Bachelor thesis
Computing Science

Radboud University

Cryptanalysis of SPARKLE's ARX-Box Alzette

Author:
Ties Speel
S1020150

First supervisor/assessor:
prof. dr. J.J.C. Daemen (Joan)
j.daemen@cs.ru.nl

Second assessor:
dr. ir. B.J.M. Mennink (Bart)
b.mennink@cs.ru.nl

June 10, 2022

Abstract

The NIST Lightweight Cryptography competition has reached its final round, leaving just 10 cryptographic algorithms in the running. Among these 10 finalists is an ARX-based submission named SPARKLE. In this paper, SPARKLE's ARX-box Alzette is tested using differential and linear cryptanalysis. In particular, its collision resistance against differential collision attacks using 2-dimensional querysets is determined in a keyed hashing setup, and the clustering of its linear trails is observed and explained.

Contents

Notations

F_2	The field with elements $\{0, 1\}$
F_2^n	The vector space with n dimensions, with elements in $\{0, 1\}$.
word	An element of the set F_2^{32}
branch	A pair of words
$\#A$	The number of items in set A
$\oplus$	XOR / Exclusive OR
$\|\|$	Bitstring concatenation
$x \ \mathsf{n} \ n$	Bitstring x left rotated by n bits
$x \ \mathsf{o} \ n$	Bitstring x right rotated by n bits
$x \ll n$	Bitstring x left shifted by n bits
$x \gg n$	Bitstring x right shifted by n bits
ARX	Add-Rotation-XOR

Chapter 1

Introduction

1.1 NIST Lightweight Cryptography Competition

The National Institute of Standards and Technology (NIST) is an organization regarding technical standards that is part of the U.S. Department of Commerce. NIST is the main driving force behind standardizing cryptographic algorithms, responsible for standardizing algorithms like DES and AES. In 2015, NIST announced a new cryptographic competition for lightweight cryptography designed for microcontrollers and other IoT devices, of which the winning algorithm becomes the new cryptographic standard for these devices [9]. Submissions for the competition closed in February 2019.
NIST urges researchers to do as much research as possible on all candidates of this competition, so that they have as much research as possible to help them decide the winner. This is because there is a knowledge gap present in these candidates, since all candidates are new or newly improved algorithms. This knowledge gap is most apparent in the finalist SPARKLE [2], because it is an ARX-box based cipher and ARX is notoriously more difficult to research.

1.2 SPARKLE

SPARKLE is a family of cryptographic permutations based on ARX-boxes. It is the core of the hash function family ESCH and authenticated encryption scheme SCHWAEMM, which are finalists in the NIST Lightweight cryptography competition. SPARKLE comes in three versions corresponding to three block sizes, SPARKLE256, SPARKLE384 and SPARKLE512 [2].

1.3 Research Question

To test SPARKLE's security strength, it will be tested against differential and linear cryptanalysis in this paper. First, the collision resistance of its ARX-box Alzette will be tested in a differential collision attack in a keyed hashing setup, followed by an inspection of the clustering of its linear trails.

Chapter 2

Preliminaries

This chapter contains the specifications of SPARKLE/Alzette and all background information required for chapter 3, where Alzette's security against differential collision attacks is determined in a keyed hashing setup and the clustering of linear trails in Alzette is observed.

2.1 SPARKLE's specification

SPARKLE consists of multiple instances of a non-linear 64-bit mapping called Alzette, defined in section 2.2, and a linear layer that combines and shuffles the output of these instances. The structure of SPARKLE is depicted in figure 2.1.

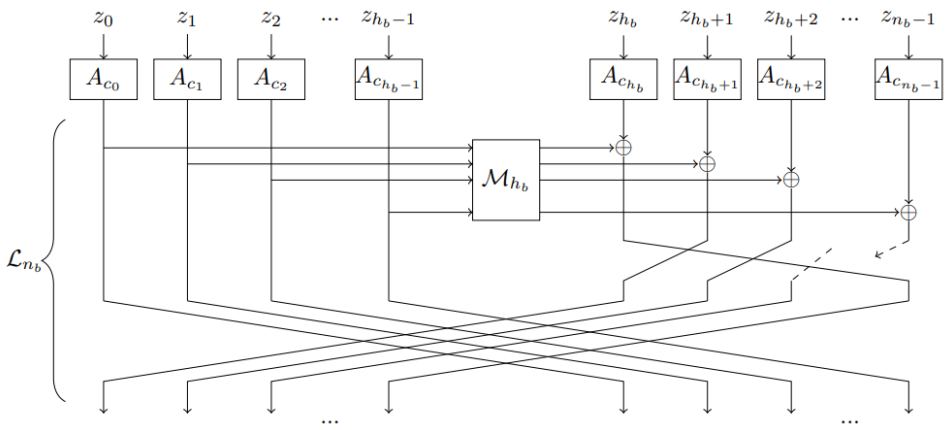

Figure 2.1: SPARKLE's structure where z_i is 64-bit input (x_i, y_i) to its corresponding Alzette instance [2].

To be secure, ciphers need to be non-linear, and all its output bits have

to depend on all bits of the input (diffusion). In the case of SPARKLE, the ARX-boxes provide non-linearity and internal diffusion on every branch while the linear layer combines the output of the ARX-boxes to provide diffusion between all branches.

2.1.1 ARX

The principle behind ARX is to alternate between addition and XOR. Despite the fact that both addition and XOR are linear in their own right, they are non-linear when defined in terms of one another. In modular addition words are interpreted as integers, while they are interpreted as vectors with binary components in XOR.
An attacker has to choose which representation he uses in his attack, making the other operation non-linear.

2.1.2 Linear Layer

SPARKLE's linear layer is denoted $\mathcal{L}_{nb}$ in figure 2.1. It employs a Feistel structure, with $\mathcal{M}_{hb}$ as its Feistel function, where nb is the number of branches and $hb = \frac{nb}{2}$.

Definition 1. $\mathcal{M}_{hb}$ *[2]:*
Let hb *be an integer* > 1. $\mathcal{M}_{hb}$ *is the permutation of* $(\mathsf{F}_2^{32})^w$ *such that*
$\mathcal{M}_{hb}((x_0, y_0), ..., (x_{hb-1}, y_{hb-1})) = ((u_0, v_0), ..., (u_{hb-1}, v_{hb-1}))$
where (u_i, v_i) *must satisfy the following equations:*
$t_y \leftarrow \bigoplus_{i=0}^{hb-1} y_i, t_x \leftarrow \bigoplus_{i=0}^{hb-1} x_i$,
$u_i \leftarrow x_i \oplus \ell(t_y), \forall i \in \{0, ..., hb-1\}$,
$v_i \leftarrow y_i \oplus \ell(t_x), \forall i \in \{0, ..., hb-1\}$,
where all indices are taken modulo hb *and where* $\ell : \mathsf{F}_2^{32} \rightarrow \mathsf{F}_2^{32}$ *is a permutation de ned as:*
$\ell(\mathrm{x}) = (\mathrm{x} \gg 16) \oplus (\mathrm{x} \wedge \mathbf{0x0000ffff})$,
where $x \wedge y$ *denotes the bitwise* AND *of* x *and* y.
This means that if y *and* z *are in* F_2^{16} *so that* $y \| z \in \mathsf{F}_2^{32}$, *then*
$\ell(y \| z) = z \| (y \oplus z)$.

The linear layer $\mathcal{L}_{nb}$ applies the corresponding Feistel function $\mathcal{M}_{nb}$, rotates the right branches by 1 branch to the left, after which it swaps the left branches and the right branches. The exact algorithms describing the linear layers used in the three different branch-width versions of SPARKLE are given below:

Algorithm 1 $\mathcal{L}_4$ [2]
Input/Output: $((x_0, y_0), ..., (x_3, y_3)) \in (\mathsf{F}_2^{32} \times \mathsf{F}_2^{32})^4$

$(t_x, t_y) \leftarrow (x_0 \oplus x_1, y_0 \oplus y_1)$
$(t_x, t_y) \leftarrow ((t_x \oplus (t_x \ll 16))$ n $16, (t_y \oplus (t_y \ll 16))$ n $16)$
$(y_2, y_3) \leftarrow (y_2 \oplus y_0 \oplus t_x, y_3 \oplus y_1 \oplus t_x)$
$(x_2, x_3) \leftarrow (x_2 \oplus x_0 \oplus t_y, x_3 \oplus x_1 \oplus t_y)$
$(x_0, x_1, x_2, x_3) \leftarrow (x_3, x_2, x_0, x_1)$
$(y_0, y_1, y_2, y_3) \leftarrow (y_3, y_2, y_0, y_1)$
return $((x_0, y_0), ..., (x_3, y_3))$

Algorithm 2 $\mathcal{L}_6$ [2]
Input/Output: $((x_0, y_0), ..., (x_5, y_5)) \in (\mathsf{F}_2^{32} \times \mathsf{F}_2^{32})^6$

$(t_x, t_y) \leftarrow (x_0 \oplus x_1 \oplus x_2, y_0 \oplus y_1 \oplus y_2)$
$(t_x, t_y) \leftarrow ((t_x \oplus (t_x \ll 16))$ n $16, (t_y \oplus (t_y \ll 16))$ n $16)$
$(y_3, y_4, y_5) \leftarrow (y_3 \oplus y_0 \oplus t_x, y_4 \oplus y_1 \oplus t_x, y_5 \oplus y_2 \oplus t_x)$
$(x_3, x_4, x_5) \leftarrow (x_3 \oplus x_0 \oplus t_y, x_4 \oplus x_1 \oplus t_y, x_5 \oplus x_2 \oplus t_y)$
$(x_0, x_1, x_2, x_3, x_4, x_5) \leftarrow (x_4, x_5, x_3, x_0, x_1, x_2)$
$(y_0, y_1, y_2, y_3, y_4, y_5) \leftarrow (y_4, y_5, y_3, y_0, y_1, y_2)$
return $((x_0, y_0), ..., (x_5, y_5))$

Algorithm 3 $\mathcal{L}_8$ [2]
Input/Output: $((x_0, y_0), ..., (x_7, y_7)) \in (\mathsf{F}_2^{32} \times \mathsf{F}_2^{32})^8$

$(t_x, t_y) \leftarrow (x_0 \oplus x_1 \oplus x_2 \oplus x_3, y_0 \oplus y_1 \oplus y_2 \oplus y_3)$
$(t_x, t_y) \leftarrow ((t_x \oplus (t_x \ll 16))$ n $16, (t_y \oplus (t_y \ll 16))$ n $16)$
$(y_4, y_5, y_6, y_7) \leftarrow (y_4 \oplus y_0 \oplus t_x, y_5 \oplus y_1 \oplus t_x, y_6 \oplus y_2 \oplus t_x, y_7 \oplus y_3 \oplus t_x)$
$(x_4, x_5, x_6, x_7) \leftarrow (x_4 \oplus x_0 \oplus t_y, x_5 \oplus x_1 \oplus t_y, x_6 \oplus x_2 \oplus t_y, x_7 \oplus x_3 \oplus t_y)$
$(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7) \leftarrow (x_5, x_6, x_7, x_4, x_0, x_1, x_2, x_3)$
$(y_0, y_1, y_2, y_3, y_4, y_5, y_6, y_7) \leftarrow (y_5, y_6, y_7, y_4, y_0, y_1, y_2, y_3)$
return $((x_0, y_0), ..., (x_7, y_7))$

2.2 Alzette

Alzette is a non-linear 64-bit mapping, providing non-linearity and diffusion within its branch. It relies on ARX, which is why its creators have dubbed it an ARX-box. It has a Feistel-like structure, as shown in figure 2.2. Its exact definition is given in algorithm 4. Alzette generally consists of 4 rounds,

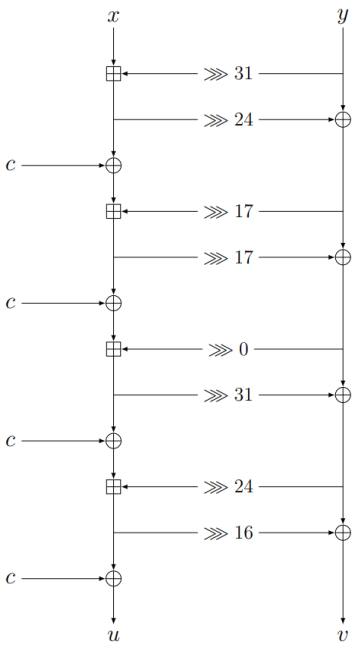

Figure 2.2: Alzette's structure [2, 3]

where every round consists of a linear and non-linear step. As shown in algorithm 4, one round of Alzette is made up by the following three steps:

1. $x \leftarrow x + (y \text{ o } n)$
2. $y \leftarrow y \oplus (x \text{ o } n)$
3. $x \leftarrow x \oplus c$

In section 3.2, the five round version of Alzette is used, meaning the following three lines are added to the end of algorithm 4:

1. $x \leftarrow x + (y \text{ o } 31)$
2. $y \leftarrow y \oplus (x \text{ o } 24)$
3. $x \leftarrow x \oplus c$

Algorithm 4 ARX-Box Alzette (A_c) [2, 3]
Input/Output: $(x, y) \in \mathsf{F}_2^{32} \times \mathsf{F}_2^{32}$

$x \leftarrow x + (y \ \mathsf{o} \ 31)$
$y \leftarrow y \oplus (x \ \mathsf{o} \ 24)$
$x \leftarrow x \oplus c$
$x \leftarrow x + (y \ \mathsf{o} \ 17)$
$y \leftarrow y \oplus (x \ \mathsf{o} \ 17)$
$x \leftarrow x \oplus c$
$x \leftarrow x + (y \ \mathsf{o} \ 0)$
$y \leftarrow y \oplus (x \ \mathsf{o} \ 31)$
$x \leftarrow x \oplus c$
$x \leftarrow x + (y \ \mathsf{o} \ 24)$
$y \leftarrow y \oplus (x \ \mathsf{o} \ 16)$
$x \leftarrow x \oplus c$
return (x, y)

2.3 Keyed Hashing

A keyed hashing function F is defined as follows [1]:
$F = K \times M \rightarrow A$,
with:

- K: The finite key space
- M: The finite message space
- A: The finite digest space that forms an additive group

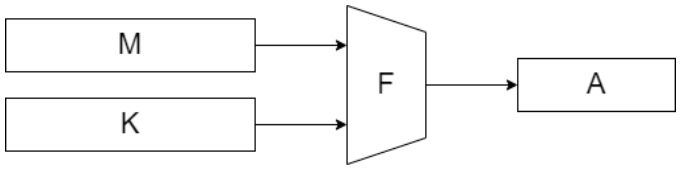

Figure 2.3: A high level overview of Keyed Hashing

A keyed hashing function has a message and key as input, and a digest as output. The message and the key can be any length with padding (as long as the length of the key is at least as long as the message), but the digest always has a fixed length. The goal of a keyed hashing function is to make it difficult to generate collisions for an attacker that does not know the secret key.

To turn any fixed-length permutation function into a keyed hashing function, multiple keyed hashing setups can be used:

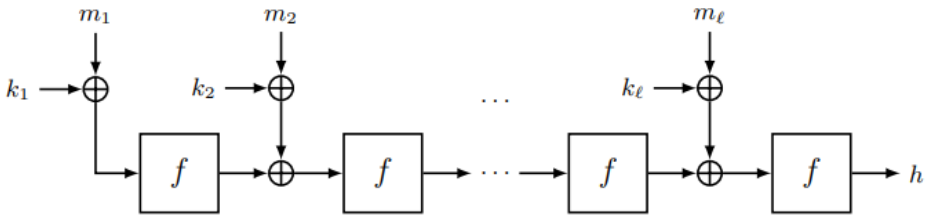

Figure 2.4: Serial keyed hashing setup [8].

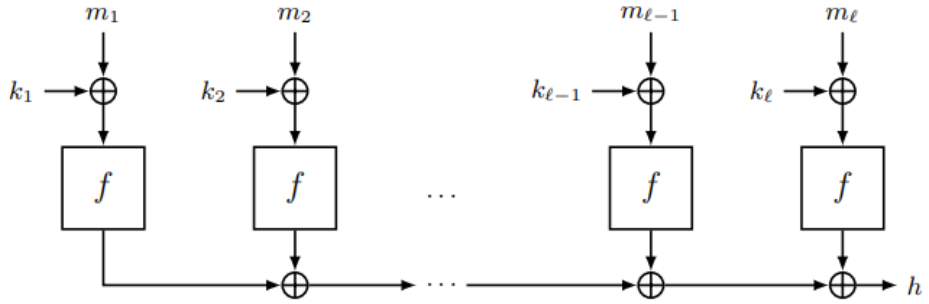

Figure 2.5: Parallel keyed hashing setup [8].

Both setups take a message of the form $m_1 \| m_2 \| \ldots \| m_{l-1} \| m_l$ and a key of the form $k_1 \| k_2 \| \ldots \| k_{l-1} \| k_l$, where all l keys are independent. The main difference in the setups is that the parallel setup is parallelizable, while the serial setup is not.
The possible messages for these setups are strings of elements of a *block space*, where the block space G is the set of elements that form an additive group. Both setups make use of a fixed-length permutation f. This function f maps elements of the block space G to the digest space A.
Examples of such a permutation function f include the SPARKLE permutations, and reduced-round SPARKLE permutations as seen in chapter 3, where 1-round SPARKLE is plugged into the serial setup and tested against differential collision attacks.

2.4 Differential Cryptanalysis

Differential cryptanalysis is a method of analysing how differences in the permutation input affect corresponding differences in the permutation output. In this paper, all differences are defined in terms of XOR (F_2), but

they could also be defined additively (modulo 2^{32} for 32-bit words). Later in section 2.4.2, it is discussed how these differences can be used for a collision attack on keyed hashing. A pair of input difference α and output difference β is called a differential. The probability of such differentials is dependant on the non-linearity of a cipher. This non-linearity often comes from S-boxes, which are lookup tables commonly used in block ciphers to obscure the relationship between the key and ciphertext, but in the case of SPARKLE, the differential probabilities stem from its ARX-box *Alzette*.

Definition 2. ***Differential probability [1]:***
Let $f : G \to A$ be a block function and let $\alpha \in G, \beta \in A$. The differential probability *denoted as DP(α, β) is:*

$$DP(\alpha, \beta) = \frac{\#\{x \in G | f(x) \oplus f(x \oplus \alpha) = \beta\}}{\#G} \tag{2.1}$$

In this paper, f is a permutation and $G = A$.

Definition 3. ***Differential trails [6]:***
Let $f : G \to A$ be a block function with r rounds and let (a, b) be a di erential of f. The di erential trail Q of (a, b) in f consists of a sequence $Q = (q_0, q_1, ..., q_{r-1}, q_r)$ where $q_0 = a$ and $q_r = b$.

Lemma 4. *[6] Let $f : G \to A$ be a block function with r rounds and let Q be a di erential trail of the di erential (a, b) in f consisting of sequence $Q = (q_0, q_1, ..., q_{r-1}, q_r)$ where $q_0 = a$ and $q_r = b$. The di erential probability of trail Q is the number of message pairs with input di erence a that follow trail Q through f to end up with output di erence b, divided by the total number of message pairs with input di erence a.*

Lemma 5. *[6] Let $f : G \to A$ be a block function with r rounds and let (a, b) be a di erential of f. The di erential probability of (a, b) is the sum of the di erential probabilities of all di erential trails Q with $q_0 = a$ and $q_r = b$.*

2.4.1 Differential propagation in Alzette

To illustrate how differentials propagate through Alzette, a differential trail is given below of the differential $(80000100\ 00000080, 80404100\ 41004041)$.

$x \leftarrow x \oplus c$

While the round constants have impact on the linear properties of Alzette, they have no impact on its differentials, as $(x \oplus c) \oplus (x \oplus \delta \oplus c) = \delta$.

The propagation of differential (80000100 00000080, 80404100 41004041) through ARX-box Alzette.

1:	80000100 00000080
2:	$x \leftarrow x + (y \gg 31)$
3:	80000000 00000080
4:	$y \leftarrow y \oplus (x \gg 24)$
5:	80000000 00000000
6:	$x \leftarrow x \oplus c$
7:	80000000 00000000
8:	$x \leftarrow x + (y \gg 17)$
9:	80000000 00000000
10:	$y \leftarrow y \oplus (x \gg 17)$
11:	80000000 00004000
12:	$x \leftarrow x \oplus c$
13:	80000000 00004000
14:	$x \leftarrow x + y$
15:	80004000 00004000
16:	$y \leftarrow y \oplus (x \gg 31)$
17:	80004000 0000c001
18:	$x \leftarrow x \oplus c$
19:	80004000 0000c001
20:	$x \leftarrow x + (y \gg 24)$
21:	80404100 0000c001
22:	$y \leftarrow y \oplus (x \gg 16)$
23:	80404100 41004041
24:	$x \leftarrow x \oplus c$
25:	80404100 41004041

$y \leftarrow y \oplus (x \gg n)$

This operation changes the shape of the differential, but does not directly impact the differential probability. This is because XOR and rotation are linear in regards to differentials defined in terms of XOR, so the differences propagate deterministically through this step. However, this does not mean that this operation is not required for differential security strength. As visible in line 8; $x \leftarrow x + (y \gg 17)$ does not change the difference because $y = 00000000$. Without $y \leftarrow y \oplus (x \gg 17)$ on line 10, this would stay that way, creating a differential with $DP(\alpha, \beta) = 1$.

$x \leftarrow x + (y \gg n)$

This operation is the source of the non-deterministic propagation of differentials defined in terms of XOR in Alzette. As mentioned in section 2.1.1, for XOR the input words can be seen as bit vectors with 32 dimensions.

Since $x + y$ has a carry and vector addition (XOR) does not, this operation becomes non-linear.
Given a message pair with difference $\delta = \delta x \| \delta y$: $(x \| y, x \oplus \delta x \| y \oplus \delta y)$, we can look at a single bit position i in x and y where $\delta x_i = 1$ to see how differences propagate.
For example, let say $y_i = 1$. Now the outcome of $x \leftarrow x + y$ depends on x_i. Two things can happen (with probability 2^{-1}) for a random x:

1. $x_i = 0$; $x_i + y_i = 1$ and there is no carry.
2. $x_i = 1$; $x_i + y_i = 0$ and there is a carry.

When there is a carry, the difference δx_i propagates to the next bit, where the same two things can happen again, based on x_{i+1} and y_{i+1}. This means addition can cause bits to shift to the left, with the probability halving at each bit shift.
For example, the differential $(80000100\ 00000080, 80404100\ 41004041)$ of Alzette has probability 2^{-6}, but $x \leftarrow x + (y \ \texttt{o}\ 24)$ on line 20 could also have propagated to 80404300 0000c001, resulting in a different differential with probability 2^{-7}. Similarly, $(80000300\ 00000080, 80404100\ 41004041)$ is also a differential of Alzette of probability 2^{-7} (After $x \leftarrow x + (y \ \texttt{o}\ 31)$, 80000300 00000080 propagates to 80000000 00000080 at half the probability of 80000100 00000080 doing the same).

2.4.2 Attack vector for keyed hashing

Given a differential of high probability (a, b) of block function f, collisions can be created for both serial and parallel keyed hashing setups.

Serial setup

Figure 2.6 depicts how collisions can be generated in case of a serial keyed hashing setup.
This makes use of a query set $\{M1, M2\}$, where $M1 = m1||m2$ and $M2 = m1 \oplus a||m2 \oplus b$. As shown in figure 2.6, the digest of $M1$ and $M2$ will be equal if $f(m1 \oplus k1 \oplus a) = f(m1 \oplus k1) \oplus b$.
Given an input difference a and an output difference b, the probability of getting a collision in this setup with a 2-message attack is $DP(a, b)$ as defined in definition 2.

Parallel setup

Figure 2.7 depicts how collisions can be generated in case of a parallel keyed hashing setup.

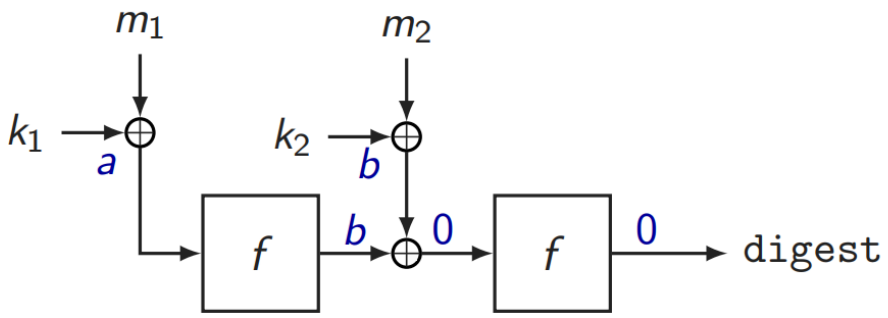

Figure 2.6: Serial keyed hashing setup, where a, b and 0 are differences between input pairs and (a, b) is a differential over f [1, 8].

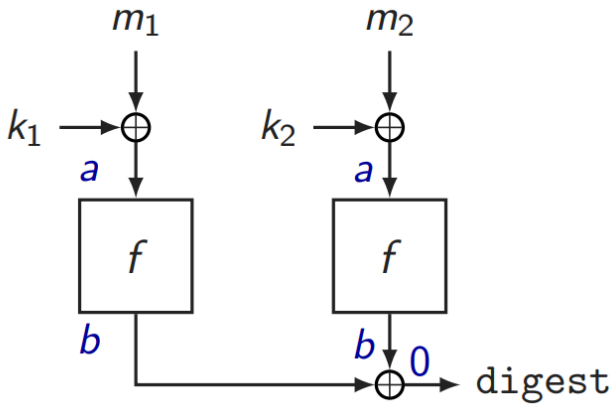

Figure 2.7: Parallel keyed hashing setup, where a, b and 0 are differences between input pairs and (a, b) is a differential over f [1, 8].

This makes use of a query set $\{M1, M2\}$, where $M1 = m1||m2$ and $M2 = m1 \oplus a||m2 \oplus a$. As shown in figure 2.7, the digest of $M1$ and $M2$ will be equal if $f(m1 \oplus k1) \oplus f(m1 \oplus k1 \oplus a) = f(m2 \oplus k2) \oplus f(m2 \oplus k2 \oplus a)$. Given an input difference a, the probability of getting a collision in this setup with a 2-message attack is $\sum_b DP(a, b)^2$. This probability is squared since input difference a needs to propagate to an output difference b in both $f(m1 \oplus k1 \oplus a)$ and $f(m2 \oplus k2 \oplus a)$ for a collision, and the sum is taken over all possible values for b since it does not matter what b is, as long as both $f(m1 \oplus k1 \oplus a)$ and $f(m2 \oplus k2 \oplus a)$ propagate to the same output difference b.

2.4.3 Naive Querysets

From now on, given a message $M = m1 \| m2$ and a differential $d = (\alpha, \beta)$, the message $m1 \oplus \alpha \| m2 \oplus \beta$ will be written as $M \oplus d$.
The simplest collision attack uses a single pair $\{M, M'\}$ as its queryset with $M \neq M'$. The probability of getting a collision with this queryset is equal to the differential probability of the difference between M and M'. Thus the best way of getting a collision this way is to choose M and M' such that they differ by a high-probability differential d, giving you the queryset $\{M, M \oplus d\}$.
Now lets say the attacker sends 4 messages instead of only 2. Now he can query two querysets $\{M, M \oplus d\}$ and $\{M', M' \oplus d\}$, with $M \neq M'$ and d being a high-probability differential, to get double the probability of a collision as before with one queryset. However, this way there are more potential pairings that are lost to the attacker, since there are only two pairings with a high-probability difference.

2.4.4 Multidimensional Querysets

To make use of other possible pairings, the following 2-dimensional queryset can be used: $\{M, M \oplus d1, M \oplus d2, M \oplus d1 \oplus d2\}$ where $d1$ and $d2$ are two high-probability differentials. This 2-dimensional queryset is visualised in figure 2.8. In this queryset, there are 4 pairings with high probability differences, namely M and $M \oplus d1$, M and $M \oplus d2$, $M \oplus d1$ and $M \oplus d1 \oplus d2$ and $M \oplus d2$ and $M \oplus d1 \oplus d2$, giving you 4 chances at a collision as compared to 1 in the queryset $\{M, M \oplus d1\}$. This also generalises to more than 2 dimensions, giving you message sets of size 2^n where n is the number of dimensions.
Assuming the differential probabilities are independent, this queryset gives you a total differential probability of $2DP(d1) + 2DP(d2) + 2DP(d1 \oplus d2)$. However, as is described in section 3.1, these probabilities are typically not independent.

Freedom of d1 $\oplus$ d2

The diagonals in figure 2.8 can also collide with probability $DP(d1 \oplus d2)$, however when $d1$ and $d2$ are chosen to have the highest possible differential probability, the differential probability of $d1 \oplus d2$ is typically much smaller. This is because while you choose $d1$ and $d2$, you can not choose what $d1 \oplus d2$ will be.

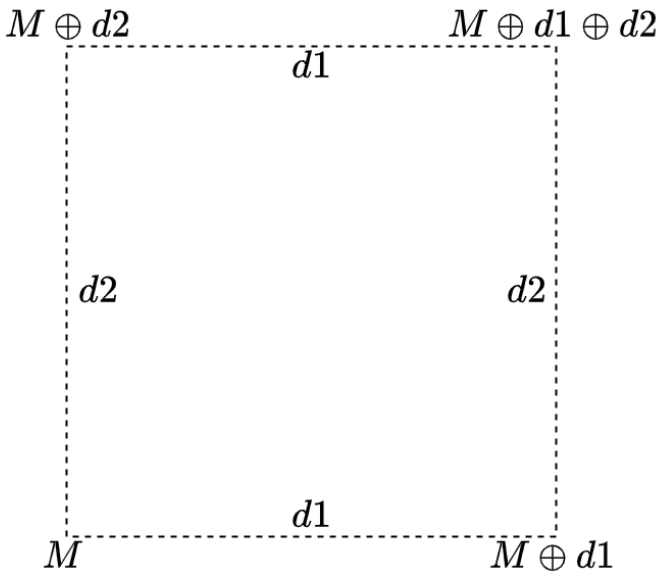

Figure 2.8: 2-dimensional queryset $\{M, M \oplus d1, M \oplus d2, M \oplus d1 \oplus d2\}$

2.5 Linear cryptanalysis

While differential cryptanalysis studies how differences in the permutation input affect corresponding differences in the permutation output, linear cryptanalysis studies probabilistic linear relations between the permutation input and output. If there is a linear relation between permutation input and output of high probability, an attacker could use this relations to estimate (bits of) the secret key. Below are some definitions needed to describe these linear relations [6].

Definition 6. *A parity of a Boolean vector is a binary Boolean function that takes the XOR of a speci c selection of bits in the vector.*

Definition 7. *The linear mask w of a parity is a Boolean vector such that bits included in the parity have a 1 in the same position in the linear mask and a 0 otherwise. The parity of a vector x with linear mask w is denoted $w^T x$.*

This means that a vector x with n bits has 2^n different parities. This set of parities is equal to the set of all linear binary Boolean functions of that vector.

Definition 8. *The correlation $C(f, g)$ between two binary Boolean functions $f(x)$ and $g(x)$ is de ned as:*

$$C(f, g) = 2 \times Prob(f(x) = g(x)) - 1 \tag{2.2}$$

This implies that $C(f,g) = C(g,f)$. The correlation ranges between -1 and 1. If the correlation is not 0, the functions are correlated; If the correlation is 1, the functions are equal; If the correlation is -1, the functions are each other's complement.

Definition 9. *Real valued function:*

$$\hat{f}(x) = (-1)^{f(x)} \tag{2.3}$$

The real-valued function $\hat{f}(x)$ is -1 if f(x) = 1 and 1 if f(x) = 0.

Definition 10. *The inner product of two binary Boolean functions f and g is de ned as:*

$$\langle \hat{f}, \hat{g} \rangle = \sum_{x} \hat{f}(x)\hat{g}(x) \tag{2.4}$$

This implies the following norm:

$$||\hat{f}|| = \sqrt{\langle \hat{f}, \hat{f} \rangle}. \tag{2.5}$$

This means the correlation can also be defined as:

$$C(f,g) = \frac{\langle \hat{f}, \hat{g} \rangle}{||\hat{f}|| \times ||\hat{g}||}. \tag{2.6}$$

For a Boolean function with n variables the norm is always the same, meaning the last two equations can be rewritten in the following way:

$$||\hat{f}|| = \sqrt{2^n}. \tag{2.7}$$

and:

$$C(f,g) = \frac{\langle \hat{f}, \hat{g} \rangle}{2^n}. \tag{2.8}$$

2.5.1 Walsh-Hadamard transform

From now on the correlation between a binary Boolean function $f(x)$ and a parity $w^T x$, $C(f(x), w^T x)$, will be written as $F(w)$.
$\hat{f}(x)$ can be represented in terms of $F(w)$ [6]:

$$\hat{f}(x) = \sum_{w} F(w)(-1)^{w^T x} \tag{2.9}$$

This then gives us:

$$F(w) = 2^{-n} \sum_{x} \hat{f}(x)(-1)^{w^T x} \tag{2.10}$$

This transformation $W : f(x) \mapsto F(w)$ is called the Walsh-Hadamard transform. This transformation can be used to create correlation matrices for the linear masks of the components of a cipher such as S-boxes.

2.5.2 Linear trails

Definition 11. ***Linear trails*** *[6]:*
Let $f : G \to A$ be a block function with r rounds and let (u, v) be a pair of input and output masks. The linear trail U of (u, v) in f consists of a sequence $U = (u_0, u_1, ..., u_{r-1}, u_r)$ where $u_0 = u$ and $u_r = v$.

Definition 12. ***Correlation contribution*** *[6]:*
The correlation contribution of a linear trail is the product of the correlation of all its steps $(u_0, u_1, ..., u_{r-1}, u_r)$. This contribution can be either positive or negative.

Lemma 13. ***Linear approximation*** *[6]:*
Let $f : G \to A$ be a block function with r rounds and let (u, v) be a pair of input and output masks. The correlation of a linear approximation (u, v) is the sum over the correlation contributions of all linear trails with input mask u and output mask v.

The steps that add round constants in Alzette ($x \leftarrow x \oplus c$ in algorithm 4) influence the signs of individual linear trails. The designers of SPARKLE have chosen these round constants so that these individual trails do not add up in a constructive way in the largest linear approximations. The result these round constants have on the correlation of the linear approximations will be studied in section 3.2.2.

Chapter 3

Research

This chapter is split into two sections, one covering an experiment that tests Alzette against differential cryptanalysis and another covering an experiment that describes observed dominant linear trails in Alzette.

3.1 Differential Cryptanalysis

In this section, an experiment is described that tests Alzette's security against differential collision attacks using multidimensional querysets. In particular, this experiments gives insight into the grouping of differentials in Alzette.

3.1.1 Differential grouping

As discussed in section 2.4.4, given a random message M and differentials d, $d1$ and $d2$ with equal probabilities, a 2-dimensional queryset $\{M, M \oplus d1, M \oplus d2, M \oplus d1 \oplus d2\}$ should give you differential probability $2DP(d1) + 2DP(d2) + 2DP(d1 \oplus d2)$ compared to $DP(d)$ for queryset $\{M, M \oplus d\}$ if you just add up the probabilities of the differentials. Adding up the probabilities assumes that the probabilities are independent of each other, however, in practice these probabilities are not independent.
Given two differentials $d1$ and $d2$ of probability p, the queryset $\{M, M \oplus d1, M \oplus d2, M \oplus d1 \oplus d2\}$ gives 6 possible pairings for a collision, of which 4 have probability p and 2 have smaller probability. This means that you can have more than one collision in a single queryset. However, for an attack it does not matter how many collisions there are in a queryset, as long as there is at least one. This means multiple collisions in a single queryset are not desirable for an attacker, as they reduce the total number of querysets in which a collision occurs. So while this queryset gives you more chances at collisions, it could actually result in a smaller differential probability per message queried. This means that the designers of a permutation function

want its differentials to group together as much as possible, while an attacker wants there to be as little grouping of the collisions as possible.

3.1.2 Grouping of Alzette's differentials

To study the grouping of Alzette's differentials, the queryset $\{M, M \oplus d1, M \oplus d2, M \oplus d1 \oplus d2\}$ has been queried to Alzette for a random message M and a sample size of 2^{24}. For the differentials $d1$ and $d2$, all 2-combinations have been taken over the differentials given in table 3.1. The results of this experiment are given in table 3.2.

α	β
80000100 00000080	80404100 41004041
80000100 00000080	80c04100 410040c1
00804001 80400000	80000180 81808001
00804001 80400000	80000080 80808001
a0008140 000040a0	80000100 01008001
80020100 00010080	01010000 00030101
80020100 00010080	03010000 00030301

Table 3.1: Differential trails with probability 2^{-6} over ARX-box Alzette as given in [3].

The higher the value in the column **prob d1 & d2** in table 3.2, the better the differentials $d1$ and $d2$ are for an attacker to use in a 2-dimensional queryset, as a higher value means fewer double collisions per queryset. Thus the best differentials to pick for a 2-dimensional queryset are (80000100 00000080, 80c04100 410040c1) and (00804001 80400000, 80000080 80808001), with differential probability 1.87×2^{-5}.
It should also be noted that all tested 2-dimensional querysets where $d1$ $\alpha = d2$ α have a differential probability of 2^{-5}. This is because in these querysets, the message $M \oplus d1 \oplus d2$ has no input difference with the message M, since the input differences of $d1$ and $d2$ are the same, cancelling each other out. This means that collisions in this queryset always come in pairs, either with $\Delta = d1$ or $\Delta = d2$ (there can not be a double collision with $\Delta = d1 \oplus d2$ since that would mean a collision with input difference 0).

Differentials					Probability of double collisions*		
d1 α	d1 β	d2 α	d2 β	prob d1 & d2	$\Delta = d1$	$\Delta = d2$	$\Delta = d1 \oplus d2$
80000100 00000080	80404100 41004041	80000100 00000080	80c04100 410040c1	2^{-5}	2^{-6}	2^{-6}	0
80000100 00000080	80404100 41004041	00804001 80400000	80000180 81808001	1.45×2^{-5}	1.89×2^{-9}	1.75×2^{-7}	1.56×2^{-11}
80000100 00000080	80404100 41004041	00804001 80400000	80000080 80808001	1.85×2^{-5}	1.89×2^{-9}	1.17×2^{-12}	0
80000100 00000080	80404100 41004041	a0008140 000040a0	80000100 01008001	1.58×2^{-5}	1.69×2^{-10}	2^{-6}	1.02×2^{-9}
80000100 00000080	80404100 41004041	80020100 00010080	01010000 00030101	1.61×2^{-5}	1.25×2^{-7}	1.49×2^{-9}	1.97×2^{-12}
80000100 00000080	80404100 41004041	80020100 00010080	03010000 00030301	1.6×2^{-5}	1.25×2^{-7}	1.76×2^{-9}	2^{-11}
80000100 00000080	80c04100 410040c1	00804001 80400000	80000180 81808001	1.47×2^{-5}	1.53×2^{-9}	1.74×2^{-7}	1.58×2^{-11}
80000100 00000080	80c04100 410040c1	00804001 80400000	80000080 80808001	1.87×2^{-5}	1.53×2^{-9}	1.13×2^{-12}	0
80000100 00000080	80c04100 410040c1	a0008140 000040a0	80000100 01008001	1.54×2^{-5}	1.79×2^{-10}	2^{-6}	1.46×2^{-9}
80000100 00000080	80c04100 410040c1	80020100 00010080	01010000 00030101	1.61×2^{-5}	1.25×2^{-7}	1.5×2^{-9}	2^{-11}
80000100 00000080	80c04100 410040c1	80020100 00010080	03010000 00030301	1.59×2^{-5}	1.25×2^{-7}	1.74×2^{-9}	1.98×2^{-12}
00804001 80400000	80000180 81808001	00804001 80400000	80000080 80808001	2^{-5}	2^{-6}	2^{-6}	0
00804001 80400000	80000180 81808001	a0008140 000040a0	80000100 01008001	1.6×2^{-5}	1.22×2^{-8}	2^{-7}	1.01×2^{-10}
00804001 80400000	80000180 81808001	80020100 00010080	01010000 00030101	1.57×2^{-5}	1.49×2^{-7}	1.61×2^{-10}	2^{-12}
00804001 80400000	80000180 81808001	80020100 00010080	03010000 00030301	1.55×2^{-5}	1.49×2^{-7}	2^{-9}	2^{-12}
00804001 80400000	80000080 80808001	a0008140 000040a0	80000100 01008001	1.6×2^{-5}	1.87×2^{-9}	2^{-7}	0
00804001 80400000	80000080 80808001	80020100 00010080	01010000 00030101	1.57×2^{-5}	1.5×2^{-7}	1.59×2^{-10}	2^{-12}
00804001 80400000	80000080 80808001	80020100 00010080	03010000 00030301	1.55×2^{-5}	1.49×2^{-7}	2^{-9}	2^{-12}
a0008140 000040a0	80000100 01008001	80020100 00010080	01010000 00030101	1.51×2^{-5}	1.78×2^{-7}	1.03×2^{-9}	1.07×2^{-11}
a0008140 000040a0	80000100 01008001	80020100 00010080	03010000 00030301	1.5×2^{-5}	1.78×2^{-7}	1.2×2^{-9}	1.03×2^{-11}
80020100 00010080	01010000 00030101	80020100 00010080	03010000 00030301	2^{-5}	2^{-6}	2^{-6}	0

* Probability that queryset $\{M, M \oplus d1, M \oplus d2, M \oplus d1 \oplus d2\}$ leads to two collisions with difference Δ for random message M.

Table legend for table 3.2:

$d1$ α Input difference of differential $d1$.

$d1$ β Output difference of differential $d1$.

$d2$ α Input difference of differential $d2$.

$d2$ β Output difference of differential $d2$.

prob $d1\&d2$ Probability of a collision in queryset $\{M, M \oplus d1, M \oplus d2, M \oplus d1 \oplus d2\}$.

$\Delta = d1$ Probability of collisions in both $\{M, M \oplus d1\}$ and $\{M \oplus d2, M \oplus d1 \oplus d2\}$.

$\Delta = d2$ Probability of collisions in both $\{M, M \oplus d2\}$ and $\{M \oplus d1, M \oplus d1 \oplus d2\}$.

$\Delta = d1 \oplus d2$ Probability of collisions in both $\{M, M \oplus d1 \oplus d2\}$ and $\{M \oplus d1, M \oplus d2\}$.

Table 3.2: 2-dimensional probabilities

3.2 Linear Cryptanalysis

In this section, an experiment is described that determines the number of dominant linear trails and their probabilities in the five rounds version of Alzette.
The sign of the correlations have been left out in this chapter, since the sign of a correlation is much less important than the size of the correlation for security against linear cryptanalysis.

3.2.1 Deducing linear trails from correlations

The correlation of a linear approximation between an input and output mask combination can be used to deduce properties of the dominant linear trails in a cipher. This is possible by looking at the linear approximation's correlation in binary representation, since individual trails typically have a correlation that is a power of two.
For example, given a linear approximation between two masks with binary correlation 0.000010101 (0.041015625 in decimal), you can deduce the presence of three trails with contributions of 2^{-5}, 2^{-7} and 2^{-9} respectively, which all contribute positively to the correlation (or all contribute negatively). Similarly, for a linear approximation with binary correlation 0.0000011010, the same trails with contributions 2^{-5}, 2^{-7} and 2^{-9} can be deduced. However, in this case the three trails do not all contribute positively/negatively to the correlation, but instead the trails with contributions 2^{-5} and 2^{-9} contribute positively and the trail with contribution 2^{-7} contributes negatively (or the other way around).

3.2.2 Linear trails in Alzette

To determine the number of linear trails present in the five rounds version of Alzette, the correlations for input mask 0000020180020180 and output mask 01c00181c1808081 of 100 randomly generated round constants were approximated over a sample size of 2^{24} random messages. Using the technique from section 3.2.1, it was found that there are three dominant trails:

A Trail with correlation contribution 2^{-5}

B Trail with correlation contribution 2^{-7}

C Trail with correlation contribution 2^{-9}

Since there are three dominant trails, the round constants fall into 8 groups (sorted from largest to smallest correlations):

1. Group of constants where trails A,B and C are all either positive or negative, and where smaller trails have the same sign. The binary representation of this group starts with 0.0000101011 or 0.0000101010.

2. Group of constants where trails A,B and C are all either positive or negative, and where smaller trails have a different sign. The binary representation of this group starts with 0.0000101001.

3. Group of constants where trails A and B are either positive or negative, trail C has a different sign to trails A and B, and where smaller trails have the same sign as A and B. The binary representation of this group starts with 0.0000100111 or 0.0000100110.

4. Group of constants where trails A and B are either positive or negative, trail C has a different sign to trails A and B, and where smaller trails have a different sign to A and B. The binary representation of this group starts with 0.0000100101.

5. Group of constants where trails A and C are either positive or negative, trail B has a different sign to trails A and C, and where smaller trails have the same sign as A and C. The binary representation of this group starts with 0.0000011010.

6. Group of constants where trails A and C are either positive or negative, trail B has a different sign to trails A and C, and where smaller trails have a different sign to A and C. The binary representation of this group starts with 0.0000011001.

7. Group of constants where trails B and C are either positive or negative, trail A has a different sign to trails B and C, and where smaller trails have the same sign as A. The binary representation of this group starts with 0.0000010110.

8. Group of constants where trails B and C are either positive or negative, trail A has a different sign to trails B and C, and where smaller trails have a different sign to A. The binary representation of this group starts with 0.0000010101 or 0.0000010100.

The correlations can be found in table A.1, and the code used to run the experiment can be found in the appendix A.1.

Chapter 4

Related Work

All SPARKLE documentation can be found in the official SPARKLE specification *Schwaemm and Esch: Lightweight Authenticated Encryption and Hashing using the Sparkle Permutation Family* [2], together with the results of the authors own differential and linear cryptanalysis of SPARKLE. Similar information be found for ARX-box Alzette in *Alzette: a 64-bit ARX-box (feat. CRAX and TRAX)* [3].

Details regarding differential and linear cryptanalysis can be found in *The Design of Rijndael: AES | The Advanced Encryption Standard* [6], where they are applied to AES.

The methodology of generating collisions in keyed hashing using differential cryptanalysis is discussed in detail in an internal rapport by Radboud on collision-resistance [1].

The multidimensional querysets studied in this paper can be seen as a more specific version of polytopic cryptanalysis. Both polytopic cryptanalysis and the experiment in section 3.1 look at sets with differences between elements, whereas standard differential cryptanalysis only looks at differences between message pairs. The difference between this paper and polytopic cryptanalysis is that the querysets studied in section 3.1 are tailored specifically for collisions, while polytopic cryptanalysis is more general. Details regarding polytopic cryptanalysis can be found in *Polytopic Cryptanalysis* [10]. A similar study on querysets using multiple differentials can be found in *Multiple Di erential Cryptanalysis: Theory and Practice* [5].

While not relevant for the attacks described in this paper, it should be noted that there exist specialised types of differential cryptanalysis. These include *Higher-order diferential cryptanalysis* [7], *Truncated di erential cryptanalysis* [7], *Impossible di erential cryptanalysis* [4] and *Boomerang attacks* [11].

Chapter 5

Conclusions

In this paper, SPARKLE has been tested with both differential and linear cryptanalysis.
The collision resistance of its ARX-box Alzette against a differential collision attack using 2-dimensional querysets has been studied in a keyed hashing setup, of which the results can be found in table 3.2. The best 2-dimensional queryset found had differential probability 1.87×2^{-5}.
This shows that, even for the 2-dimensional queryset with highest probability, the differential probabilities are not independent. If the differential probabilities were independent, you would get a probability of $2DP(d1) + 2DP(d2) + 2DP(d1 \oplus d2)$. When $d1$ and $d2$ are chosen such that $DP(d1) = 2^{-6}$ and $DP(d2) = 2^{-6}$, this probability would lie somewhere between 2^{-4} and 1.5×2^{-4}, depending on what $DP(d1 \oplus d2)$ is for the chosen $d1$ and $d2$.

The clustering of Alzette's linear trails has been studied for a specific input/output mask combination, which lead to the observation that there are three dominant trails with correlation contribution 2^{-5},2^{-7} and 2^{-9} respectively. This caused round constants to fall into one of 8 groups as described in section 3.2.2.

Further work could include increasing the number of dimensions in the differential collision attack to more than two dimensions. Since this paper was mostly focused on Alzette, SPARKLE was limited to one round. Further work could also include extending the experiments done to multiple rounds of SPARKLE, which would expand the scope to its linear layer as well.

Bibliography

[1] Radboud internal report on collision-resistance.

[2] Christof Beierle, Alex Biryukov, Luan Cardoso dos Santos, Johann Großschädl, Amir Moradi, Léo Perrin, Aein Rezaei Shahmirzadi, Aleksei Udovenko, Vesselin Velichkov, and Qingju Wang. Schwaemm and esch: Lightweight authenticated encryption and hashing using the sparkle permutation family. https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/finalist-round/updated-spec-doc/sparkle-spec-final.pdf.

[3] Christof Beierle, Alex Biryukov, Luan Cardoso dos Santos, Johann Großschädl, Léo Perrin, Aleksei Udovenko, Vesselin Velichkov, and Qingju Wang. Alzette: a 64-bit arx-box (feat. crax and trax). Cryptology ePrint Archive, Report 2019/1378, 2019. https://ia.cr/2019/1378.

[4] Eli Biham, Alex Biryukov, and Adi Shamir. Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. Journal of Cryptology, 2005. https://doi.org/10.1007/s00145-005-0129-3.

[5] Céline Blondeau and Benoît Gérard. Multiple differential cryptanalysis: Theory and practice. Springer Berlin Heidelberg, 2011. https://doi.org/10.1007/978-3-642-21702-9_3.

[6] Joan Daemen and Vincent Rijmen. The design of rijndael. Springer-Verlag, November 26, 2001. https://cs.ru.nl/~joan/papers/JDA_VRI_Rijndael_2002.pdf.

[7] Lars R. Knudsen. Truncated and higher order differentials. Springer Berlin Heidelberg, 1995. https://doi.org/10.1007/3-540-60590-8_16.

[8] Figures made by Jonathan Fuchs.

[9] Kerry McKay (NIST), Lawrence Bassham (NIST), Meltem Sönmez Turan (NIST), and Nicky Mouha (NIST). Report on lightweight cryptography. National Institute of Standards and Technology Internal Report 8114, 2017. https://doi.org/10.6028/NIST.IR.8114.

[10] Tyge Tiessen. Polytopic cryptanalysis. Springer Berlin Heidelberg, 2016. https://doi.org/10.1007/978-3-662-49890-3_9.

[11] David Wagner. The boomerang attack. Springer Berlin Heidelberg, 1999. https://doi.org/10.1007/3-540-48519-8_12.

Appendix A

Appendix

A.1 Code

The code used to run the experiments can be found here:
`https://github.com/TSpeel/Sparkle-thesis/`

A.2 Linear cryptanalysis results

This table shows the correlation of 100 randomly generated round constants for input mask 0000020180020180 and output mask 01c00181c1808081, sampled over 2^{24} random messages.

rc	$-\log_2(\lvert c\rvert)$	rc	$-\log_2(\lvert c\rvert)$	rc	$-\log_2(\lvert c\rvert)$	rc	$-\log_2(\lvert c\rvert)$
40555465	5.61	b3a9d47b	5.51	5942d277	5.29	978bda9f	4.72
42538192	5.59	34452fe4	5.51	20a76d2c	5.28	9106722f	4.71
3f44b42f	5.58	cab8ed62	5.5	5f1c3b1d	5.28	8a37a4a7	4.64
429c63ae	5.58	b43b8a8d	5.5	581d5898	5.28	8e9edbe3	4.64
44a9d499	5.58	b15d90df	5.49	a7054d8c	5.27	0ae81014	4.63
c43bdeec	5.58	31d48e1d	5.49	27720ce7	5.27	702e08ce	4.63
b9bf8796	5.58	30948165	5.48	5cc3b8df	5.26	73fa47e7	4.63
c3f33ece	5.58	523a4254	5.36	5c7656e2	5.25	f6915da7	4.62
3d789a1a	5.58	52b3b4a7	5.35	a096afd5	5.24	f3f215f2	4.62
be811cce	5.57	d1286714	5.35	6518be93	4.78	f2312bcb	4.62
3d54d5ec	5.57	2fcfda16	5.34	9f4e0155	4.77	74acae12	4.62
3bec6afd	5.57	d23660f8	5.34	6091b7ec	4.77	76556c4f	4.62
4666ffe5	5.56	ab7dfeb0	5.33	1d19c0fc	4.77	88d3a127	4.62
4641eaf1	5.55	56ba1aa7	5.33	e771d074	4.76	79f1d074	4.61
b84f540f	5.55	5446d427	5.33	e66ac4a8	4.76	074632cd	4.6
ba34fe94	5.55	558597ef	5.32	1b11d627	4.76	7e1fc1df	4.6
c4d94b76	5.55	29552f94	5.32	9e050bb1	4.75	04595a44	4.6
390bef3a	5.54	aadda9b9	5.32	e759ad29	4.75	84e33dd3	4.6
c6b73613	5.54	2e25ced3	5.32	968bc50d	4.74	040f3d9a	4.59
456582f8	5.54	2b3b3ade	5.32	97218447	4.74	8374fac8	4.59
b5f5eee1	5.53	2d66db51	5.31	96a2e730	4.74	7949c9bb	4.59
b61d8f10	5.52	d832a428	5.31	e81422fc	4.74	85b5adab	4.58
b4ef942a	5.52	a6326cea	5.3	171e029f	4.73	808aef56	4.58
cebb15cb	5.51	54a50e46	5.3	93c49633	4.73	046500d0	4.58
c83a3584	5.51	21daf2d6	5.29	956077d5	4.73	7eb03c2b	4.57

Table A.1: Correlations