1007/3-540-45760-7_9. URL https://doi.org/10.1007/3-540-45760-7_9.

[9] 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, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007. ISBN 978-3-540-74734-5. doi: 10.1007/978-3-540-74735-2_31. URL https://doi.org/10.1007/978-3-540-74735-2_31.

[10] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçin. PRINCE - A Low-Latency Block Cipher for Pervasive Computing Applications - Extended Abstract. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 208–225. Springer, 2012. ISBN 978-3-642-34960-7. doi: 10.1007/978-3-642-34961-4_14. URL https://doi.org/10.1007/978-3-642-34961-4_14.

[11] Gilles Brassard, editor. *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, 1990. Springer. ISBN 3-540-97317-6. doi: 10.1007/0-387-34805-0. URL https://doi.org/10.1007/0-387-34805-0.

[12] Nicolas Courtois and Josef Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In Yuliang Zheng, editor, *Advances in Cryptology - ASIACRYPT 2002, 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002, Proceedings*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287. Springer, 2002. ISBN 3-540-00171-9. doi: 10.1007/3-540-36178-2_17. URL https://doi.org/10.1007/3-540-36178-2_17.

[13] Nicolas T. Courtois. How Fast can be Algebraic Attacks on Block Ciphers? In Eli Biham, Helena Handschuh, Stefan Lucks, and Vincent Rijmen, editors, *Symmetric Cryptography*, number 07021 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2007. Internationales Begegnungs-

und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany. URL http://drops.dagstuhl.de/opus/volltexte/2007/1013.

[14] 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. doi: 10.1007/978-3-662-04722-4. URL https://doi.org/10.1007/978-3-662-04722-4.

[15] Ivan Damgård. A Design Principle for Hash Functions. In Brassard [11], pages 416–427. ISBN 3-540-97317-6. doi: 10.1007/0-387-34805-0_39. URL https://doi.org/10.1007/0-387-34805-0_39.

[16] Nicolaas Govert de Bruijn. Some direct decompositions of the set of integers. *Mathematics of Computation*, 18(88):537–546, 1964.

[17] Des. Data Encryption Standard. In *In FIPS PUB 46, Federal Information Processing Standards Publication*, pages 46–2, 1977.

[18] Itai Dinur, Orr Dunkelman, Masha Gutman, and Adi Shamir. Improved Top-Down Techniques in Differential Cryptanalysis. In Kristin E. Lauter and Francisco Rodríguez-Henríquez, editors, *Progress in Cryptology - LATINCRYPT 2015 - 4th International Conference on Cryptology and Information Security in Latin America, Guadalajara, Mexico, August 23-26, 2015, Proceedings*, volume 9230 of *Lecture Notes in Computer Science*, pages 139–156. Springer, 2015. ISBN 978-3-319-22173-1. doi: 10.1007/978-3-319-22174-8_8. URL https://doi.org/10.1007/978-3-319-22174-8_8.

[19] Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009. ISBN 978-0-521-89806-5. URL http://www.cambridge.org/uk/catalogue/catalogue.asp?isbn=9780521898065.

[20] Anna Gál and Peter Bro Miltersen. The cell probe complexity of succinct data structures. In *International Colloquium on Automata, Languages, and Programming*, pages 332–344. Springer, 2003.

[21] Rosario Gennaro and Matthew Robshaw, editors. *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, 2015. Springer. ISBN 978-3-662-47988-9. doi: 10.1007/978-3-662-47989-6. URL https://doi.org/10.1007/978-3-662-47989-6.

[22] Howard M. Heys and Stafford E. Tavares. Substitution-Permutation Networks Resistant to Differential and Linear Cryptanalysis. *J. Cryptology*, 9(1):1–19, 1996. doi: 10.1007/BF02254789. URL https://doi.org/10.1007/BF02254789.

[23] Deukjo Hong, Jaechul Sung, Seokhie Hong, Jongin Lim, Sangjin Lee, Bonseok Koo, Changhoon Lee, Donghoon Chang, Jesang Lee, Kitae Jeong, Hyun Kim, Jongsung Kim, and Seongtaek Chee. HIGHT: A New Block Cipher Suitable for Low-Resource Device. In Louis Goubin and Mitsuru Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, volume 4249 of *Lecture Notes in Computer Science*, pages 46–59. Springer, 2006. ISBN 3-540-46559-6. doi: 10.1007/11894063_4. URL https://doi.org/10.1007/11894063_4.

[24] Deukjo Hong, Jung-Keun Lee, Dong-Chan Kim, Daesung Kwon, Kwon Ho Ryu, and Donggeon Lee. LEA: A 128-Bit Block Cipher for Fast Encryption on Common Processors. In Yongdae Kim, Heejo Lee, and Adrian Perrig, editors, *Information Security Applications - 14th International Workshop, WISA 2013, Jeju Island, Korea, August 19-21, 2013, Revised Selected Papers*, volume 8267 of *Lecture Notes in Computer Science*, pages 3–27. Springer, 2013. ISBN 978-3-319-05148-2. doi: 10.1007/978-3-319-05149-9_1. URL https://doi.org/10.1007/978-3-319-05149-9_1.

[25] Auguste Kerckhoffs. *La cryptographie militaire*, volume 9. University Microfilms, 1883.

[26] Neal Koblitz and Alfred Menezes. Another Look at "Provable Security". *J. Cryptology*, 20(1):3–37, 2007. doi: 10.1007/s00145-005-0432-z. URL https://doi.org/10.1007/s00145-005-0432-z.

[27] Zhengbin Liu, Yongqiang Li, and Mingsheng Wang. Optimal Differential Trails in SIMON-like Ciphers. *IACR Trans. Symmetric Cryptol.*, 2017(1): 358–379, 2017. URL http://tosc.iacr.org/index.php/ToSC/article/view/598.

[28] Michael Luby and Charles Rackoff. How to Construct Pseudo-Random Permutations from Pseudo-Random Functions. In Hugh C. Williams, editor, *Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings*, volume 218 of *Lecture Notes in Computer Science*, page 447. Springer, 1985. ISBN 3-540-16463-4. doi: 10.1007/3-540-39799-X_34. URL https://doi.org/10.1007/3-540-39799-X_34.

[29] Mitsuru Matsui. Linear Cryptanalysis Method for DES Cipher. In Tor Helleseth, editor, *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer, 1993. ISBN 3-540-57600-2. doi: 10.1007/3-540-48285-7_33. URL https://doi.org/10.1007/3-540-48285-7_33.

[30] Mitsuru Matsui. On Correlation Between the Order of S-boxes and the Strength of DES. In Alfredo De Santis, editor, *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application*

of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings, volume 950 of *Lecture Notes in Computer Science*, pages 366–375. Springer, 1994. ISBN 3-540-60176-7. doi: 10.1007/BFb0053451. URL https://doi.org/10.1007/BFb0053451.

[31] Ralph C. Merkle. One Way Hash Functions and DES. In Brassard [11], pages 428–446. ISBN 3-540-97317-6. doi: 10.1007/0-387-34805-0_40. URL https://doi.org/10.1007/0-387-34805-0_40.

[32] Brice Minaud and Yannick Seurin. The Iterated Random Permutation Problem with Applications to Cascade Encryption. In Gennaro and Robshaw [21], pages 351–367. ISBN 978-3-662-47988-9. doi: 10.1007/978-3-662-47989-6_17. URL https://doi.org/10.1007/978-3-662-47989-6_17.

[33] Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and Linear Cryptanalysis Using Mixed-Integer Linear Programming. In Chuankun Wu, Moti Yung, and Dongdai Lin, editors, *Information Security and Cryptology - 7th International Conference, Inscrypt 2011, Beijing, China, November 30 - December 3, 2011. Revised Selected Papers*, volume 7537 of *Lecture Notes in Computer Science*, pages 57–76. Springer, 2011. ISBN 978-3-642-34703-0. doi: 10.1007/978-3-642-34704-7_5. URL https://doi.org/10.1007/978-3-642-34704-7_5.

[34] Mridul Nandi. A Simple Proof of a Distinguishing Bound of Iterated Uniform Random Permutation. *IACR Cryptology ePrint Archive*, 2015:579, 2015. URL http://eprint.iacr.org/2015/579.

[35] Kaisa Nyberg. Perfect Nonlinear S-Boxes. In Donald W. Davies, editor, *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings*, volume 547 of *Lecture Notes in Computer Science*, pages 378–386. Springer, 1991. ISBN 3-540-54620-0. doi: 10.1007/3-540-46416-6_32. URL https://doi.org/10.1007/3-540-46416-6_32.

[36] Kaisa Nyberg. S-boxes and Round Functions with Controllable Linearity and Differential Uniformity. In Preneel [39], pages 111–130. doi: 10.1007/3-540-60590-8_9. URL https://doi.org/10.1007/3-540-60590-8_9.

[37] Kaisa Nyberg. Generalized Feistel Networks. In Kwangjo Kim and Tsutomu Matsumoto, editors, *Advances in Cryptology - ASIACRYPT '96, International Conference on the Theory and Applications of Cryptology and Information Security, Kyongju, Korea, November 3-7, 1996, Proceedings*, volume 1163 of *Lecture Notes in Computer Science*, pages 91–104. Springer, 1996. ISBN 3-540-61872-4. doi: 10.1007/BFb0034838. URL https://doi.org/10.1007/BFb0034838.

[38] Jacques Patarin. Generic Attacks on Feistel Schemes. In Colin Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference*

on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings, volume 2248 of *Lecture Notes in Computer Science*, pages 222–238. Springer, 2001. ISBN 3-540-42987-5. doi: 10.1007/3-540-45682-1_14. URL https://doi.org/10.1007/3-540-45682-1_14.

[39] Bart Preneel, editor. *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings*, volume 1008 of *Lecture Notes in Computer Science*, 1995. Springer. doi: 10.1007/3-540-60590-8. URL https://doi.org/10.1007/3-540-60590-8.

[40] Ronald L. Rivest. The RC5 Encryption Algorithm. In Preneel [39], pages 86–96. doi: 10.1007/3-540-60590-8_7. URL https://doi.org/10.1007/3-540-60590-8_7.

[41] Yu Sasaki, Yosuke Todo, Kazumaro Aoki, Yusuke Naito, Takeshi Sugawara, Yumiko Murakami, Mitsuru Matsui, and Shoichi Hirose. Minalpher. *A submission to CAESAR*, 2014.

[42] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson. Twofish: A 128-bit block cipher. *NIST AES Proposal*, 15, 1998.

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

[44] 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 Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 158–178. Springer, 2014. ISBN 978-3-662-45610-1. doi: 10.1007/978-3-662-45611-8_9. URL https://doi.org/10.1007/978-3-662-45611-8_9.

[45] Qingju Wang, Zhiqiang Liu, Kerem Varici, Yu Sasaki, Vincent Rijmen, and Yosuke Todo. Cryptanalysis of Reduced-Round SIMON32 and SIMON48. In Willi Meier and Debdeep Mukhopadhyay, editors, *Progress in Cryptology - INDOCRYPT 2014 - 15th International Conference on Cryptology in India, New Delhi, India, December 14-17, 2014, Proceedings*, volume 8885 of *Lecture Notes in Computer Science*, pages 143–160. Springer, 2014. ISBN 978-3-319-13038-5. doi: 10.1007/978-3-319-13039-2_9. URL https://doi.org/10.1007/978-3-319-13039-2_9.

[46] Gangqiang Yang, Bo Zhu, Valentin Suder, Mark D. Aagaard, and Guang Gong. The Simeck Family of Lightweight Block Ciphers. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded*

Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings, volume 9293 of *Lecture Notes in Computer Science*, pages 307–329. Springer, 2015. ISBN 978-3-662-48323-7. doi: 10.1007/978-3-662-48324-4_16. URL https://doi.org/10.1007/978-3-662-48324-4_16.

10 Appendices

A S-boxes for ATC

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	00	04	00	04	08	0c	08	0c	00	04	08	0c	08	0c	00	04
01	00	06	00	06	08	0e	08	0e	00	06	08	0e	08	0e	00	06
02	00	04	04	00	08	0c	0c	08	00	04	0c	08	08	0c	04	00
03	00	06	04	02	08	0e	0c	0a	00	06	0c	0a	08	0e	04	02
04	01	05	01	05	09	0d	09	0d	01	05	09	0d	09	0d	01	05
05	01	07	01	07	09	0f	09	0f	01	07	09	0f	09	0f	01	07
06	01	05	05	01	09	0d	0d	09	01	05	0d	09	09	0d	05	01
07	01	07	05	03	09	0f	0d	0b	01	07	0d	0b	09	0f	05	03
08	02	06	02	06	0a	0e	0a	0e	03	07	0b	0f	0b	0f	03	07
09	02	04	02	04	0a	0c	0a	0c	03	05	0b	0d	0b	0d	03	05
0a	02	06	06	02	0a	0e	0e	0a	03	07	0f	0b	0b	0f	07	03
0b	02	04	06	00	0a	0c	0e	08	03	05	0f	09	0b	0d	07	01
0c	03	07	03	07	0b	0f	0b	0f	02	06	0a	0e	0a	0e	02	06
0d	03	05	03	05	0b	0d	0b	0d	02	04	0a	0c	0a	0c	02	04
0e	03	07	07	03	0b	0f	0f	0b	02	06	0e	0a	0a	0e	06	02
0f	03	05	07	01	0b	0d	0f	09	02	04	0e	08	0a	0c	06	00

Table 10: S-box S_1

B Unit vectors for key differences generating cycles in the Simon32 key schedule

Unit vectors for keys generating cycles with phase 1 [2]			
1010101010101010	1010101010101010	1010101010101010	1010101010101010
0101010101010101	0101010101010101	0101010101010101	0101010101010101
Unit vectors for keys generating cycles with phase 2 [4]			
1010101010101010	0000000000000000	1010101010101010	0000000000000000
0101010101010101	0000000000000000	0101010101010101	0000000000000000
0000000000000000	1010101010101010	0000000000000000	1010101010101010
0000000000000000	0101010101010101	0000000000000000	0101010101010101
Unit vectors for keys generating cycles with phase 3 [2]			
1010101010101010	1010101010101010	1010101010101010	1010101010101010
0101010101010101	0101010101010101	0101010101010101	0101010101010101
Unit vectors for keys generating cycles with phase 4 [8]			
1001100110011001	0000000000000000	0011001100110011	0000000000000000
0101010101010101	0000000000000000	0101010101010101	0000000000000000
0011001100110011	0000000000000000	0110011001100110	0000000000000000

0000000000000000	1001100110011001	0000000000000000	0011001100110011
0000000000000000	0101010101010101	0000000000000000	0101010101010101
0000000000000000	0011001100110011	0000000000000000	0110011001100110
0000000000000000	0000000000000000	1111111111111111	0000000000000000
0000000000000000	0000000000000000	0000000000000000	1111111111111111

Unit vectors for keys generating cycles with phase 5 [2]

1010101010101010	1010101010101010	1010101010101010	1010101010101010
0101010101010101	0101010101010101	0101010101010101	0101010101010101

Unit vectors for keys generating cycles with phase 6 [4]

1010101010101010	0000000000000000	1010101010101010	0000000000000000
0101010101010101	0000000000000000	0101010101010101	0000000000000000
0000000000000000	1010101010101010	0000000000000000	1010101010101010
0000000000000000	0101010101010101	0000000000000000	0101010101010101

Unit vectors for keys generating cycles with phase 7 [2]

1010101010101010	1010101010101010	1010101010101010	1010101010101010
0101010101010101	0101010101010101	0101010101010101	0101010101010101

Unit vectors for keys generating cycles with phase 8 [16]

1000011110000111	0000000000000000	0000111100001111	0001000100010001
0100010001000100	0000000000000000	0001000100010001	0000000000000000
0010001000100010	0000000000000000	0001000100010001	0000000000000000
0001000100010001	0000000000000000	0001000100010001	0000000000000000
0000111100001111	0000000000000000	0001111000011110	0001000100010001
0000000000000000	1000011110000111	0001000100010001	0000111100001111
0000000000000000	0100010001000100	0000000000000000	0001000100010001
0000000000000000	0010001000100010	0000000000000000	0001000100010001
0000000000000000	0001000100010001	0000000000000000	0001000100010001
0000000000000000	0000111100001111	0001000100010001	0001111000011110
0000000000000000	0000000000000000	1001100110011001	0000000000000000
0000000000000000	0000000000000000	0101010101010101	0000000000000000
0000000000000000	0000000000000000	0011001100110011	0000000000000000
0000000000000000	0000000000000000	0000000000000000	1001100110011001
0000000000000000	0000000000000000	0000000000000000	0101010101010101
0000000000000000	0000000000000000	0000000000000000	0011001100110011

Unit vectors for keys generating cycles with phase 9 [2]

1010101010101010	1010101010101010	1010101010101010	1010101010101010
0101010101010101	0101010101010101	0101010101010101	0101010101010101

Unit vectors for keys generating cycles with phase 10 [4]

1010101010101010	0000000000000000	1010101010101010	0000000000000000
0101010101010101	0000000000000000	0101010101010101	0000000000000000
0000000000000000	1010101010101010	0000000000000000	1010101010101010
0000000000000000	0101010101010101	0000000000000000	0101010101010101

Unit vectors for keys generating cycles with phase 11 [2]

1010101010101010	1010101010101010	1010101010101010	1010101010101010
0101010101010101	0101010101010101	0101010101010101	0101010101010101

Unit vectors for keys generating cycles with phase 12 [8]

1001100110011001	0000000000000000	0011001100110011	0000000000000000
0101010101010101	0000000000000000	0101010101010101	0000000000000000
0011001100110011	0000000000000000	0110011001100110	0000000000000000
0000000000000000	1001100110011001	0000000000000000	0011001100110011
0000000000000000	0101010101010101	0000000000000000	0101010101010101
0000000000000000	0011001100110011	0000000000000000	0110011001100110
0000000000000000	0000000000000000	1111111111111111	0000000000000000
0000000000000000	0000000000000000	0000000000000000	1111111111111111

Unit vectors for keys generating cycles with phase 13 [2]

1010101010101010	1010101010101010	1010101010101010	1010101010101010
0101010101010101	0101010101010101	0101010101010101	0101010101010101

Unit vectors for keys generating cycles with phase 14 [4]

1010101010101010	0000000000000000	1010101010101010	0000000000000000
0101010101010101	0000000000000000	0101010101010101	0000000000000000
0000000000000000	1010101010101010	0000000000000000	1010101010101010
0000000000000000	0101010101010101	0000000000000000	0101010101010101

Unit vectors for keys generating cycles with phase 15 [2]

1010101010101010	1010101010101010	1010101010101010	1010101010101010
0101010101010101	0101010101010101	0101010101010101	0101010101010101

Unit vectors for keys generating cycles with phase 16 [32]

1000000001111111	0000000000000000	0000000011111111	0000000000000000
0100000001000000	0000000000000000	0000000100000001	0000000000000000
0010000000100000	0000000000000000	0000000100000001	0000000000000000
0001000000010000	0000000000000000	0000000100000001	0000000000000000
0000100000001000	0000000000000000	0000000100000001	0000000000000000
0000010000000100	0000000000000000	0000000100000001	0000000000000000
0000001000000010	0000000000000000	0000000100000001	0000000000000000
0000000100000001	0000000000000000	0000000100000001	0000000000000000
0000000011111111	0000000000000000	0000000111111110	0000000000000000
0000000000000000	1000000001111111	0000000000000000	0000000011111111
0000000000000000	0100000001000000	0000000000000000	0000000100000001
0000000000000000	0010000000100000	0000000000000000	0000000100000001
0000000000000000	0001000000010000	0000000000000000	0000000100000001
0000000000000000	0000100000001000	0000000000000000	0000000100000001
0000000000000000	0000010000000100	0000000000000000	0000000100000001
0000000000000000	0000001000000010	0000000000000000	0000000100000001
0000000000000000	0000000100000001	0000000000000000	0000000100000001
0000000000000000	0000000011111111	0000000000000000	0000000111111110
0000000000000000	0000000000000000	1000000110000001	0000000000000000
0000000000000000	0000000000000000	0100000101000001	0000000000000000
0000000000000000	0000000000000000	0010000100100001	0000000000000000
0000000000000000	0000000000000000	0001000100010001	0000000000000000
0000000000000000	0000000000000000	0000100100001001	0000000000000000
0000000000000000	0000000000000000	0000010100000101	0000000000000000
0000000000000000	0000000000000000	0000001100000011	0000000000000000

0000000000000000	0000000000000000	0000000000000000	1000000110000001
0000000000000000	0000000000000000	0000000000000000	0100000101000001
0000000000000000	0000000000000000	0000000000000000	0010000100100001
0000000000000000	0000000000000000	0000000000000000	0001000100010001
0000000000000000	0000000000000000	0000000000000000	0000100100001001
0000000000000000	0000000000000000	0000000000000000	0000010100000101
0000000000000000	0000000000000000	0000000000000000	0000001100000011

Unit vectors for keys generating cycles with phase 17 [2]

1010101010101010	1010101010101010	1010101010101010	1010101010101010
0101010101010101	0101010101010101	0101010101010101	0101010101010101

Unit vectors for keys generating cycles with phase 18 [4]

1010101010101010	0000000000000000	1010101010101010	0000000000000000
0101010101010101	0000000000000000	0101010101010101	0000000000000000
0000000000000000	1010101010101010	0000000000000000	1010101010101010
0000000000000000	0101010101010101	0000000000000000	0101010101010101

Unit vectors for keys generating cycles with phase 19 [2]

1010101010101010	1010101010101010	1010101010101010	1010101010101010
0101010101010101	0101010101010101	0101010101010101	0101010101010101

Unit vectors for keys generating cycles with phase 20 [8]

1001100110011001	0000000000000000	0011001100110011	0000000000000000
0101010101010101	0000000000000000	0101010101010101	0000000000000000
0011001100110011	0000000000000000	0110011001100110	0000000000000000
0000000000000000	1001100110011001	0000000000000000	0011001100110011
0000000000000000	0101010101010101	0000000000000000	0101010101010101
0000000000000000	0011001100110011	0000000000000000	0110011001100110
0000000000000000	0000000000000000	1111111111111111	0000000000000000
0000000000000000	0000000000000000	0000000000000000	1111111111111111

Unit vectors for keys generating cycles with phase 21 [2]

1010101010101010	1010101010101010	1010101010101010	1010101010101010
0101010101010101	0101010101010101	0101010101010101	0101010101010101

Unit vectors for keys generating cycles with phase 22 [4]

1010101010101010	0000000000000000	1010101010101010	0000000000000000
0101010101010101	0000000000000000	0101010101010101	0000000000000000
0000000000000000	1010101010101010	0000000000000000	1010101010101010
0000000000000000	0101010101010101	0000000000000000	0101010101010101

Unit vectors for keys generating cycles with phase 23 [2]

1010101010101010	1010101010101010	1010101010101010	1010101010101010
0101010101010101	0101010101010101	0101010101010101	0101010101010101

Unit vectors for keys generating cycles with phase 24 [16]

1000011110000111	0000000000000000	0000111100001111	0001000100010001
0100010001000100	0000000000000000	0001000100010001	0000000000000000
0010001000100010	0000000000000000	0001000100010001	0000000000000000
0001000100010001	0000000000000000	0001000100010001	0000000000000000
0000111100001111	0000000000000000	0001111000011110	0001000100010001
0000000000000000	1000011110000111	0001000100010001	0000111100001111

0000000000000000	0100010001000100	0000000000000000	0001000100010001
0000000000000000	0010001000100010	0000000000000000	0001000100010001
0000000000000000	0001000100010001	0000000000000000	0001000100010001
0000000000000000	0000111100001111	0001000100010001	0001111000011110
0000000000000000	0000000000000000	1001100110011001	0000000000000000
0000000000000000	0000000000000000	0101010101010101	0000000000000000
0000000000000000	0000000000000000	0011001100110011	0000000000000000
0000000000000000	0000000000000000	0000000000000000	1001100110011001
0000000000000000	0000000000000000	0000000000000000	0101010101010101
0000000000000000	0000000000000000	0000000000000000	0011001100110011

Unit vectors for keys generating cycles with phase 25 [2]

1010101010101010	1010101010101010	1010101010101010	1010101010101010
0101010101010101	0101010101010101	0101010101010101	0101010101010101

Unit vectors for keys generating cycles with phase 26 [4]

1010101010101010	0000000000000000	1010101010101010	0000000000000000
0101010101010101	0000000000000000	0101010101010101	0000000000000000
0000000000000000	1010101010101010	0000000000000000	1010101010101010
0000000000000000	0101010101010101	0000000000000000	0101010101010101

Unit vectors for keys generating cycles with phase 27 [2]

1010101010101010	1010101010101010	1010101010101010	1010101010101010
0101010101010101	0101010101010101	0101010101010101	0101010101010101

Unit vectors for keys generating cycles with phase 28 [8]

1001100110011001	0000000000000000	0011001100110011	0000000000000000
0101010101010101	0000000000000000	0101010101010101	0000000000000000
0011001100110011	0000000000000000	0110011001100110	0000000000000000
0000000000000000	1001100110011001	0000000000000000	0011001100110011
0000000000000000	0101010101010101	0000000000000000	0101010101010101
0000000000000000	0011001100110011	0000000000000000	0110011001100110
0000000000000000	0000000000000000	1111111111111111	0000000000000000
0000000000000000	0000000000000000	0000000000000000	1111111111111111

Unit vectors for keys generating cycles with phase 29 [2]

1010101010101010	1010101010101010	1010101010101010	1010101010101010
0101010101010101	0101010101010101	0101010101010101	0101010101010101

Unit vectors for keys generating cycles with phase 30 [4]

1010101010101010	0000000000000000	1010101010101010	0000000000000000
0101010101010101	0000000000000000	0101010101010101	0000000000000000
0000000000000000	1010101010101010	0000000000000000	1010101010101010
0000000000000000	0101010101010101	0000000000000000	0101010101010101

Unit vectors for keys generating cycles with phase 31 [2]

1010101010101010	1010101010101010	1010101010101010	1010101010101010
0101010101010101	0101010101010101	0101010101010101	0101010101010101

Unit vectors for keys generating cycles with phase 32 [62]

1000000000000000	0000000000000000	0000000000000001	0000000000000000
0100000000000000	0000000000000000	0000000000000001	0000000000000000
0010000000000000	0000000000000000	0000000000000001	0000000000000000

0001000000000000 0000000000000000 0000000000000001 0000000000000000
0000100000000000 0000000000000000 0000000000000001 0000000000000000
0000010000000000 0000000000000000 0000000000000001 0000000000000000
0000001000000000 0000000000000000 0000000000000001 0000000000000000
0000000100000000 0000000000000000 0000000000000001 0000000000000000
0000000010000000 0000000000000000 0000000000000001 0000000000000000
0000000001000000 0000000000000000 0000000000000001 0000000000000000
0000000000100000 0000000000000000 0000000000000001 0000000000000000
0000000000010000 0000000000000000 0000000000000001 0000000000000000
0000000000001000 0000000000000000 0000000000000001 0000000000000000
0000000000000100 0000000000000000 0000000000000001 0000000000000000
0000000000000010 0000000000000000 0000000000000001 0000000000000000
0000000000000001 0000000000000000 0000000000000001 0000000000000000
0000000000000000 1000000000000000 0000000000000000 0000000000000001
0000000000000000 0100000000000000 0000000000000000 0000000000000001
0000000000000000 0010000000000000 0000000000000000 0000000000000001
0000000000000000 0001000000000000 0000000000000000 0000000000000001
0000000000000000 0000100000000000 0000000000000000 0000000000000001
0000000000000000 0000010000000000 0000000000000000 0000000000000001
0000000000000000 0000001000000000 0000000000000000 0000000000000001
0000000000000000 0000000100000000 0000000000000000 0000000000000001
0000000000000000 0000000010000000 0000000000000000 0000000000000001
0000000000000000 0000000001000000 0000000000000000 0000000000000001
0000000000000000 0000000000100000 0000000000000000 0000000000000001
0000000000000000 0000000000010000 0000000000000000 0000000000000001
0000000000000000 0000000000001000 0000000000000000 0000000000000001
0000000000000000 0000000000000100 0000000000000000 0000000000000001
0000000000000000 0000000000000010 0000000000000000 0000000000000001
0000000000000000 0000000000000001 0000000000000000 0000000000000001
0000000000000000 0000000000000000 1000000000000001 0000000000000000
0000000000000000 0000000000000000 0100000000000001 0000000000000000
0000000000000000 0000000000000000 0010000000000001 0000000000000000
0000000000000000 0000000000000000 0001000000000001 0000000000000000
0000000000000000 0000000000000000 0000100000000001 0000000000000000
0000000000000000 0000000000000000 0000010000000001 0000000000000000
0000000000000000 0000000000000000 0000001000000001 0000000000000000
0000000000000000 0000000000000000 0000000100000001 0000000000000000
0000000000000000 0000000000000000 0000000010000001 0000000000000000
0000000000000000 0000000000000000 0000000001000001 0000000000000000
0000000000000000 0000000000000000 0000000000100001 0000000000000000
0000000000000000 0000000000000000 0000000000010001 0000000000000000
0000000000000000 0000000000000000 0000000000001001 0000000000000000
0000000000000000 0000000000000000 0000000000000101 0000000000000000
0000000000000000 0000000000000000 0000000000000011 0000000000000000
0000000000000000 0000000000000000 0000000000000000 1000000000000001
0000000000000000 0000000000000000 0000000000000000 0100000000000001

0000000000000000	0000000000000000	0000000000000000	0010000000000001
0000000000000000	0000000000000000	0000000000000000	0001000000000001
0000000000000000	0000000000000000	0000000000000000	0000100000000001
0000000000000000	0000000000000000	0000000000000000	0000010000000001
0000000000000000	0000000000000000	0000000000000000	0000001000000001
0000000000000000	0000000000000000	0000000000000000	0000000100000001
0000000000000000	0000000000000000	0000000000000000	0000000010000001
0000000000000000	0000000000000000	0000000000000000	0000000001000001
0000000000000000	0000000000000000	0000000000000000	0000000000100001
0000000000000000	0000000000000000	0000000000000000	0000000000010001
0000000000000000	0000000000000000	0000000000000000	0000000000001001
0000000000000000	0000000000000000	0000000000000000	0000000000000101
0000000000000000	0000000000000000	0000000000000000	0000000000000011

Unit vectors for the keys generating cycles with phase 64 [64]

1000000000000000	0000000000000000	0000000000000000	0000000000000000
0100000000000000	0000000000000000	0000000000000000	0000000000000000
0010000000000000	0000000000000000	0000000000000000	0000000000000000
0001000000000000	0000000000000000	0000000000000000	0000000000000000
0000100000000000	0000000000000000	0000000000000000	0000000000000000
0000010000000000	0000000000000000	0000000000000000	0000000000000000
0000001000000000	0000000000000000	0000000000000000	0000000000000000
0000000100000000	0000000000000000	0000000000000000	0000000000000000
0000000010000000	0000000000000000	0000000000000000	0000000000000000
0000000001000000	0000000000000000	0000000000000000	0000000000000000
0000000000100000	0000000000000000	0000000000000000	0000000000000000
0000000000010000	0000000000000000	0000000000000000	0000000000000000
0000000000001000	0000000000000000	0000000000000000	0000000000000000
0000000000000100	0000000000000000	0000000000000000	0000000000000000
0000000000000010	0000000000000000	0000000000000000	0000000000000000
0000000000000001	0000000000000000	0000000000000000	0000000000000000
0000000000000000	1000000000000000	0000000000000000	0000000000000000
0000000000000000	0100000000000000	0000000000000000	0000000000000000
0000000000000000	0010000000000000	0000000000000000	0000000000000000
0000000000000000	0001000000000000	0000000000000000	0000000000000000
0000000000000000	0000100000000000	0000000000000000	0000000000000000
0000000000000000	0000010000000000	0000000000000000	0000000000000000
0000000000000000	0000001000000000	0000000000000000	0000000000000000
0000000000000000	0000000100000000	0000000000000000	0000000000000000
0000000000000000	0000000010000000	0000000000000000	0000000000000000
0000000000000000	0000000001000000	0000000000000000	0000000000000000
0000000000000000	0000000000100000	0000000000000000	0000000000000000
0000000000000000	0000000000010000	0000000000000000	0000000000000000
0000000000000000	0000000000001000	0000000000000000	0000000000000000
0000000000000000	0000000000000100	0000000000000000	0000000000000000
0000000000000000	0000000000000010	0000000000000000	0000000000000000

```
0000000000000000  0000000000000001  0000000000000000  0000000000000000
0000000000000000  0000000000000000  1000000000000000  0000000000000000
0000000000000000  0000000000000000  0100000000000000  0000000000000000
0000000000000000  0000000000000000  0010000000000000  0000000000000000
0000000000000000  0000000000000000  0001000000000000  0000000000000000
0000000000000000  0000000000000000  0000100000000000  0000000000000000
0000000000000000  0000000000000000  0000010000000000  0000000000000000
0000000000000000  0000000000000000  0000001000000000  0000000000000000
0000000000000000  0000000000000000  0000000100000000  0000000000000000
0000000000000000  0000000000000000  0000000010000000  0000000000000000
0000000000000000  0000000000000000  0000000001000000  0000000000000000
0000000000000000  0000000000000000  0000000000100000  0000000000000000
0000000000000000  0000000000000000  0000000000010000  0000000000000000
0000000000000000  0000000000000000  0000000000001000  0000000000000000
0000000000000000  0000000000000000  0000000000000100  0000000000000000
0000000000000000  0000000000000000  0000000000000010  0000000000000000
0000000000000000  0000000000000000  0000000000000001  0000000000000000
0000000000000000  0000000000000000  0000000000000000  1000000000000000
0000000000000000  0000000000000000  0000000000000000  0100000000000000
0000000000000000  0000000000000000  0000000000000000  0010000000000000
0000000000000000  0000000000000000  0000000000000000  0001000000000000
0000000000000000  0000000000000000  0000000000000000  0000100000000000
0000000000000000  0000000000000000  0000000000000000  0000010000000000
0000000000000000  0000000000000000  0000000000000000  0000001000000000
0000000000000000  0000000000000000  0000000000000000  0000000100000000
0000000000000000  0000000000000000  0000000000000000  0000000010000000
0000000000000000  0000000000000000  0000000000000000  0000000001000000
0000000000000000  0000000000000000  0000000000000000  0000000000100000
0000000000000000  0000000000000000  0000000000000000  0000000000010000
0000000000000000  0000000000000000  0000000000000000  0000000000001000
0000000000000000  0000000000000000  0000000000000000  0000000000000100
0000000000000000  0000000000000000  0000000000000000  0000000000000010
0000000000000000  0000000000000000  0000000000000000  0000000000000001
```

C RoadRunneR

This paper resulted from a workshop at the Lorentz Center that I participated in during my masters project.

Refined Probability of Differential Characteristics Including Dependency Between Multiple Rounds

Anne Canteaut[1], Eran Lambooij[2], Samuel Neves[3], Shahram Rasoolzadeh[4], Yu Sasaki[5] and Marc Stevens[6]

[1] Inria, France, Anne.Canteaut@inria.fr
[2] Technische Universiteit Eindhoven, The Netherlands, e.lambooij@student.tue.nl
[3] CISUC, Dept. of Informatics Engineering, University of Coimbra, Portugal, sneves@dei.uc.pt
[4] Ruhr-Universität Bochum, Germany, Rasoolzadeh.shahram@gmail.com
[5] NTT Secure Platform Laboratories, Japan, sasaki.yu@lab.ntt.co.jp
[6] CWI Amsterdam, The Netherlands, marc.stevens@cwi.nl

Abstract. The current paper studies the probability of differential characteristics for an unkeyed (or with a fixed key) construction. Most notably, it focuses on the gap between two probabilities of differential characteristics: probability with independent S-box assumption, p_{ind}, and exact probability, p_{exact}. It turns out that p_{exact} is larger than p_{ind} in Feistel network with some S-box based inner function. The mechanism of this gap is then theoretically analyzed. The gap is derived from interaction of S-boxes in three rounds, and the gap depends on the size and choice of the S-box. In particular the gap can never be zero when the S-box is bigger than six bits. To demonstrate the power of this improvement, a related-key differential characteristic is proposed against a lightweight block cipher RoadRunneR. For the 128-bit key version, p_{ind} of 2^{-48} is improved to p_{exact} of 2^{-43}. For the 80-bit key version, p_{ind} of 2^{-68} is improved to p_{exact} of 2^{-62}. The analysis is further extended to SPN with an almost-MDS binary matrix in the core primitive of the authenticated encryption scheme Minalpher: p_{ind} of 2^{-128} is improved to p_{exact} of 2^{-96}, which allows to extend the attack by two rounds.

Keywords: differential cryptanalysis · independent S-box · fixed key · unkeyed construction · exact probability · RoadRunneR · Minalpher

1 Introduction

Differential cryptanalysis [BS90, BS93] is one of the most fundamental cryptanalytic approaches targeting symmetric-key primitives. While its basic concept in an idealized environment under several assumptions can easily be understood, predicting the actual behavior of concrete algorithms is quite complex and a lot of research has been done regarding this topic.

Most block ciphers are designed to iterate a small keyed permutation, called the round function, with many rounds being performed to build a conversion between the plaintext and ciphertext. The plaintext x_0 is updated by round function RF_i in the ith round by processing $x_{i+1} \leftarrow RF_i(x_i)$ for $i = 0, 1, 2, \cdots$. The most common approach for evaluating the effect of differential analysis consists in applying the Markov assumption to the cipher [LMM91] and evaluating the probability of differential propagation for each round. The probability of the differential characteristic over the entire cipher is then equal to the product of the probabilities of the differentials of all rounds.

Given a pair of differences (a_i, a_{i+1}), the corresponding probability $p_i \triangleq \Pr_{x\in\mathbf{P}}[RF_i(x)\oplus RF_i(x \oplus a_i) = a_{i+1}]$ is searched for each i, where $\mathbf{P}$ is the plaintext space, and $\Pi_i p_i$ is the probability of the characteristic $(a_0, a_1, \ldots, a_r)$ for the entire r-round cipher.

The hidden argument in the above explanation is the treatment of a key k or subkeys k_i. The Markov assumption can be established when the state x_i is first xored with a subkey k_i and all subkeys are chosen independently uniformly at random. Therefore, most analyses are based on bounds on the *expected* probability of a differential characteristic, i.e., the probability averaged over all keys. However, the implementation environment for symmetric-key primitives does not allow to store all independent subkeys, thus k_i is usually expanded from k, and the Markov assumption collapses.

Moreover, subkeys may not be xored in every round to all state bits, which can be seen in designs of lightweight cryptographic schemes such as Simon [BSS+13], SKINNY [BJK+16] and LED [GPPR11]. Also some primitives, like hash functions or Even-Mansour schemes [DKS12, EM91, EM97], are based on an iterated permutation which does not involve any key at all. In such a case, the evaluation using the Markov assumption may still give some insight about the security against differential analysis, but never leads to the exact probability of the differential propagation for multiple rounds.

To conclude, evaluating the probability of differential propagations for multiple rounds precisely without the Markov assumption is a big challenge.

1.1 Related Work on Precise Evaluation of Differential Probability

Our work then focuses on the evaluation of the probability of a differential characteristic for a primitive with a fixed key, or for a keyless primitive. It is worth noticing that both contexts are similar in the sense that the absence of a key can equivalently be seen as the insertion of an all-zero key. Conversely, a structure with a fixed key is equivalent to an unkeyed one with different building blocks. For instance, using an S-box S with a fixed round-key k is equivalent to using $S' : x \mapsto S_k(x)$ as an S-box without any key. Let E be a block cipher with a fixed key and let ΔP and ΔC be the plaintext and ciphertext differences, respectively. Suppose that the goal is to precisely evaluate the probability of $\Pr[E(x) \oplus E(x \oplus \Delta P) = \Delta C]$, where the probability is taken over all plaintexts x. Besides the issue of subkeys for multiple rounds, there are several aspects to precisely evaluate this probability.

The first issue we would like to mention is the contrast between differential characteristics and differential effect. The differential characteristics specify not only $(\Delta P, \Delta C)$ but also differences in intermediate states, often the initial difference in each round, and evaluate the probability of each section and multiplies all the probabilities. On the contrary, the differential effect sums up the probabilities of all possible differential characteristics, thus gives a more precise probability. A lot of research has been done to evaluate the exact maximum expected differential probability (and the maximum expected linear potential) in particular for AES, e.g. [HLL+00, KMT01, PSC+02, PSLL03, DR06, KS07, CR15], and for Feistel or MISTY networks, e.g. [NK92, Mat96]. Those researches are different from the current paper with respect to the point that all state bits are xored by subkeys which are assumed to be chosen independently and uniformly at random.

In contrary, our work focuses on determining the exact probability of a differential characteristic when the key is fixed. This fixed-key probability has been determined in a very few cases only. The most prominent example is the AES, for which the probabilities of 2-round characteristics have been determined, for all possible values of the key [DR07].

Alternative approaches can be used when such a theoretical analysis is out of reach. One approach is carrying out some experiment, which exhaustively chooses plaintexts $P \in \mathbf{P}$ and actually computes $E_K(x) \oplus E_K(x \oplus \Delta P)$. The experiment is then iterated for several keys (see e.g. [BG10]). The experiment can include any complex event, however, the lack of theoretical analysis limits its versatility to be applied to other ciphers. Of

course the approach can only be applied to ciphers with small block sizes, often 32-bit block sizes, such as SIMON and KATAN [DDGS15, CDK09]. Another approach introduced in [BBL13] consists in computing the maximal expected probability of a characteristic and deriving a bound on the probability of the existence of characteristics whose fixed-key probability exceeds a given value. This result can be used by designers to guarantee that characteristics with high probability are very unlikely. However, this bound exhibits a large gap between the fixed-key and the expected probabilities (see Table 1 in [BBL13]). It is then of little use to the cryptanalyst who needs to estimate the exact probability of some characteristic for a given key.

1.2 Our Contributions

In this paper, we evaluate the exact probabilities of the differential characteristics in some unkeyed constructions. In particular, we provide an in-depth study of the probabilities of the differential characteristics over three rounds of an unkeyed Feistel network. Most notably, when the inner function follows an SPN construction with an S-box having differential uniformity 4, the exact probability of a 3-round characteristic is either zero or a value which is greater than or equal to the usual estimate with independent S-box assumption, p_{ind}. A more thorough analysis is then provided when the inner function consists of a single n-bit S-box with differential uniformity 4. We show that, in this case, the exact probability of any 3-round characteristic with only active Sboxes is either zero, or exceeds p_{ind} by a factor of 2^{ℓ} where $\ell \geq \max(0, n-6)$.

The above analysis is then applied to the lightweight 64-bit block cipher RoadRunneR [BS15]. It adopts a Feistel construction and its inner function starts and ends with the S-box application without applying any subkey, therefore the above generic analysis can be applied. Although no security is claimed against related-key attacks, the designers mentioned related-key differential characteristics with 24 active S-boxes on the full (12) rounds of RoadRunneR-128, whose probability is expected to be $2^{-2\cdot 24} = 2^{-48}$. The designers also speculated that the number of active S-boxes could be reduced further with more careful analysis. In this paper, we first concretize the related-key characteristic with 24 active S-boxes and show that the exact probability is higher than the original expectation. The comparison of two probabilities is shown in Table 1. The attack is implemented up to 8 rounds and the improved factor is verified. We prove that the minimum number of active S-boxes is 24 by using a SAT solver, thus our characteristic is fairly tight.

Finding related-key differential characteristics is much harder in RoadRunneR-80 due to its key schedule. We propose an 8-round characteristic with $p_{\text{ind}} = 2^{-68}$ which are unlikely to be satisfied even with a full codebook, but the improvement with p_{exact} increases it to 2^{-62}.

We then extend the application of our observations to SPN-based structures with almost-MDS binary matrices. In particular, we analyze p_{exact} of the differential characteristic in an authenticated encryption scheme Minalpher [STA+14], which offers 128-bit security. The previous differential characteristic reaches 2^{-128} for 6 (out of 17.5) rounds. We show that for this characteristic a refined estimate of the exact probability is 2^{-96}. This significant increase enables us to extend the attack by two rounds. The comparison of the probabilities are given in Table 1.

1.3 Paper Outline

The paper is organized as follows. Section 2 provides theoretical analysis of p_{exact} for 3-round Feistel structure. Section 3 applies the observation to RoadRunneR with 128-bit key. Section 4 extends the application to SPN with almost-MDS matrices in Minalpher.

Table 1: Improved probability of characteristics for RoadRunneR-128 and Minalpher.

Rounds	1	2	3	4	5	6	7	8	9	10	11	12
RoadRunneR-128												
p_{ind}	-4	-8	-12	-16	-20	-24	-28	-32	-36	-40	-44	-48
p_{exact}	-4†	-8†	-12†	-15†	-19†	-22†	-26†	-29†	-33	-36	-40	-43
RoadRunneR-80												
p_{ind}	-8	-17	-26	-34	-42	-51	-60	-68				
p_{exact}	-8†	-17†	-25†	-32†	-39†	-47	-55	-62				
Minalpher												
p_{ind}	-16	-48	-64	-80	-112	-128						
p_{exact}	-16†	-40‡	-48	-64	-88	-96	-112	-128				

Numbers denote logarithm of the probabilities. Probabilities with † were experimentally verified. Probability with ‡ was experimentally verified only for the essential part, namely the probability of passing through S-boxes that are affected by our analysis was verified.

2 Probabilities of 3-Round Characteristics in some Keyless Feistel Networks

In this section, we evaluate the exact probability of a differential characteristic over three rounds of an unkeyed Feistel network whose inner function is seen as a single S-box application. We then want to determine the probability over all possible inputs (x_0, x_1) of the three-round characteristic depicted in Figure 1, where the difference at the output of the ith S-box is defined as $b_i = a_{i+1} \oplus a_{i-1}$. It is worth noticing that the differential probabilities for an unkeyed 3-round Feistel have been previously investigated in order to determine the smallest differential uniformity we can get for an S-box which follows this construction [LW14, CDL15]. However, these papers focus on the maximum possible probability for a 3-round differential characteristic, while we want to obtain a formula which captures any given characteristic.

Using that $x_3 = S(x_2) \oplus x_1$, we get that the probability of the three-round characteristic defined by $(a_0, \ldots, a_4)$ is equal to the following probability:

$$\begin{aligned} p_{\text{exact}} = &\Pr_{x_1, x_2 \in \mathbb{F}_2^n}[S(S(x_2) \oplus x_1 \oplus a_1 \oplus b_2) \oplus S(S(x_2) \oplus x_1) = b_3 \\ &\text{and } S(x_2 \oplus a_2) \oplus S(x_2) = b_2 \text{ and } S(x_1 \oplus a_1) \oplus S(x_1) = b_1]\,. \end{aligned}$$

We will show that this probability may differ from the usual estimate obtained when assuming that the inputs of the three S-boxes are independent, i.e. from

$$\begin{aligned} p_{\text{ind}} = &\Pr_{x_3 \in \mathbb{F}_2^n}[S(x_3 \oplus a_1 \oplus b_2) \oplus S(x_3) = b_3] \times \Pr_{x_2 \in \mathbb{F}_2^n}[S(x_2 \oplus a_2) \oplus S(x_2) = b_2] \\ &\times \Pr_{x_1 \in \mathbb{F}_2^n}[S(x_1 \oplus a_1) \oplus S(x_1) = b_1]\,. \end{aligned}$$

The difference between the two probabilities mainly comes from some dependencies due to the fact that the input of the S-box in the third round is the sum of two elements, x_1 and $S(x_2)$, where x_1 and x_2 respectively conform to the S-box differentials (a_1, b_1) and (a_2, b_2). Also, we show that the size of the S-box and, for a given size, the choice of the S-box may affect the factor between the exact probability and the usual estimate.

More precisely, we first show that, in many cases, including when S has an SPN structure based on an S-box with differential uniformity at most 4, the factor λ between these two probabilities is either zero or a power of 2 whose exponent corresponds to the dimension of a well-defined linear space. Most notably, if S corresponds to a single S-box with differential uniformity at most 4, then

$$p_{\text{exact}} = \lambda p_{\text{ind}}.$$

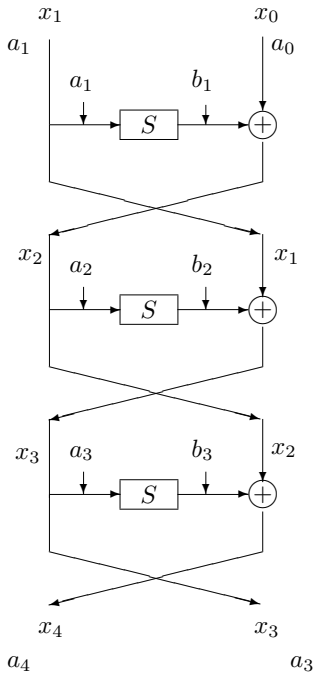

Figure 1: Differential characteristic of a three-round Feistel network where $b_i = a_{i+1} \oplus a_{i-1}$.

with $\lambda \in \{0, 2^\ell, \text{ with } \max(0, n-6) \leq \ell \leq n-2\}$, unless one of the three S-boxes in the differential path is inactive, which corresponds to $p_{\text{exact}} = p_{\text{ind}}$.

2.1 General result

The technique used in the proof is similar to the one used by Daemen and Rijmen for computing the fixed-key probabilities of the differentials over two rounds of the AES [DR07]. It mainly relies on the algebraic structure of the sets of inputs (resp. of outputs) of the S-box conforming to a given differential. These sets are defined as follows.

Definition 1. Let S be an n-bit to n-bit S-box. For any pair (a, b) of differences, we use the following notation:

$$\mathcal{X}_S(a,b) \triangleq \{x \in \mathbb{F}_2^n \colon S(x \oplus a) \oplus S(x) = b\},$$

and

$$\mathcal{Y}_S(a,b) \triangleq \{S(x) \in \mathbb{F}_2^n \colon S(x \oplus a) \oplus S(x) = b\}.$$

Remark 1. In the following, we will use some relationships between the sets $\mathcal{X}_S(a,b)$ and $\mathcal{Y}_S(a,b)$. Obviously,

$$\mathcal{Y}_S(a,b) = S\left(\mathcal{X}_S(a,b)\right) \ .$$

Moreover, if S is a permutation,

$$\mathcal{Y}_S(a,b) = \mathcal{X}_{S^{-1}}(b,a) \ .$$

Indeed, $y \in \mathcal{Y}_S(a,b)$ if and only if $x = S^{-1}(y)$ satisfies

$$S(x \oplus a) \oplus S(x) = b \ .$$

Then, we have

$$S(S^{-1}(y) \oplus a) = y \oplus b$$

which is equivalent to

$$S^{-1}(y) \oplus a = S^{-1}(y \oplus b)\,,$$

i.e., $y \in \mathcal{X}_{S^{-1}}(b, a)$.

Now, we focus on the following data transformation depicted in Figure 2:

$$z = S(x_2) \oplus x_1 \text{ such that } x_1 \in \mathcal{X}_S(a_1, b_1) \text{ and } x_2 \in \mathcal{X}_S(a_2, b_2).$$

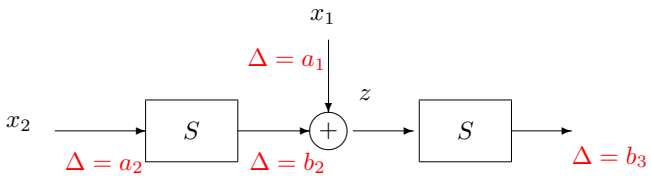

Figure 2: Target structure.

When the three sets $\mathcal{X}_S(a_1, b_1)$, $\mathcal{Y}_S(a_2, b_2)$ and $\mathcal{X}_S(a_1 \oplus b_2, b_3)$ are affine subspaces, we get the following result.

Theorem 1. *Let S be a permutation of $\mathbb{F}_2^n$, and let a_1, b_1, a_2, b_2, b_3 be five elements in $\mathbb{F}_2^n$. Assume that there exist $\alpha_1, \alpha_2, \alpha_3 \in \mathbb{F}_2^n$ and three linear subspaces $V_1, V_2, V_3 \subseteq \mathbb{F}_2^n$ such that*

$$\mathcal{X}_S(a_1, b_1) = \alpha_1 + V_1,\ \mathcal{Y}_S(a_2, b_2) = \alpha_2 + V_2,\ \text{and } \mathcal{X}_S(a_1 \oplus b_2, b_3) = \alpha_3 + V_3\,.$$

Then, the multiset

$$\{(x_1, x_2) \in \mathcal{X}_S(a_1, b_1) \times \mathcal{X}_S(a_2, b_2) \colon S(S(x_2) \oplus x_1 \oplus a_1 \oplus b_2) \oplus S(S(x_2) \oplus x_1) = b_3\}$$

is either empty or has size 2^d with

$$d = \dim V_1 + \dim V_2 + \dim V_3 - \dim(V_1 + V_2 + V_3)$$

where $V_1 + V_2 + V_3$ denotes the linear space formed by all elements of the form $v_1 + v_2 + v_3$ with $v_i \in V_i$.

Proof. We first observe that we do not need to restrict ourselves to the situation where the input differences of all S-boxes are nonzero. Indeed, if the input difference of one S-box is zero (i.e. $a_1 = 0$ or $a_2 = 0$ or $a_1 = b_2$), either the corresponding output difference is nonzero, which implies that $p_{\text{exact}} = 0$ and the multiset we consider is empty, or the corresponding output difference is zero, and the associated set (i.e. $\mathcal{X}_S(a_1, b_1)$ or $\mathcal{Y}_S(a_2, b_2)$ or $\mathcal{X}_S(a_1 \oplus b_2, b_3)$) satisfies the hypothesis since it equals the whole space $\mathbb{F}_2^n$.

Let us now define the following set (without multiplicity)

$$\mathcal{Z} = \{(S(x_2) \oplus x_1) \colon (x_1, x_2) \in \mathcal{X}_S(a_1, b_1) \times \mathcal{X}_S(a_2, b_2)\}\,.$$

Then, $\mathcal{Z} = (\alpha_1 \oplus \alpha_2) + (V_1 + V_2)$, and each element in $\mathcal{Z}$ corresponds to 2^r pairs (x_1, x_2) in $\mathcal{X}_S(a_1, b_1) \times \mathcal{X}_S(a_2, b_2)$ with $r = \dim V_1 + \dim V_2 - \dim(V_1 + V_2)$. We want to determine the size of the set

$$\mathcal{S} = \{z \in \mathcal{Z} \colon S(z \oplus a_1 \oplus b_2) \oplus S(z) = b_3\}\,.$$

Clearly, this set corresponds to the intersection between $\mathcal{Z}$ and $\mathcal{X}_S(a_1 \oplus b_2, b_3)$, which are both affine subspaces of $\mathbb{F}_2^n$. Since the intersection between two affine subspaces is either empty or a coset of the intersection between the corresponding linear subspaces, we deduce that, if $\mathcal{S} \neq \emptyset$, then there exists some s such that

$$\mathcal{S} = s + ((V_1 + V_2) \cap V_3)\,.$$

Recall that, for any two linear subspaces U and V,

$$\dim(U+V) = \dim U + \dim V - \dim(U \cap V)\,. \tag{1}$$

It follows from (1) that, if $\mathcal{S} \neq \emptyset$, we have

$$\dim \mathcal{S} = \dim((V_1 + V_2) \cap V_3) = \dim(V_1 + V_2) + \dim V_3 - \dim(V_1 + V_2 + V_3)\,.$$

Since each element in $\mathcal{Z}$ and then in $\mathcal{S}$ corresponds to 2^r pairs (x_1, x_2) in $\mathcal{X}_S(a_1, b_1) \times \mathcal{X}_S(a_2, b_2)$, we deduce that the multiset

$$\{(x_1, x_2) \in \mathcal{X}_S(a_1, b_1) \times \mathcal{X}_S(a_2, b_2) \colon S(S(x_2) \oplus x_1 \oplus a_1 \oplus b_2) \oplus S(S(x_2) \oplus x_1) = b_3\}$$

is either empty or has size 2^d with

$$\begin{aligned} d &= r + \dim \mathcal{S} \\ &= \dim V_1 + \dim V_2 - \dim(V_1 + V_2) + \dim(V_1 + V_2) + \dim V_3 - \dim(V_1 + V_2 + V_3) \\ &= \dim V_1 + \dim V_2 + \dim V_3 - \dim(V_1 + V_2 + V_3)\,. \end{aligned}$$

□

Remark 2. For the sake of simplicity, the previous theorem considers a 3-round Feistel network with the same keyless S-box. However, since the result only relies on the structure of the three sets $\mathcal{X}_S(a_1, b_1)$, $\mathcal{Y}_S(a_2, b_2)$ and $\mathcal{X}_S(a_1 \oplus b_2, b_3)$, it clearly appears that Theorem 1 also holds for a Feistel network with three different S-boxes, S_1, S_2 and S_3, as soon as $\mathcal{X}_{S_1}(a_1, b_1)$, $\mathcal{Y}_{S_2}(a_2, b_2)$ and $\mathcal{X}_{S_3}(a_1 \oplus b_2, b_3)$ are affine subspaces.

As a direct consequence of Theorem 1, we get the following corollary.

Corollary 1. *Let S be a permutation of $\mathbb{F}_2^n$, and let a_1, b_1, a_2, b_2, b_3 be five elements in $\mathbb{F}_2^n$. Assume that there exist $\alpha_1, \alpha_2, \alpha_3 \in \mathbb{F}_2^n$ and three linear subspaces $V_1, V_2, V_3 \subseteq \mathbb{F}_2^n$ such that*

$$\mathcal{X}_S(a_1, b_1) = \alpha_1 + V_1,\ \mathcal{Y}_S(a_2, b_2) = \alpha_2 + V_2,\ \text{and } \mathcal{X}_S(a_1 \oplus b_2, b_3) = \alpha_3 + V_3\,.$$

Let

$$\begin{aligned} p_{\text{exact}} \quad = \quad & \Pr_{x_1, x_2 \in \mathbb{F}_2^n}\,[S(S(x_2) \oplus x_1 \oplus a_1 \oplus b_2) \oplus S(S(x_2) \oplus x_1) = b_3 \text{ and} \\ & S(x_2 \oplus a_2) \oplus S(x_2) = b_2 \text{ and } S(x_1 \oplus a_1) \oplus S(x_1) = b_1] \end{aligned}$$

and

$$\begin{aligned} p_{\text{ind}} \quad = \quad & \Pr_{x_1 \in \mathbb{F}_2^n}[S(x_1 \oplus a_1) \oplus S(x_1) = b_1] \times \Pr_{x_2 \in \mathbb{F}_2^n}[S(x_2 \oplus a_2) \oplus S(x_2) = b_2] \\ & \times \Pr_{z \in \mathbb{F}_2^n}[S(z \oplus a_1 \oplus b_2) \oplus S(z) = b_3] \end{aligned}$$

Then, either $p_{\text{exact}} = 0$ or

$$p_{\text{exact}} = 2^\ell p_{\text{ind}} \text{ with } \ell = n - \dim(V_1 + V_2 + V_3)\,.$$

Most notably, $0 \leq \ell \leq n - 2$.

Proof. Let us focus on the case where $p_{\text{exact}} \neq 0$. We deduce from Theorem 1 that

$$p_{\text{exact}} = 2^{\dim V_1 + \dim V_2 + \dim V_3 - \dim(V_1+V_2+V_3) - 2n}\,.$$

Since $p_{\text{ind}} = 2^{\dim V_1 + \dim V_2 + \dim V_3 - 3n}$, we obtain that

$$\lambda = \frac{p_{\text{exact}}}{p_{\text{ind}}} = 2^{\ell}$$

with

$$\begin{aligned}\ell &= \dim V_1 + \dim V_2 + \dim V_3 - \dim(V_1 + V_2 + V_3) - 2n - (\dim V_1 + \dim V_2 + \dim V_3 - 3n)\\ &= n - \dim(V_1 + V_2 + V_3)\,.\end{aligned}$$

Since $V_1 + V_2 + V_3$ is a subspace of $\mathbb{F}_2^n$, its dimension does not exceed n. On the other hand, when $p_{\text{ind}} \neq 0$, V_1 (resp. V_2) contains at least two elements, 0 and a_1 (resp. 0 and b_2). It follows that, if $a_1 \neq b_2$, then $V_1 + V_2$ contains the linear space spanned by a_1 and b_2, i.e. $\langle a_1, b_2\rangle$, which has dimension 2, implying that $\dim(V_1 + V_2 + V_3) \geq 2$. This lower bound also holds when $a_1 = b_2$ since this corresponds to $V_3 = \mathbb{F}_2^n$, leading to $\dim(V_1 + V_2 + V_3) = n$. Therefore, we have proved that

$$2 \leq \dim(V_1 + V_2 + V_3) \leq n$$

implying

$$0 \leq \ell \leq n - 2\,.$$

□

The hypothesis required for applying by this result, i.e., the fact that the three sets $\mathcal{X}_S(a_1, b_1)$, $\mathcal{Y}_S(a_2, b_2)$ and $\mathcal{X}_S(a_1 \oplus b_2, b_3)$ are affine subspaces, is satisfied in many practical cases. Indeed, when an S-box σ has differential uniformity at most 4, i.e., when 4 is the maximal value in the difference distribution table of σ, all sets $\mathcal{X}_\sigma(a, b)$ and $\mathcal{Y}_\sigma(a, b)$ are affine subspaces (see e.g., Lemma 2 in [DR07]). Therefore, the hypothesis is satisfied when S has an SPN structure based on a smaller differentially 4-uniform S-box σ: in this case, $\mathcal{X}_S(a, b)$ (resp. $\mathcal{Y}_S(a, b)$) corresponds to the Cartesian product of sets of the form $\mathcal{X}_\sigma(a, b)$ (resp. $\mathcal{Y}_\sigma(a, b)$).

An interesting observation deduced from the previous corollary is that, in all the previously mentioned situations, if the exact probability of a 3-round differential characteristic is non-zero, then it is greater than or equal to the usual estimate p_{ind}.

2.2 When S is differentially 4-uniform

There is a specific case where the factor λ between the two probabilities can be easily lower-bounded: when S itself is a function with differential uniformity at most 4.

Theorem 2. *Let S be a permutation of $\mathbb{F}_2^n$ with differential uniformity at most 4. Let a_1, b_1, a_2, b_2, b_3 be five nonzero elements in $\mathbb{F}_2^n$. Let*

$$\begin{aligned}p_{\text{exact}} \;=\; &\Pr_{x_1, x_2 \in \mathbb{F}_2^n}\left[S(S(x_2) \oplus x_1 \oplus a_1 \oplus b_2) \oplus S(S(x_2) \oplus x_1) = b_3 \text{ and}\right.\\ &\left. S(x_2 \oplus a_2) \oplus S(x_2) = b_2 \text{ and } S(x_1 \oplus a_1) \oplus S(x_1) = b_1\right]\end{aligned}$$

and

$$\begin{aligned}p_{\text{ind}} \;=\; &\Pr_{x_1 \in \mathbb{F}_2^n}[S(x_1 \oplus a_1) \oplus S(x_1) = b_1] \times \Pr_{x_2 \in \mathbb{F}_2^n}[S(x_2 \oplus a_2) \oplus S(x_2) = b_2]\\ &\times \Pr_{z \in \mathbb{F}_2^n}[S(z \oplus a_1 \oplus b_2) \oplus S(z) = b_3]\end{aligned}$$

Then,

- *if* $a_1 = 0$ *or* $a_2 = 0$ *or* $a_1 = b_2$*, then*

$$p_{\text{exact}} = p_{\text{ind}} \; ;$$

- *if the three S-boxes are active, i.e.* $a_1 \neq 0$ *and* $a_2 \neq 0$ *or* $a_1 \neq b_2$*, then either* $p_{\text{exact}} = 0$ *or*

$$p_{\text{exact}} = 2^{\ell} p_{\text{ind}} \text{ with } \max(0, n-6) \leq \ell \leq n-2 \, .$$

Moreover, if all three differentials (a_1, b_1)*,* (a_2, b_2)*, and* $(a_1 \oplus b_2, b_3)$ *have probability* 2^{1-n}*, then* $\lambda \in \{0, 2^{n-2}\}$*.*

Proof. We know from Corollary 1 that $p_{\text{exact}} = 0$ or $p_{\text{exact}} = 2^{\ell} p_{\text{ind}}$ with $\ell = n - \dim(V_1 + V_2 + V_3)$. Since $V_1 + V_2 + V_3$ is a subspace of $\mathbb{F}_2^n$, its dimension does not exceed n and is also smaller than the sum of the dimensions of the three subspaces. Since the S-box has differential uniformity at most 4, all V_i have dimension at most 2 unless the corresponding S-box is inactive, which is equivalent to $V_i = \mathbb{F}_2^n$.

- Let us first assume that the input difference of one of the S-boxes is zero. If the corresponding output difference is nonzero, the transition is not valid. In this case, we have $p_{\text{exact}} = p_{\text{ind}} = 0$. If the corresponding output is zero, i.e. if the S-box is inactive, the associated linear space V_i equals the whole space. It follows that $\ell = n - \dim(V_1 + V_2 + V_3) = 0$, leading to $p_{\text{exact}} = p_{\text{ind}}$.

- Let us now assume that all the three S-boxes are active. Then, $\dim(V_1 + V_2 + V_3)$ is smaller than 6. We derive that

$$\max(0, n-6) \leq \ell \leq n-2 \, .$$

Moreover, when all three subspaces V_1, V_2, and V_3 have dimension 1, then

$$V_1 + V_2 + V_3 = \langle a_1, b_2 \rangle \, .$$

It follows that, in this case,

$$\lambda \in \{0, 2^{n-2}\} \, .$$

In other words,

$$p_{\text{exact}} \in \{0, 2^{-2n+1}\} \, .$$

□

Most notably, when $n > 6$, if the differential path contains three active S-boxes, then its exact probability can never be equal to the product of the probabilities of the three constituent transitions.

Example 1. Theorem 2 can be verified for instance when S is the AES S-box, which operates on $\mathbb{F}_2^8$. Most differentials for the AES S-box have probability 2^{-7}. For such differential paths, we can check that $p_{\text{exact}} \in \{0, 2^{-15}\}$. For instance, for $(a_1, b_1) = (\texttt{0x01}, \texttt{0xca})$, $(a_2, b_2) = (\texttt{0xe5}, \texttt{0x18})$, and $b_3 = \texttt{0xb3}$, there are exactly two pairs $(x_1, x_2) \in \mathcal{X}_S(a_1, b_1) \times \mathcal{X}_S(a_2, b_2)$ such that $(S(x_2) \oplus x_1)$ satisfies the differential $(a_1 \oplus b_2, b_3)$. Then, the probability of the whole differential path is 2^{-15} while all three differentials have probability 2^{-7}, i.e., $\lambda = 2^{-15+21} = 2^6$. This factor varies when some of the involved differentials have probability 2^{-6}. For $(a_1, b_1) = (\texttt{0x01}, \texttt{0x1f})$, $(a_2, b_2) = (\texttt{0x33}, \texttt{0x0f})$ and $b_3 = \texttt{0xb8}$, the probability of the whole differential path is again 2^{-15}, while the second differential has probability 2^{-7} and the other two have probability 2^{-6}. We then have $\lambda = 2^{-15+19} = 2^4$.

If all S-boxes are active, the highest possible value for p_{exact} is $2^{n-2} \times \left(2^{-(n-2)}\right)^3 = 2^{-2n+4}$. It is worth noticing that this also corresponds to the highest possible value for p_{exact} when only two S-boxes are active, i.e. $p_{\text{exact}} = \left(2^{-(n-2)}\right)^2 = 2^{-2n+4}$. We now give a simple necessary condition on a_1 and b_2 for obtaining differential paths with three active S-boxes achieving this maximal probability.

Proposition 1. *Let S be a permutation of $\mathbb{F}_2^n$ with differential uniformity exactly 4. If there exist nonzero $a_1, b_1, a_1, b_2, b_3 \in \mathbb{F}_2^n$ such that $p_{exact} = 2^{-2n+4}$, then there exist x and y in $\mathbb{F}_2^n$ such that the second-order derivatives of S and S^{-1} satisfy*

$$D_{a_1} D_{b_2} S(x) = 0 \text{ and } D_{a_1} D_{b_2} S^{-1}(y) = 0\,, \tag{2}$$

where $D_u D_v S(x) = S(x) \oplus S(x \oplus u) \oplus S(x \oplus v) \oplus S(x \oplus u \oplus v)$.

It is worth noticing that, if S is an involution, then there always exists a pair (a_1, b_2) such that Condition (2) holds for some x and y in $\mathbb{F}_2^n$.

Proof. By hypothesis, all the three S-boxes are active. Then, $p_{\text{ind}} \leq 2^{-3n+6}$ and we know from Theorem 2 that $\lambda \leq 2^{n-2}$. It follows that $p_{\text{exact}} = 2^{-2n+4}$ if and only if $\lambda = 2^{n-2}$ (i.e., if $\dim(V_1 + V_2 + V_3) = 2$) and all the three involved differentials have probability $2^{-(n-2)}$. Since the differential (a_1, b_1) has probability $2^{-(n-2)}$, there exists $x, v_1 \in \mathbb{F}_2^n$ with $v_1 \neq \{0, a_1\}$ such that $\mathcal{X}_S(a_1, b_1) = x + \langle a_1, v_1 \rangle$. This implies that

$$S(x) \oplus S(x \oplus a_1) = b_1 = S(x \oplus v_1) \oplus S(x \oplus v_1 \oplus a_1)$$

leading to

$$D_{a_1} D_{v_1} S(x) = 0\,.$$

Similarly, a_2 is such that $\mathcal{Y}_S(a_2, b_2) = y + \langle b_2, v_2 \rangle$ for some $y, v_2 \in \mathbb{F}_2^n$ with $v_2 \notin \{0, b_2\}$. We now use the fact that, for any permutation S, $\mathcal{Y}_S(a, b) = \mathcal{X}_{S^{-1}}(b, a)$ (see Remark 1). From the same arguments as for v_1, we deduce that

$$D_{b_2} D_{v_2} S^{-1}(y) = 0\,.$$

But, since $\lambda = 2^{n-2}$, we know that

$$\dim(V_1 + V_2) = \dim\langle a_1, b_2, v_1, v_2 \rangle = 2\,.$$

It follows that $v_1 \in \{b_2, b_2 \oplus a_1\}$ and $v_2 \in \{a_1, a_1 \oplus b_2\}$. This implies that $D_{a_1} D_{b_2} S(x) = 0$ and $D_{a_1} D_{b_2} S^{-1}(y) = 0$.

It is well-known that there is no pair of nonzero distinct elements (a, b) such that $D_a D_b S$ takes the value 0 if and only if S is APN (i.e., its differential uniformity equals 2) [Nyb94]. In our case, S is not APN, implying that such a pair (a, b) exists. When S is an involution, is also satisfies $D_a D_b S^{-1}(y) = 0$ for some y. □

Example 2 (RoadRunneR S-box)**.** It is easy to check that, for the RoadRunneR [BS15] S-box, there is no pair of nonzero distinct elements (a_1, b_2) such that both $D_{a_1} D_{b_2} S$ and $D_{a_1} D_{b_2} S^{-1}$ vanish at some points. We then deduce that any differential path with three active S-boxes satisfies $p_{\text{exact}} \leq 2^{-5}$. By examining all second-order derivatives of this S-box which take the value 0, we have searched for all $(a_1, b_1, a_2, b_2, b_3)$ such that all three differentials have probability 2^{-2} and lead to a differential path with overall probability 2^{-5}. We have found 136 such configurations. One example is

$$a_1 = \texttt{0x1}, b_1 = \texttt{0x1}, a_2 = \texttt{0x8}, b_2 = \texttt{0x4}, b_3 = \texttt{0x8}\,.$$

Among these patterns, the only one which satisfies $a_2 = a_1 \oplus b_2$ and such that also the differentials (a_1, b_2) and (a_1, b_3) have probability 2^{-2} is the one we will use in the next section:

$$a_1 = \texttt{0xd}, a_2 = \texttt{0xc} \text{ and } b_1 = b_2 = b_3 = \texttt{0x1}\,,$$

and the configuration obtained by inverting the roles of a_1 and a_2.

Example 3 (Klein S-box)**.** The Klein [GNL11] S-box is an involution over $\mathbb{F}_2^4$. Then, there exist some pairs of nonzero distinct elements (a_1, b_2) such that both $D_{a_1}D_{b_2}S$ and $D_{a_1}D_{b_2}S^{-1}$ vanish at some points. For instance, $a_1 = \texttt{0x1}$ and $b_2 = \texttt{0x2}$ satisfy this property. For this S-box, the differential path defined by

$$a_1 = \texttt{0x1}, b_1 = \texttt{0x3}, a_2 = \texttt{0xd}, b_2 = \texttt{0x2}, \text{ and } b_3 = \texttt{0xe}$$

has overall probability 2^{-4}. In other words, any pair of elements (x_1, x_2) satisfying the first two differentials also leads to some $(S(x_2) \oplus x_1)$ which satisfies the third one.

All previous results hold in the keyless setting, but are still valid when the three S-boxes are distinct permutations with differential uniformity 4. This enables us to cover the fixed-key scenario since using S with a fixed round-key k is equivalent to using $S' : x \mapsto S_k(x)$. For instance, in the fixed-key scenario, Theorem 2 states that a differential path with three active S-boxes satisfies $p_{\text{exact}} = \lambda p_{\text{ind}}$ with $\lambda \in \{0, 2^\ell, \text{ with } \max(0, n-6) \leq \ell \leq n-2\}$. However, for a given differential path, the value of λ may vary with the key. For instance, the same differential path may have probability zero for some round-keys, and probability $p_{\text{exact}} > 0$ for the other ones.

3 Application to RoadRunneR

3.1 Description of RoadRunneR

RoadRunneR is a lightweight block cipher recently proposed by Baysal and Sahin [BS15]. It has a Feistel network structure with a 64-bit block size and it supports both 80 and 128-bit keys. In the 80-bit version, the number of rounds is 10, whereas in the 128-bit version the number of rounds is 12. Whitening keys (WK_0 and WK_1) are applied to the left half of the block in the first and last round. The general structure of RoadRunneR is depicted in Figure 3.

Round Function. RoadRunneR's round function, named F, takes as input a 32-bit block L_i, a 96-bit subkey K_i, and a 32-bit constant C_i. The constant C_i for round i is the 32-bit value $N_r - i$, where N_r is the total number of rounds of the cipher as defined above.

The round function in RoadRunneR consists of three subsequent applications of SLK, which is composed of a substitution layer followed by a linear layer and a key addition layer. After three SLK layers a single substitution layer (S) is performed. In between the second and third SLK layer the constant C_i is added (cf. Figure 3).

Key Schedule. The key expansion of the 128-bit RoadRunneR version chops the key up in four 32-bit words. The round keys are permutations of these words. Similarly, in the 80-bit version the key is split into five 16-bit words, and the key schedule is a permutation of six words. Table 2 lists the exact permutations for the round and whitening keys.

Substitution Layer. The substitution layer S consists of a parallel composition of the 4×4-bit S-box of Table 3[1] to every 4-bit nibble of the block.

Linear Layer. The linear layer L applies the function $L' : \mathbb{F}_2^8 \mapsto \mathbb{F}_2^8$ to each individual byte of the block

$$L'(x) = x \oplus (x \lll 1) \oplus (x \lll 2).$$

This construction is known to be invertible in general for distinct rotation offsets [Riv11], and the designers of RoadRunneR argue that this particular set of rotation offsets has good diffusion properties.

[1] This is the "optimal" S-box 13 in [UCI+11, Table 4].

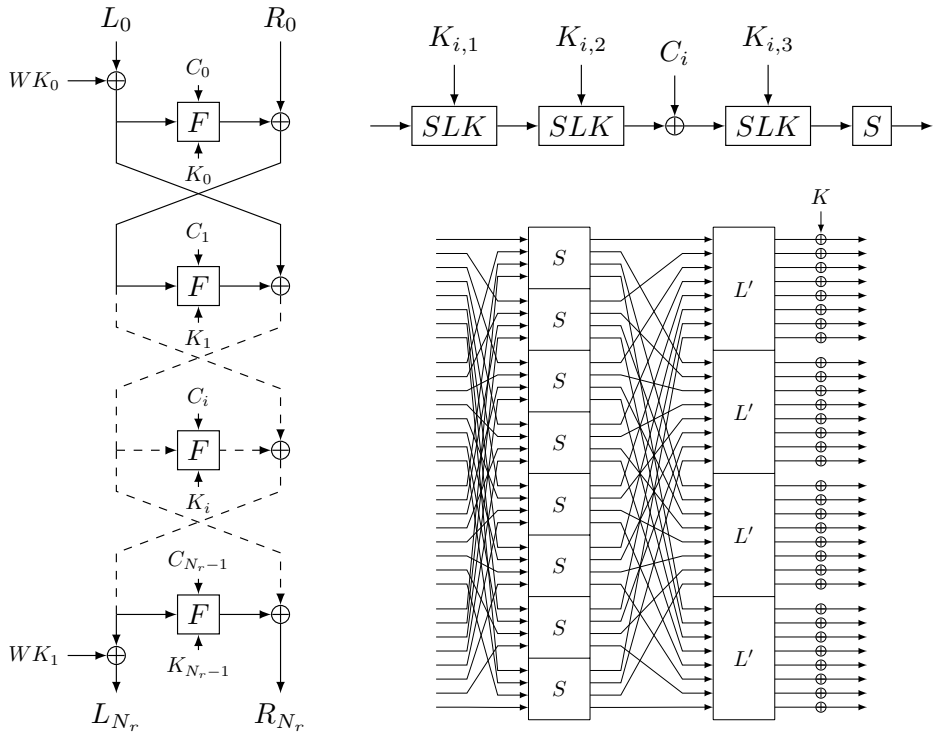

Figure 3: Overview of the RoadRunneR block cipher. Left: Feistel network with whitening keys xored in the first and last round. Top right: The round function F, taking in as input a 32-bit word, a 32-bit constant and a 96-bit round key. Bottom right: The core SLK function, which consists of an S-box layer followed by a linear diffusion layer and finally a key addition.

3.2 Security Analysis by the Designers

The designers claim no security in the related-key setting, due to the fact that the key schedule uses the master key without any change in between rounds. The designers in fact mention in the paper that each F can be passed with only two active S-boxes in a related key attack, with total of 24 active S-boxes, and that this total number may be further reduced in a more detailed analysis. We stress that no information about concrete characteristics, such as plaintext and subkey difference is provided.

In the single-key setting, the designers show that the minimum number of active S-boxes in an active F is 10 along with concrete propagation patterns. The authors experimentally checked that the probability of characteristics and differentials is correct. In their experiments they report that, the differential probability does not significantly increase from the theoretically calculated characteristic probability. Based on this experiment, the authors assume that each active S-box multiplies the probability with 2^{-2} and an active F has approximately a probability of 2^{-20}.

3.3 Applications of our Observations

By comparing Figure 1 and Figure 3, it is easy to see that the analysis in Section 2 can directly be applied to RoadRunneR when the number of rounds is more than two. We

Table 2: RoadRunneR's key schedule.

(a) 128-bit key.

Round Number	Key schedule
WK_0	A
WK_1	B
0 (mod 4)	$B\|\|C\|\|D$
1 (mod 4)	$A\|\|B\|\|C$
2 (mod 4)	$D\|\|A\|\|B$
3 (mod 4)	$C\|\|D\|\|A$

(b) 80-bit key.

Round Number	Key schedule
WK_0	$A\|\|B$
WK_1	$C\|\|D$
0 (mod 5)	$C\|\|D\|\|E\|\|A\|\|B\|\|C$
1 (mod 5)	$D\|\|E\|\|A\|\|B\|\|C\|\|D$
2 (mod 5)	$E\|\|A\|\|B\|\|C\|\|D\|\|E$
3 (mod 5)	$A\|\|B\|\|C\|\|D\|\|E\|\|A$
4 (mod 5)	$B\|\|C\|\|D\|\|E\|\|A\|\|B$

Table 3: The RoadRunneR S-box.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	0	8	6	D	5	F	7	C	4	E	2	3	9	1	B	A

emphasize that the observations can be applied both in the single-key and related-key settings. We also notice that the observation does not contradict the experiments by the designers that verified the probability of differentials within one round. What we are showing is that even before calculating the effect of collecting multiple differences, the actual probability of characteristics p_{exact} is higher than theoretically calculated one, p_{ind}, under the independent S-box assumption when the number of rounds is more than two.

In the following sections, we demonstrate the power of our observations with applications to concrete attacks.

3.4 Attack on RoadRunneR-128

First, we concretize the characteristic having only two active S-boxes per round mentioned by the designers. Suppose that a 128-bit master key K is denoted by four 32-bit values and the difference of those values are denoted by $\Delta_0, \Delta_1, \Delta_2$ and Δ_3. By following the key schedule described in Table 2, the difference of the initial whitening key is $\Delta WK_0 = \Delta_0$. Then, subkey differences are $(\Delta_1, \Delta_2, \Delta_3)$ for the first round, $(\Delta_0, \Delta_1, \Delta_2)$ for the second round, $(\Delta_3, \Delta_0, \Delta_1)$ for the third round, and so on. Four rounds with those subkey differences are illustrated in Figure 4.

We then choose $\Delta_0, \Delta_1, \Delta_2$ and Δ_3. There are four S-layers in each round. Our strategy consists in canceling the difference from Δ_1 with Δ_2 after the S-layer, which makes the next S-layer inactive. Then canceling the difference from Δ_3 with Δ_0 after the S-layer, which makes the next S-layer inactive. By iterating this, non-active S-layers and active S-layers appear alternately, and we only have 2 active S-boxes per round.

As a result of our analysis, we construct a 4-round iterative characteristic by satisfying the following four conditions.

$$\Pr_{x\in\mathbb{F}_2^4}[S(x) \oplus S(x \oplus \delta_1) = \gamma_2] = 2^{-2}, \tag{3}$$

$$\Pr_{x\in\mathbb{F}_2^4}[S(x) \oplus S(x \oplus \delta_3) = \gamma_0] = 2^{-2}, \tag{4}$$

$$\Pr_{x\in\mathbb{F}_2^4}[S(x) \oplus S(x \oplus \delta_1) = \delta_1 \oplus \delta_3] = 2^{-2}, \tag{5}$$

$$\Pr_{x\in\mathbb{F}_2^4}[S(x) \oplus S(x \oplus \delta_3) = \delta_1 \oplus \delta_3] = 2^{-2}, \tag{6}$$

where δ_1 is a group of 4 bits in the 32-bit differences Δ_1 and the 4 bits gather into a single active S-box after the bit-permutation around the S-layer. δ_3 can similarly be defined.

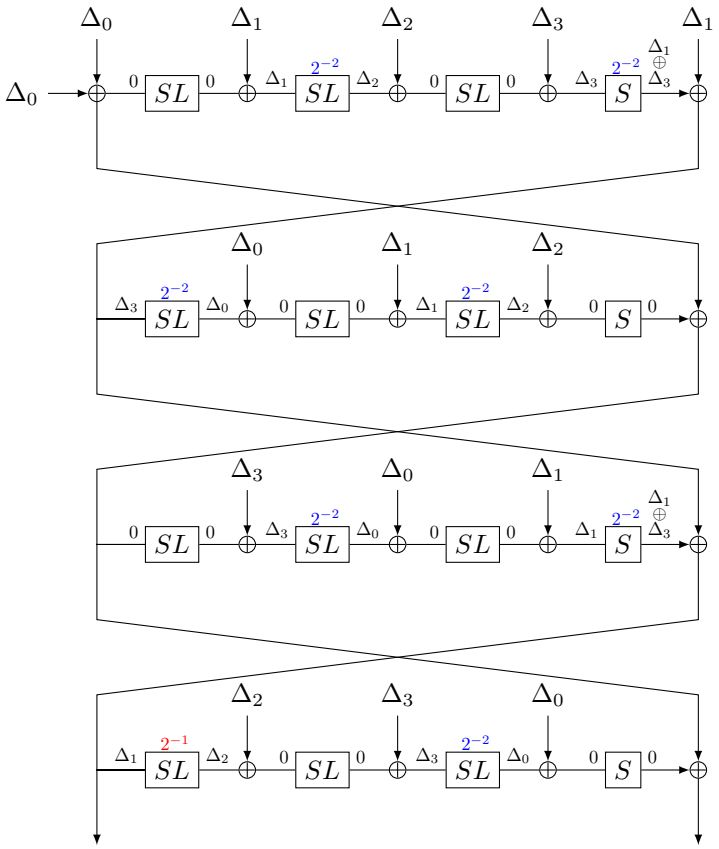

Figure 4: Four-round iterative differential characteristic against RoadRunneR-128.

The difference γ_0 (resp. γ_2) corresponds to the corresponding nibble of $L^{-1}(\Delta_0)$ (resp. of $L^{-1}(\Delta_2)$) where L denotes the whole linear layer. For example, when the active S-box position is fixed to the top in Figure 3, $\delta_1 = \texttt{0xf}$ corresponds to $\Delta_1 = \texttt{0x01010101}$.

We note that by setting $\Delta_0 = \Delta_2 = L(\Delta_1 \oplus \Delta_3)$, the first two conditions can always be satisfied when the last two conditions are satisfied. The characteristic is iterative after 4 rounds including subkey differences, and can be extended to 12 rounds easily.

By analyzing the differential distribution table (DDT) of the S-box, we chose $\delta_1 = \texttt{0xc}$ and $\delta_3 = \texttt{0xd}$ (or $\Delta_1 = \texttt{0x01010000}$ and $\Delta_3 = \texttt{0x01010001}$). Then, $\delta_1 \oplus \delta_3 = \texttt{0x1}$ ($\Delta_0 = \Delta_2 = L(\texttt{0x00000001})$). This configuration satisfies the above listed conditions.

Evaluation of p_{ind} and p_{exact}. From Eqs. (3) to (6), p_{ind} can be calculated from the transition probability for each S-box, 2^{-2}, and the number of active S-boxes, leading to $2^{-2 \cdot 24} = 2^{-48}$.

Recall that for any pair (a, b) of differences, we use the following notation: $\mathcal{X}_S(a, b) = \{x \in \mathbb{F}_2^4 : S(x) \oplus S(x \oplus a) = b\}$ and $\mathcal{Y}_S(a, b) = \{S(x) \in \mathbb{F}_2^4 : S(x) \oplus S(x \oplus a) = b\}$. By applying the analysis in Section 2, p_{exact} of the first S-layer in round 4 in Figure 4 is

$$\Pr_{x \in \mathcal{X}_S(\texttt{0xd},\texttt{0x1}), y \in \mathcal{Y}_S(\texttt{0xc},\texttt{0x1})}[x \oplus y \in \mathcal{X}_S(\texttt{0xc}, \texttt{0x1})]. \tag{7}$$

By analyzing the DDT, we obtain $\mathcal{X}_S(\texttt{0xd}, \texttt{0x1}) = \{\texttt{0x0}, \texttt{0x1}, \texttt{0xc}, \texttt{0xd}\}$, $\mathcal{Y}_S(\texttt{0xc}, \texttt{0x1}) = \{\texttt{0x4}, \texttt{0x5}, \texttt{0xe}, \texttt{0xf}\}$, and $\mathcal{X}_S(\texttt{0xc}, \texttt{0x1}) = \{\texttt{0x4}, \texttt{0x5}, \texttt{0x8}, \texttt{0x9}\}$, which leads to $p_{\text{exact}} = 2^{-1}$.

Similarly, p_{exact} of the first S-layer in rounds 6, 8, 10, and 12 are 2^{-1}, which leads to 2^{-43}.

It is important to notice that this probability is evaluated in the keyless scenario studied in the previous section because it is not affected by the values of the round-keys. Indeed, the round-key is inserted *after* applying the S-box and then does not affect $\mathcal{X}_S(\texttt{0xc}, \texttt{0x1})$ and $\mathcal{X}_S(\texttt{0xd}, \texttt{0x1})$. Moreover, the S-box involved in $\mathcal{Y}_S(\texttt{0xc}, \texttt{0x1})$ corresponds to the last S-box-layer in the third round and is independent from the key. It follows that, in this situation, p_{exact} takes the same value for any fixed-key.

Experiments. First of all, we experimentally proved that 24 active S-boxes in 12 rounds is minimal by using the SAT-solver based tool [MP13]. Differently from the expectation by the designers, the number of active S-boxes will not be further reduced.

We then implemented the attack up to 8 rounds. We refer back to Table 1 for the results, which clearly indicates the gap between p_{ind} and p_{exact} in rounds 4, 6 and 8.

3.5 Attack on RoadRunneR-80

In this part, we present an 8-round attack against RoadRunneR-80. Differently from RoadRunneR-128, the key is divided into 16-bit values (A, B, C, D, E) and each of them can be both the top half or the bottom half of 32-bit subkeys. Hence, constructing systematic subkeys is harder than in RoadRunneR-128.

By applying the bit-permutation around S, a group of 4 bits for a single S-box will move to symmetric positions in the 32-bit state. To exploit this fact, we set $\Delta A = \Delta B = \Delta C = \Delta D = \Delta E$ to make all 32-bit subkey differences identical and symmetric.

We set subkey difference to the xor of two differences ΔX and ΔZ. ΔX takes a role of input difference to the subsequent S-layer, and ΔZ cancels the difference from the previous S-layer. Namely, in every S-layer, cancellation and injection of differences are performed. The characteristic is illustrated in Figure 5, which is iterative after four rounds.

We then choose Δ_X and Δ_Z, where $\Delta_Z \triangleq L(\Delta_Y)$. We define δ_X, δ_Y similarly to the previous section, namely 4-bit difference in the 32-bit variable corresponding to an active S-box. Because subkey difference is symmetric, Δ_X and Δ_Y must be symmetric, which further limits δ_X, δ_Y to be symmetric (and non-zero). Therefore, $\delta_X, \delta_Y \in \{\texttt{5}, \texttt{a}, \texttt{f}\}$. According to the characteristic in Figure 5, we have the following two conditions;

$$\Pr_{x \in \mathbb{F}_2^4}[S(x) \oplus S(x \oplus \delta_X) = \delta_Y] > 0, \tag{8}$$

$$\Pr_{x \in \mathbb{F}_2^4}[S(x) \oplus S(x \oplus \delta_X \oplus \delta_Y) = \delta_Y] > 0, \tag{9}$$

$$\delta_X \neq \delta_Y. \tag{10}$$

From DDT, there is only one choice, $\delta_X = \texttt{5}$ ($\Delta_X = \texttt{0x00010001}$) and $\delta_Y = \texttt{a}$, which satisfies Conditions (8) and (9) with probability 2^{-2} and 2^{-3}, respectively.

Evaluation of p_{ind} and p_{exact}. We first evaluate p_{ind}. In every two rounds, there are seven active S-boxes with probability of 2^{-2} and there is one active S-box with probability of 2^{-3}. Thus p_{ind} is 2^{-17} in every 2 rounds and 2^{-68} for 8 rounds, which are unlikely to be satisfied with 2^{64} plaintexts of the full codebook.

The mechanism of occurring the advantage of p_{exact} is the same as in the attack against RoadRunneR-128, but we now have an active S-box at the beginning of the inner function in every round. Therefore, from the third round, p_{exact} is higher than p_{ind} by a factor of 2, which improves the probability of 8-rounds to $2^{-8-9-8-7-7-8-8-7} = 2^{-62}$.

In more details, p_{exact} of the first S-layer in rounds with $p_{\text{ind}} = 2^{-8}$ and $p_{\text{ind}} = 2^{-9}$ are

$$\Pr_{x \in \mathcal{X}_S(\texttt{0xf},\texttt{0xa}), y \in \mathcal{Y}_S(\texttt{0x5},\texttt{0xa})}[x \oplus y \in \mathcal{X}_S(\texttt{0x5}, \texttt{0xa})], \tag{11}$$

$$\Pr_{x \in \mathcal{X}_S(\texttt{0x5},\texttt{0xa}), y \in \mathcal{Y}_S(\texttt{0x5},\texttt{0xa})}[x \oplus y \in \mathcal{X}_S(\texttt{0xf}, \texttt{0xa})]. \tag{12}$$

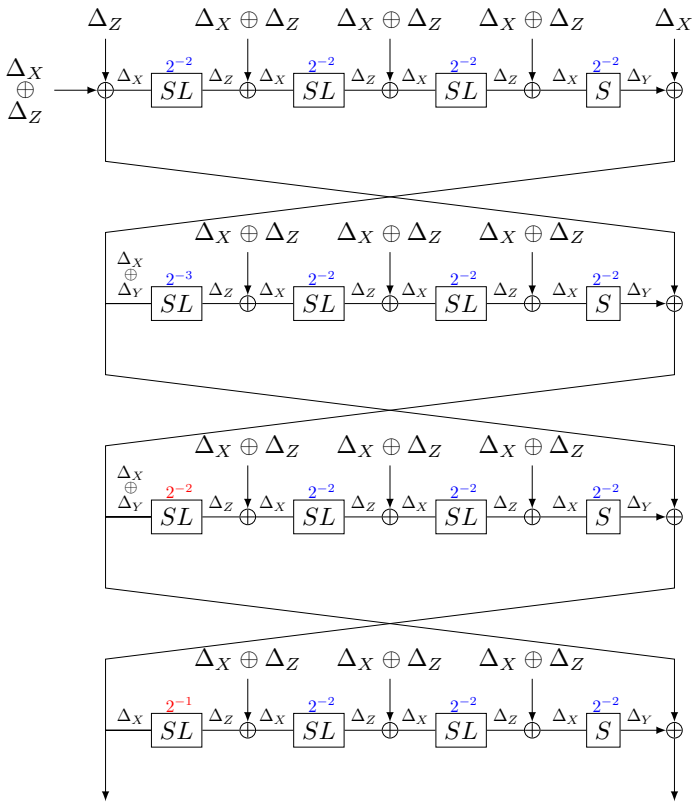

Figure 5: 4-round iterative characteristic for RoadRunneR-80. $\Delta_X = \texttt{0x5}$, $\Delta_Y = \texttt{0xA}$, $\Delta_Z = L(\Delta_Y)$. The transition probabilities in red are those which differ from the estimate with the independent S-box assumption.

Given that $\mathcal{X}_S(\texttt{0x5}, \texttt{0xa}) = \{\texttt{0x2}, \texttt{0x3}, \texttt{0x6}, \texttt{0x7}\}$, $\mathcal{Y}_S(\texttt{0x5}, \texttt{0xa}) = \{\texttt{0x6}, \texttt{0x7}, \texttt{0xc}, \texttt{0xd}\}$ and $\mathcal{X}_S(\texttt{0xf}, \texttt{0xa}) = \{\texttt{0x0}, \texttt{0xf}\}$, p_{exact} in eq. (11) is 2^{-1} instead of 2^{-2} and p_{exact} in eq. (12) is 2^{-2} instead of 2^{-3}.

Experiments. To ensure our estimates match reality, we performed some computational verification of the above differential characteristic:

- 1 round of RoadRunneR-80 yielded 65870 ($\approx 2^{16}$) matches over 2^{24} trials;
- 2 rounds of RoadRunneR-80 yielded 1011 ($\approx 2^{10}$) matches over 2^{27} trials;
- 3 rounds of RoadRunneR-80 yielded 124 ($\approx 2^{7}$) matches over 2^{32} trials;
- 4 rounds of RoadRunneR-80 yielded 28 ($\approx 2^{5}$) matches over 2^{37} trials;
- 5 rounds of RoadRunneR-80 yielded 16 ($= 2^{4}$) matches over 2^{43} trials.

These results are summarized in Table 1.

4 Extension to Almost-MDS Matrix in Minalpher-P

In this section, we show that improving the probability by evaluating p_{exact} can be extended to SPN with almost-MDS binary matrices. An example of such matrices is

$$\begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}, \tag{13}$$

which is actually adopted by Minalpher [STA+14]. The rotated version of the above matrix is more popular, which can be seen in several designs e.g. PRINCE [BCG+12], FIDES [BBK+13], and Midori [BBI+15]. Section 4.1 provides an overview of our observation. Section 4.2 introduces the specification of Minalpher-P. Section 4.3 introduces the previous best differential characteristic evaluated by p_{ind}. Section 4.4 improves the probability by evaluating p_{exact} and extends the attack by two rounds.

4.1 Overview

Let us consider a 1-column state consisting of four cells of size n bits, thus the state size is $4n$ bits. Suppose that the state is updated by an SPN, in which the S-layer applies an n-bit S-box to all of four cells and the P-layer applies the matrix in Eq. (13). With this structure, the number of active cells can be two per rounds owing to the following property: *When two cells have an identical difference, the matrix multiplication does not change the number of active cells and the differential value.*

Let us consider the 2-round characteristic shown in Figure 6, which assumes that $\Pr_{x\in\mathbb{F}_2^n}[S(x)\oplus S(x\oplus\Delta a)=\Delta b]=2^{-n+2}$ and $\Pr_{x\in\mathbb{F}_2^n}[S(x)\oplus S(x\oplus\Delta b)=\Delta c]=2^{-n+2}$. p_{ind} is $(2^{-n+2})^4$ because of the four active S-boxes, meanwhile we show that p_{exact} is

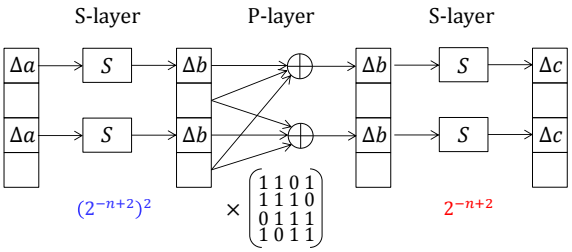

Figure 6: Overview: 2-round characteristic in SPN with single column.

$(2^{-n+2})^3$ in which the S-layer can be satisfied only with 2^{-n+2} from the second round. The state of SPN ciphers usually have more columns, thus the improvement by a factor of 2^{-n+2} can be amplified, which makes the improved factor significantly large.

4.2 Specification of Minalpher-P

The core part of Minalpher is the Even-Mansour construction in which a 256-bit plaintext is masked by a 256-bit secret value, and then a nibble-wise 256-bit permutation called Minalpher-P is computed. Finally, the output of Minalpher-P is masked by the 256-bit secret value. A 256-bit state is described as two 4×8 nibble-matrices denoted by A and B.

Let A_{i-1} and B_{i-1} be the inputs of the round function for round i. The states are updated to A_i and B_i with a round function, which consists of SubNibbles (SN), ShuffleRows (SR), SwapMatrices (SM), XorMatrix (XM) and MixColumns (MC), where

SN, SR and MC are functions from $\left\{\mathbb{F}_2^4\right\}^{4\times 8}$ to $\left\{\mathbb{F}_2^4\right\}^{4\times 8}$. In the end, the state is xored with the round constant. We use notations A^{op} and B^{op} to denote $\left\{\mathbb{F}_2^4\right\}^{4\times 8}$ data after operation op. See Figure 7 for its illustration.

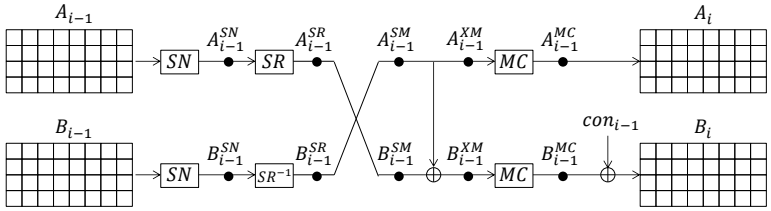

Figure 7: Illustration of the round function of Minalpher-P.

SubNibbles (SN). SN substitutes each nibble by using 4-bit involution S-box S.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	B	3	4	1	2	8	C	F	5	D	E	0	6	9	A	7

ShuffleRows (SR). SR shuffles nibble positions within each row. SR consists of two shuffle functions SR_1 and SR_2 defined as follows. Elements in 4×8 matrix A are moved according to the table below, and for B, SR^{-1} is applied instead of SR.

i	0	1	2	3	4	5	6	7
$SR_1(i)$	6	7	1	0	2	3	4	5
$SR_2(i)$	4	5	0	1	7	6	2	3
$SR_1^{-1}(i)$	3	2	4	5	6	7	0	1
$SR_2^{-1}(i)$	2	3	6	7	0	1	5	4

SwapMatrices (SM). SM swaps the matrix A and the matrix B.

XorMatrix (XM). The matrix B is xored with the matrix A.

MixColumns (MC). MC is a column-wise linear operation. As introduced before, MC is expressed as a multiplication by the matrix in Eq. (13).

Round Constant. The round constant con_{i-1} is xored to the matrix B. In this paper, the fact that the matrix A is not updated by round constant is important.

4.3 Differential Characteristics of Minalpher-P

The designers of Minalpher found a 6-round iterative truncated differential with 64 active S-boxes, which is shown in Figure 8. Note that this is not the one with minimal number of active S-boxes for 6 rounds. However, if it is iterated beyond 6 rounds, the number of active S-boxes matches the lower bound obtained by automated search.

Then, we convert the truncated differential to a specific characteristic by fixing the differential values. By calculating DDT of the 4-bit S-box, we observe that the input difference `0x4` will be mapped to the output difference `0x4` with probability 2^{-2}. So, we replace all filled cells in Figure 8 with the particular difference `0x4`.

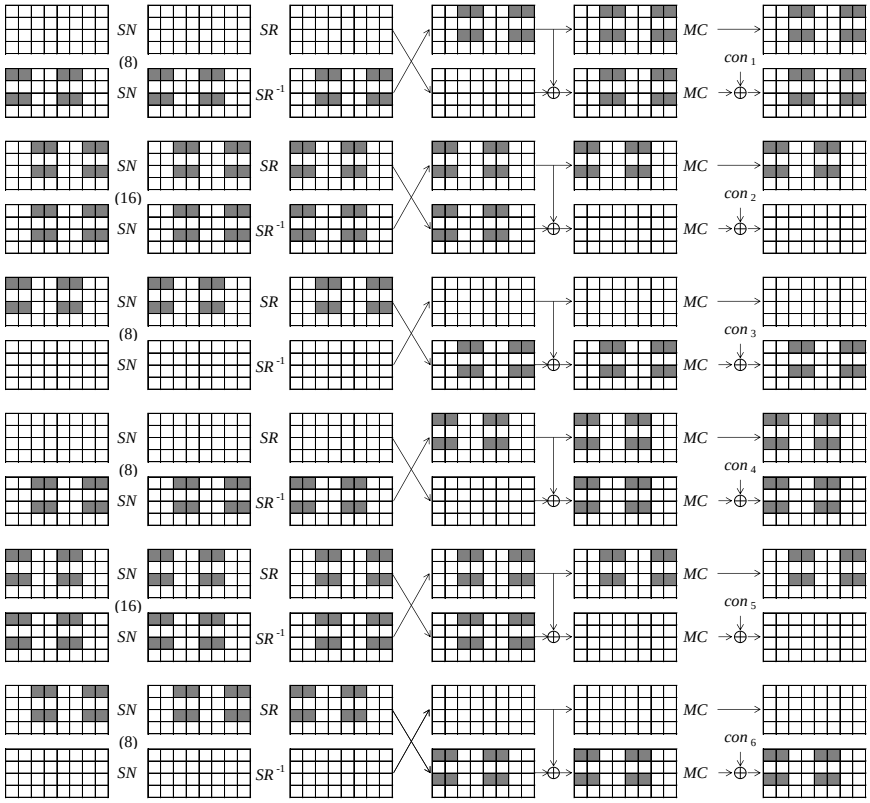

Figure 8: 6-round iterative truncated differential of Minalpher-P. Filled and empty cells denote active and inactive nibbles, respectively. Note that we rotated the original 6-round iterative characteristic by one round to optimize it in our analysis.

Let us evaluate the probability of the 6-round characteristic. Here we assume that the secret mask of the Even-Mansour construction prevents the attacker from choosing the plaintext or ciphertext to deterministically satisfy differential propagations through S-box in the first and the last rounds. The linear part is satisfied with probability 1, thus the probability only comes from the S-box, which is 2^{-2} per S-box. Because $8+16+8+8+16+8=64$ S-boxes are included in the characteristic, the probability is $(2^{-2})^{64}=2^{-128}$ when all transitions through all S-boxes are assumed to be independent. Considering that the security of Minalpher is claimed up to 128 bits, extending the characteristic by a few more rounds is impossible.

4.4 Analysis of Exact Probability

Preliminaries. Recall that for any pair (a,b) of differences, we use the following notation: $\mathcal{X}_S(a,b)=\{x\in\mathbb{F}_2^4 : S(x)\oplus S(x\oplus a)=b\}$ and $\mathcal{Y}_S(a,b)=\{S(x)\in\mathbb{F}_2^4 : S(x)\oplus S(x\oplus a)=b\}$. When S is involution as in Minalpher-P, $\mathcal{X}_S(a,a)$ is equal to $\mathcal{Y}_S(a,a)$ for any a. In particular, when $a=\mathtt{4}$ in the S-box of Minalpher-P, $\mathcal{X}_S(\mathtt{4},\mathtt{4})=\mathcal{Y}_S(\mathtt{4},\mathtt{4})=\{\mathtt{9},\mathtt{a},\mathtt{d},\mathtt{e}\}$. This is represented by an affine space $\langle\mathtt{3},\mathtt{4}\rangle+\mathtt{9}$, where $\langle x,y\rangle$ is a linear subspace.

Analysis of $p_{\mathbf{exact}}$. Here, we show that the probability of the 6-round characteristic is actually 2^{-96} instead of 2^{-128}, thus the number of attacked rounds can be extended. We

begin with the analysis of the simple case; SN and MC are iterated twice in a column, which is shown in Figure 9.

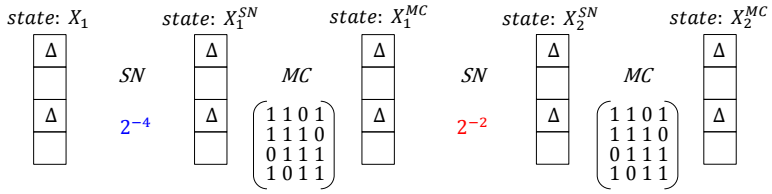

Figure 9: Analysis of simple case ($\Delta = \texttt{0x4}$). Probability is 2^{-8} if two SN operations are evaluated independently, while the exact probability is 2^{-6}.

As shown in Figure 9, the five states are denoted by $X_1, X_1^{SN}, X_1^{MC}, X_2^{SN}, X_2^{MC}$. Suppose that the 4-nibble value of X_1 is chosen uniformly at random. Then the probability of satisfying the first SN layer is $(2^{-2})^2 = 2^{-4}$. When this occurs, the value of $X_1^{SN}[0]$ and $X_1^{SN}[2]$ are limited to four choices in $\mathcal{Y}_S(4,4) = \{\texttt{9}, \texttt{a}, \texttt{d}, \texttt{e}\}$. From the specification of MC, the value of active nibbles in X_1^{MC} are calculated as

$$X_1^{MC}[0] = X_1^{SN}[0] \oplus X_1^{SN}[1] \oplus X_1^{SN}[3],$$
$$X_1^{MC}[2] = X_1^{SN}[1] \oplus X_1^{SN}[2] \oplus X_1^{SN}[3].$$

In order to satisfy the differential propagation in the second SN operation, both of $X_1^{MC}[0]$ and $X_1^{MC}[2]$ must be in the affine space of $\mathcal{X}_S(4,4) = \{\texttt{9}, \texttt{a}, \texttt{d}, \texttt{e}\}$. Considering that $X_1^{SN}[0]$ and $X_1^{SN}[2]$ are in the affine space, the condition that both of $X_1^{MC}[0]$ and $X_1^{MC}[2]$ are in the same affine space is $X_1^{SN}[1] \oplus X_1^{SN}[3]$ is in its linear subspace $\langle 3, 4\rangle = \{\texttt{0}, \texttt{3}, \texttt{4}, \texttt{7}\}$. This occurs with probability 2^{-2}, thus the probability of satisfying the second SN layer is 2^{-2}, instead of 2^{-4}.

Application to 6-Round Characteristic. All the differences in Figure 8 are fixed to `0x4`.

Round 1. Suppose that the lower half of the input state, B_0, is chosen uniformly at random. Then, the probability of satisfying the SN layer in round 1 is $(2^{-2})^8 = 2^{-16}$.

Round 2. The SR operation does not mix the value, thus irrelevant to this analysis. The state B_0^{SN} is next updated by MC and then passed to SN in round 2. Namely, the simple column-wise analysis discussed above appears in four columns. Thus the probability that the differences in A_1 are propagated to A_1^{SN} is $(2^{-2})^4 = 2^{-8}$ instead of 2^{-16}. Note that B_0^{XM} is xored with random state value B_0^{SM} and round constant, thus the probability between B_1 and B_1^{SN} is 2^{-16}. In total, the probability of round 2 is 2^{-24}.

Round 3. The same event as round 2 occurs. Namely B_1^{SN} is updated with MC and then SN in round 3. As discussed before, this probability is 2^{-8} instead of 2^{-16}.

Rounds 4–6. The probabilities for rounds 4, 5, and 6 are calculated round by round. The analysis becomes almost the same as round 1, 2, and 3, respectively because of the similarity of the active S-boxes positions. To avoid redundancy, we omit the round-by-round explanation. In the end, the probability for those rounds is $2^{-16-24-8} = 2^{-48}$.

From the above discussion we conclude that the probability of the 6-round differential characteristic in Figure 8 is 2^{-96}, which is significantly larger than p_{ind} of 2^{-128}.

Experimental Verification. The probability of the first three rounds already reach 2^{64}, which is infeasible in our environment. The gap between p_{ind} and p_{exact} first appears in state A_1^{SB} of the SN operation in the second round, which is independent of the propagation in state B_1^{SB}. We thus implement the state update from B_0^{SB} to A_1^{SB} with the limitation that values of active bytes are sampled randomly from $\mathcal{Y}_S(4,4)$.

We generated $65,536 (= 2^{16})$ random values at B_0^{SB}, and $250 (\approx 2^8)$ values satisfy the difference in A_1^{SB}, which confirms that the probability of the characteristic from B_0^{SB} to A_1^{SN} is actually $(2^{-2})^4 = 2^{-8}$ instead of $(2^{-4})^4 = 2^{-16}$.

Extension to 8 Rounds. We append 1 round to both of the beginning and the end of the 6-round iterative characteristic in Figure 8. Remember that the probability of the first round in the 6-round characteristic is 2^{-16}. Due to the iterative structure, with the same reason, the probability of the last extended round is 2^{-16}. The extended round at the beginning has eight active S-boxes. Because the advantage of p_{exact} cannot be exploited at the beginning, the probability is $(2^{-2})^8 = 2^{-16}$.

To conclude, the probability of the 8-round characteristic is $2^{-96-16-16} = 2^{-128}$. Considering that the previous 6-round characteristic has the same probability, we improved the previous attack by 2 rounds.

Note that a path with probability 2^{-128} cannot be a straightforward distinguisher with 2^{128} queries. Here our main focus is improving the previous analysis, and using the path with probability 2^{-128} is the same setting as the designers of Minalpher. Moreover, by combining with similar paths, the probability may be amplified to be greater than 2^{-128}.

5 Concluding Remarks

This paper studied the interaction between the differential transitions occurring in the multiple rounds of a fixed-key or unkeyed primitive. We showed that assuming independent input values for each S-box does not correspond to the actual situation, and p_{exact} can be much larger than p_{ind}. Our general analysis on the Feistel network showed that the gap between p_{exact} and p_{ind} depends on the S-box size and the S-box choice. In addition, having non-zero gap is inevitable when the S-box has differential uniformity 4 and a size larger than six bits (unless one Sbox is inactive).

This observation actually impacts the security of practical algorithms. We applied it to the lightweight block cipher RoadRunneRand the authenticated encryption scheme Minalpher. The results showed that with p_{exact} the number of attacked rounds could be improved compared to the evaluation with p_{ind}.

Symmetric-key primitives with unkeyed functions or public permutations are getting more popular due to its lightweight property and can be seen in many contemporary structures such as the sponge and the Even-Mansour constructions. This paper alerts us that the resistance against differential cryptanalysis needs to be analyzed carefully.

Acknowledgments

This work has been initiated during the Lorentz Center workshop on “High-Security Lightweight Cryptography”, held in Leiden, the Netherlands, in October 2016 and we would like to thank the organizers for inviting us. We also thank the anonymous reviewers for their careful reading and their valuable comments.

References

[BBI+15] Subhadeep Banik, Andrey Bogdanov, Takanori Isobe, Kyoji Shibutani, Harunaga Hiwatari, Toru Akishita, and Francesco Regazzoni. Midori: A block cipher for low energy. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 411–436. Springer, 2015.

[BBK+13] Begül Bilgin, Andrey Bogdanov, Miroslav Knezevic, Florian Mendel, and Qingju Wang. FIDES: Lightweight authenticated cipher with side-channel resistance for constrained hardware. In Guido Bertoni and Jean-Sébastien Coron, editors, *Cryptographic Hardware and Embedded Systems - CHES 2013 - 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings*, volume 8086 of *Lecture Notes in Computer Science*, pages 142–158. Springer, 2013.

[BBL13] Céline Blondeau, Andrey Bogdanov, and Gregor Leander. Bounds in shallows and in miseries. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013*, volume 8042 of *Lecture Notes in Computer Science*, pages 204–221. Springer, 2013.

[BCG+12] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçin. PRINCE - A low-latency block cipher for pervasive computing applications - extended abstract. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 208–225. Springer, 2012.

[BG10] Céline Blondeau and Benoît Gérard. Links between theoretical and effective differential probabilities: Experiments on PRESENT. Cryptology ePrint Archive, Report 2010/261, 2010. http://eprint.iacr.org/2010/261.

[BJK+16] Christof Beierle, Jérémy Jean, Stefan Kölbl, Gregor Leander, Amir Moradi, Thomas Peyrin, Yu Sasaki, Pascal Sasdrich, and Siang Meng Sim. The SKINNY family of block ciphers and its low-latency variant MANTIS. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 123–153. Springer, 2016.

[BS90] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. In Alfred Menezes and Scott A. Vanstone, editors, *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer, 1990.

[BS93] Eli Biham and Adi Shamir. *Differential Cryptanalysis of the Data Encryption Standard*. Springer, 1993.

[BS15] Adnan Baysal and Sühap Sahin. RoadRunneR: A small and fast bitslice block cipher for low cost 8-bit processors. In Tim Güneysu, Gregor Leander, and

Amir Moradi, editors, *Lightweight Cryptography for Security and Privacy - 4th International Workshop, LightSec 2015, Bochum, Germany, September 10-11, 2015, Revised Selected Papers*, volume 9542 of *Lecture Notes in Computer Science*, pages 58–76. Springer, 2015.

[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.

[CDK09] Christophe De Cannière, Orr Dunkelman, and Miroslav Knezevic. 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, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, volume 5747 of *Lecture Notes in Computer Science*, pages 272–288. Springer, 2009.

[CDL15] Anne Canteaut, Sébastien Duval, and Gaëtan Leurent. Construction of lightweight S-Boxes using Feistel and MISTY structures. In Orr Dunkelman and Liam Keliher, editors, *Selected Areas in Cryptography - SAC 2015 - 22nd International Conference, Sackville, NB, Canada, August 12-14, 2015, Revised Selected Papers*, volume 9566 of *Lecture Notes in Computer Science*, pages 373–393. Springer, 2015.

[CR15] Anne Canteaut and Joëlle Roué. On the behaviors of affine equivalent Sboxes regarding differential and linear attacks. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 45–74. Springer, 2015.

[DDGS15] Itai Dinur, Orr Dunkelman, Masha Gutman, and Adi Shamir. Improved top-down techniques in differential cryptanalysis. In Kristin E. Lauter and Francisco Rodríguez-Henríquez, editors, *Progress in Cryptology - LATINCRYPT 2015 - 4th International Conference on Cryptology and Information Security in Latin America, Guadalajara, Mexico, August 23-26, 2015, Proceedings*, volume 9230 of *Lecture Notes in Computer Science*, pages 139–156. Springer, 2015.

[DKS12] Orr Dunkelman, Nathan Keller, and Adi Shamir. Minimalism in cryptography: The Even-Mansour scheme revisited. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 336–354. Springer, 2012.

[DR06] Joan Daemen and Vincent Rijmen. Understanding two-round differentials in AES. In Roberto De Prisco and Moti Yung, editors, *Security and Cryptography for Networks, SCN 2006*, volume 4116 of *Lecture Notes in Computer Science*, pages 78–94. Springer, 2006.

[DR07] Joan Daemen and Vincent Rijmen. Plateau characteristics. *IET Information Security*, 1(1):11–17, 2007.

[EM91] Shimon Even and Yishay Mansour. A construction of a cipher from a single pseudorandom permutation. In Hideki Imai, Ronald L. Rivest, and Tsutomu Matsumoto, editors, *Advances in Cryptology - ASIACRYPT '91, International Conference on the Theory and Applications of Cryptology, Fujiyoshida, Japan,*

November 11-14, 1991, Proceedings, volume 739 of *Lecture Notes in Computer Science*, pages 210–224. Springer, 1991.

[EM97] Shimon Even and Yishay Mansour. A construction of a cipher from a single pseudorandom permutation. *J. Cryptology*, 10(3):151–162, 1997.

[GNL11] Zheng Gong, Svetla Nikova, and Yee Wei Law. KLEIN: A new family of lightweight block ciphers. In Ari Juels and Christof Paar, editors, *RFID. Security and Privacy - 7th International Workshop, RFIDSec 2011, Amherst, USA, June 26-28, 2011, Revised Selected Papers*, volume 7055 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2011.

[GPPR11] Jian Guo, Thomas Peyrin, Axel Poschmann, and Matthew J. B. Robshaw. The LED block cipher. In Bart Preneel and Tsuyoshi Takagi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2011*, volume 6917 of *Lecture Notes in Computer Science*, pages 326–341. Springer, 2011.

[HLL+00] Seokhie Hong, Sangjin Lee, Jongin Lim, Jaechul Sung, Dong Hyeon Cheon, and Inho Cho. Provable security against differential and linear cryptanalysis for the SPN structure. In Bruce Schneier, editor, *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings*, volume 1978 of *Lecture Notes in Computer Science*, pages 273–283. Springer, 2000.

[KMT01] Liam Keliher, Henk Meijer, and Stafford E. Tavares. New method for upper bounding the maximum average linear hull probability for SPNs. In Birgit Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*, volume 2045 of *Lecture Notes in Computer Science*, pages 420–436. Springer, 2001.

[KS07] Liam Keliher and Jiayuan Sui. Exact maximum expected differential and linear probability for two-round Advanced Encryption Standard. *IET Information Security*, 1(2):53–57, 2007.

[LMM91] Xuejia Lai, James L. Massey, and Sean Murphy. Markov ciphers and differential cryptanalysis. In Donald W. Davies, editor, *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings*, volume 547 of *Lecture Notes in Computer Science*, pages 17–38. Springer, 1991.

[LW14] Yongqiang Li and Mingsheng Wang. Constructing S-boxes for lightweight cryptography with Feistel structure. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, volume 8731 of *Lecture Notes in Computer Science*, pages 127–146. Springer, 2014.

[Mat96] Mitsuru Matsui. New structure of block ciphers with provable security against differential and linear cryptanalysis. In Dieter Gollmann, editor, *Fast Software Encryption, Third International Workshop, Cambridge, UK, February 21-23, 1996, Proceedings*, volume 1039 of *Lecture Notes in Computer Science*, pages 205–218. Springer, 1996.

[MP13] Nicky Mouha and Bart Preneel. Towards finding optimal differential characteristics for ARX: Application to Salsa20. Cryptology ePrint Archive, Report 2013/328, 2013. http://eprint.iacr.org/2013/328.

[NK92] Kaisa Nyberg and Lars R. Knudsen. Provable security against differential cryptanalysis. In Ernest F. Brickell, editor, *Advances in Cryptology - CRYPTO '92, 12th Annual International Cryptology Conference, Santa Barbara, California, USA, August 16-20, 1992, Proceedings*, volume 740 of *Lecture Notes in Computer Science*, pages 566–574. Springer, 1992.

[Nyb94] Kaisa Nyberg. S-boxes and round functions with controllable linearity and differential uniformity. In Bart Preneel, editor, *Fast Software Encryption: Second International Workshop. Leuven, Belgium, 14-16 December 1994, Proceedings*, volume 1008 of *Lecture Notes in Computer Science*, pages 111–130. Springer, 1994.

[PSC+02] Sangwoo Park, Soo Hak Sung, Seongtaek Chee, E-Joong Yoon, and Jongin Lim. On the security of Rijndael-like structures against differential and linear cryptanalysis. In Yuliang Zheng, editor, *Advances in Cryptology - ASIACRYPT 2002, 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002, Proceedings*, volume 2501 of *Lecture Notes in Computer Science*, pages 176–191. Springer, 2002.

[PSLL03] Sangwoo Park, Soo Hak Sung, Sangjin Lee, and Jongin Lim. Improving the upper bound on the maximum differential and the maximum linear hull probability for SPN structures and AES. In Thomas Johansson, editor, *Fast Software Encryption, 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003, Revised Papers*, volume 2887 of *Lecture Notes in Computer Science*, pages 247–260. Springer, 2003.

[Riv11] Ronald L. Rivest. The invertibility of the XOR of rotations of a binary word. *Int. J. Comput. Math.*, 88(2):281–284, 2011.

[STA+14] Yu Sasaki, Yosuke Todo, Kazumaro Aoki, Yusuke Naito, Takeshi Sugawara, Yumiko Murakami, Mitsuru Matsui, and Shoichi Hirose. Minalpher v1. CAESAR Round 1 submission, 2014.

[UCI+11] Markus Ullrich, Christophe De Cannière, Sebastiaan Indesteege, Özgül Küçük, Nicky Mouha, and Bart Preneel. Finding optimal bitsliced implementations of 4×4-bit S-boxes. In *SKEW 2011 Symmetric Key Encryption Workshop, Copenhagen, Denmark, 16–17 February*, 2011.

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	00	01	02	03	00	01	02	03	04	05	06	07	04	05	06	07
01	00	01	02	03	00	01	02	03	04	05	06	07	04	05	06	07
02	08	09	0b	0a	08	09	0b	0a	0c	0d	0f	0e	0c	0d	0f	0e
03	08	09	0b	0a	08	09	0b	0a	0c	0d	0f	0e	0c	0d	0f	0e
04	00	01	02	03	02	03	00	01	04	05	06	07	06	07	04	05
05	08	09	0a	0b	0a	0b	08	09	0c	0d	0e	0f	0e	0f	0c	0d
06	08	09	0b	0a	0a	0b	09	08	0c	0d	0f	0e	0e	0f	0d	0c
07	00	01	03	02	02	03	01	00	04	05	07	06	06	07	05	04
08	00	01	02	03	00	01	02	03	04	05	06	07	04	05	06	07
09	04	05	06	07	04	05	06	07	00	01	02	03	00	01	02	03
0a	08	09	0b	0a	08	09	0b	0a	0c	0d	0f	0e	0c	0d	0f	0e
0b	0c	0d	0f	0e	0c	0d	0f	0e	08	09	0b	0a	08	09	0b	0a
0c	00	01	02	03	02	03	00	01	04	05	06	07	06	07	04	05
0d	0c	0d	0e	0f	0e	0f	0c	0d	08	09	0a	0b	0a	0b	08	09
0e	08	09	0b	0a	0a	0b	09	08	0c	0d	0f	0e	0e	0f	0d	0c
0f	04	05	07	06	06	07	05	04	00	01	03	02	02	03	01	00

Table 11: S-box S_2

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	00	00	01	01	00	00	01	01	02	02	03	03	02	02	03	03
01	04	06	05	07	04	06	05	07	06	04	07	05	06	04	07	05
02	08	08	0d	0d	08	08	0d	0d	0a	0a	0f	0f	0a	0a	0f	0f
03	0c	0e	09	0b	0c	0e	09	0b	0e	0c	0b	09	0e	0c	0b	09
04	00	00	01	01	08	08	09	09	02	02	03	03	0a	0a	0b	0b
05	04	06	05	07	0c	0e	0d	0f	06	04	07	05	0e	0c	0f	0d
06	08	08	0d	0d	00	00	05	05	0a	0a	0f	0f	02	02	07	07
07	0c	0e	09	0b	04	06	01	03	0e	0c	0b	09	06	04	03	01
08	00	00	01	01	01	01	00	00	02	02	03	03	03	03	02	02
09	04	06	05	07	05	07	04	06	06	04	07	05	07	05	06	04
0a	08	08	0d	0d	09	09	0c	0c	0a	0a	0f	0f	0b	0b	0e	0e
0b	0c	0e	09	0b	0d	0f	08	0a	0e	0c	0b	09	0f	0d	0a	08
0c	00	00	01	01	09	09	08	08	02	02	03	03	0b	0b	0a	0a
0d	04	06	05	07	0d	0f	0c	0e	06	04	07	05	0f	0d	0e	0c
0e	08	08	0d	0d	01	01	04	04	0a	0a	0f	0f	03	03	06	06
0f	0c	0e	09	0b	05	07	00	02	0e	0c	0b	09	07	05	02	00

Table 12: S-box S_3

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	00	00	01	01	02	02	03	03	00	02	01	03	02	00	03	01
01	08	08	09	09	0a	0a	0b	0b	08	0a	09	0b	0a	08	0b	09
02	04	04	05	05	0e	0e	0f	0f	04	06	05	07	0e	0c	0f	0d
03	0c	0c	0d	0d	06	06	07	07	0c	0e	0d	0f	06	04	07	05
04	00	00	01	01	02	02	03	03	04	06	05	07	06	04	07	05
05	08	08	09	09	0a	0a	0b	0b	0c	0e	0d	0f	0e	0c	0f	0d
06	04	04	05	05	0e	0e	0f	0f	00	02	01	03	0a	08	0b	09
07	0c	0c	0d	0d	06	06	07	07	08	0a	09	0b	02	00	03	01
08	00	00	01	01	03	03	02	02	00	02	01	03	03	01	02	00
09	08	08	09	09	0b	0b	0a	0a	08	0a	09	0b	0b	09	0a	08
0a	04	04	05	05	0f	0f	0e	0e	04	06	05	07	0f	0d	0e	0c
0b	0c	0c	0d	0d	07	07	06	06	0c	0e	0d	0f	07	05	06	04
0c	00	00	01	01	03	03	02	02	04	06	05	07	07	05	06	04
0d	08	08	09	09	0b	0b	0a	0a	0c	0e	0d	0f	0f	0d	0e	0c
0e	04	04	05	05	0f	0f	0e	0e	00	02	01	03	0b	09	0a	08
0f	0c	0c	0d	0d	07	07	06	06	08	0a	09	0b	03	01	02	00

Table 13: S-box S_4