Technical University of Denmark

Secure Block Ciphers - Cryptanalysis and Design

Tiessen, Tyge; Rechberger, Christian; Knudsen, Lars Ramkilde

Publication date:
2017

Document Version
Publisher's PDF, also known as Version of record

Secure Block Ciphers

Cryptanalysis and Design

Tyge Tiessen

Ph.D. Thesis

April 2016

Document compiled on April 25, 2016.

Supervisor: Christian Rechberger
Co-supervisor: Lars R. Knudsen

Technical University of Denmark
Department of Applied Mathematics and Computer Science

ISSN: 0909-3192
Serial no.: PHD-2016-412

Abstract

The rapid evolution of computational devices and the widespread adoption of digital communication have deeply transformed the way we conduct both business and everyday life and they continue to do so. The ability to ensure confidentiality and integrity of information sent over digital channels is fundamental to this development and is absolutely essential for all private and corporate communication, ranging from bank transactions, digital citizen services, and remote computer access, to cell phone calls and instant messaging. The vast majority of secured data sent over all types of networks is encrypted using so-called symmetric ciphers. The security of our digital infrastructure thus rests at its very base on their security.

The central topic of this thesis is the security of block ciphers – the most prominent form of symmetric ciphers. This thesis is separated in two parts. The first part is an introduction to block ciphers and their cryptanalysis, the second part contains publications written and published during the PhD studies. The first publication evaluates the security of a modification of the AES in which the choice of S-box is unknown to the attacker. We find that some of the attacks that can be applied to the AES can be transferred to this block cipher, albeit with a higher attack complexity. The second publication introduces a new block cipher family which is targeted for new applications in fully homomorphic encryption and multi-party computation. We demonstrate the soundness of the design and its superior performance in these applications. The third publication treats the cryptanalysis of Simon, a cipher proposed by the NSA. In particular we discuss how the methods of differential and linear cryptanalysis can correctly be applied to ciphers of this type. The fourth publication introduces a cryptanalytic framework which generalizes differential cryptanalysis. We demonstrate that attacks based on impossible transitions in this framework can competitively break round-reduced block ciphers in the low-data setting.

Resumé

Den hurtige og fortsatte udvikling af beregningsenheder og den udbredte overgang til digital kommunikation har fundamentalt ændret den måde, hvorpå vi gør forretning og begår os i hverdagen. Evnen til at sikre fortrolighed og integritet af den information, vi sender via digitale kommunikationskanaler, er fundamental for denne udvikling og er absolut essentiel for al privat og forretningsmæssig kommunikation, spændende over alt fra bankoverførsler, digitale offentlige tjenester og fjernadgang til computere, til mobiltelefonsamtaler og chatbeskeder. Langt størstedelen af sikret data, der bliver sendt over alle typer netværk, er krypteret med såkaldte symmetriske ciphers. Vores digitale infrastrukturs sikkerhed hviler således grundlæggende på sikkerheden af disse.

Hovedemnet for denne afhandling er sikkerheden af block ciphers – den mest anvendte type af symmetriske ciphers. Afhandlingen består af to dele. Den første del er en introduktion til block ciphers og kryptoanalyse af disse, mens den anden del består af publikationer skrevet og udgivet under Ph.d.-studierne. Den første publikation undersøger sikkerheden af en modificeret udgave af AES, hvor valget af S-box ikke er kendt af angriberen. Vi konkluderer, at nogle angreb på AES kan overføres til denne block cipher, dog med en forhøjet angrebskompleksitet. Den anden publikation introducerer en ny type block cipher, som er målrettet nye anvendelser inden for fuldstændigt homomorfisk kryptering og multi-party beregninger. Vi demonstrerer designets korrekthed og overlegne ydeevne i disse anvendelser. Den tredje publikation omhandler kryptoanalyse af Simon, en block cipher udviklet af NSA. Navnligt diskuterer vi, hvordan differential og linear kryptoanalyse korrekt kan anvendes på denne type cipher. Den fjerde publikation introducerer en type kryptoanalyse som generaliserer differential kryptoanalyse. Vi viser, at angreb baseret på umulige overgange i denne type kryptoanalyse kan bruges til vellykket at angribe block ciphers.

Zusammenfassung

Rasante Fortschritte in der Computerentwicklung und die umfassende Ausbreitung digigitaler Kommunikationsmittel haben tiefgreifende Veränderungen sowohl im Wirtschafts- als auch im Alltagsleben nach sich gezogen, die unvermindert andauern. In dieser Entwicklung spielt die Fähigkeit, die Vertraulichkeit und Unversehrtheit von Information, die über digitale Kanäle gesendet wird, zu gewährleisten, eine Schlüsselrolle. Sie ist grundlegend für jegliche private und kommerzielle Kommunikation, sei es Onlinebanking, Fernwartung, Handytelefonante oder Chatprogramme. Die große Mehrheit der gesicherten Daten, die über alle Arten von Netzwerken geschickt werden, wird verschlüsselt mithilfe von sogenannten symmetrischen Chiffren. Somit bildet die Sicherheit dieser Chiffren das Fundament der Sicherheit unserer digitalen Infrastruktur.

Den zentralen Gegenstand dieser Dissertation bildet die Sicherheit von Blockchiffren – die bedeutendste Art von symmetrischen Chiffren. Diese Arbeit ist in zwei Teile aufgeteilt. Den ersten Teil bildet eine kurze Einführung zu Blockchiffren und ihrer Kryptanalyse. Der zweite Teil enthält Veröffentlichungen, die im Rahmen der Promotion entstanden und veröffentlich wurden. Die erste Veröffentlichung untersucht die Sicherheit einer Abwandlung des AES, in dem die Spezifierung der S-Box dem Angreifer unbekannt ist. Wie sich zeigt, lassen sich einige der Angriffe auf AES auch auf diese Variante übertragen, allerdings zulasten des Zeitaufwandes des Angriffs. Die zweite Veröffentlichung führt eine neue Familie von Blockchiffren ein, die speziell für den Einsatz in Protokollen, die sicheres verteiltes Rechnen oder vollständig homomorphe Verschlüsselung benutzen, entwickelt wurde. Wir zeigen, dass diese Chiffrenfamilie Sicherheit bei gleichzeitig höherer Leistungsfähigkeit in diesen Protokollen möglich macht. Die dritte Veröffentlichung behandelt die Kryptanalyse der NSA-Chiffre Simon. Insbesondere wird diskutiert, wie die Methoden der differentiellen und linearen Kryptanalyse auf Chiffren dieser Art korrekt angewandt werden können. Die vierte Veröffentlichung führt einen neuen Typus von Kryptanalyse ein, der eine Verallgemeinerung der differentiellen Kryptanalyse darstellt. Es wird gezeigt, dass Attacken, die auf unmöglichen Übergängen in dieser erweiterten Struktur fußen, beim Brechen rundenreduzierter Blockchiffren mit geringem Datenaufwand konkurrenzfähig sind.

Acknowledgements

I truly enjoyed my time as a Ph.D. student. I am aware that this is to the smallest part my own accomplishment and I would like to take this opportunity to express my gratitude to those who made this such a pleasant experience.

First of all, I would like to thank my supervisor, Christian Rechberger who was always available for guidance and support while leaving me freedom to pursue research ideas of my own. Thank you for your encouragement and patience. I would also like to thank Lars Ramkilde Knudsen, my co-supervisor, for managing a great research group and even more so for being a source of inspiration, having an open door, and putting things into perspective.

This gratitude naturally extends to all current and former members of our research section: Mohamed Ahmed Abdelraheem, Martin R. Albrecht, Hoda A. Alkhzaimi, Subhadeep Banik, Andrey Bogdanov, Christina Boura, Christian D. Jensen, Stefan Kölbl, Martin M. Lauridsen, Christiane Peters, Arnab Roy, Elmar Tischhauser, and Philip Vejre. All of you helped to make this a wonderful and inspiring research group to be in. I particularly would like to thank Martin and Stefan for their humor, endless discussions, distractions, enthusiasm, coffee, and beers.

I also thank Joan Daemen and Vincent Rijmen for joining the defense committee and Christian D. Jensen for chairing it.

I would like to thank Gregor Leander for his hospitality and advice and Thorsten Kranz and Christoph Beierle for collaboration and workshop co-organization. My thanks also go to Kaisa Nyberg and Céline Blondeau for a great month at Aalto University. I would furthermore like to thank all of my collaborators, coauthors and the many great people in the crypto community who I had the pleasure to meet in the past years.

My thanks also go to Martin Merker and Thomas Perrett for never failing to distract me with graph theory problems over a good cup of coffee. To my friends in Germany, thank you for your support, advice, tolerance, and affection.

My deep gratitude goes to my parents for their guidance and encouragement and to my brother and my sister. Finally and most of all, I would like to thank my wife, Imke, for her enduring patience, support, and love.

Contents

0 Introduction

The development and rapid evolution of digital computers is without doubt the technological advancement that most deeply changed modern life. The exponential growth in computing power together with the reduction of size and power consumption of computational devices has led us from room filling machines to computers that easily fit in our pocket while providing computing power unparalleled by the former machines. While this could certainly have been a major source of change by itself, it was the pervasion of communication networks and advances in communication technology that allowed computers to have the impact that we see today – most importantly the creation of the Internet and mobile telephone networks.

It is no coincidence that this development was accompanied by the foundation and growth of modern cryptology. The central goal of cryptology is to provide methods that enable secure communication over insecure channels. Incidentally, the channels of modern communication are often inherently insecure both due to the nature of the transportation medium, such as radio waves, and the disparate ownership structure of these networks. It is thus that cryptology came to be a cornerstone of today's communication infrastructure making services such as electronic bank transactions, remote access, and private messaging possible.

Classically there are three characteristics of secure communication: confidentiality, integrity, and authenticity. Confidentiality refers to the inability of an eavesdropper to infer any information contained in the communication. Integrity describes the assurance that the information has not been modified during transmission. Authenticity is the ability to prove the source of a certain information.

A method or algorithm that provides confidentiality of information is called a cipher. Using a cipher, a sender can encrypt information and send the encrypted information to a recipient. The recipient can then again use the cipher to decrypt and receive the original information. To avoid an eavesdropper decrypting the message, the exact method used to encrypt the information must be secret. This is commonly achieved be initializing the cipher with a secret key. A cipher thus more accurately corresponds to a family of encryption and decryption methods.

Generally speaking, there are two types of ciphers. The first and older kind are the symmetric ciphers. With these, the same key is used for both encryption and decryption. The second type are the asymmetric ciphers — also called public-key ciphers. With these, decryption requires a different key than encryption. Importantly, they are constructed such that knowledge of one key does not grant knowledge of the other key. While asymmetric ciphers generally have more powerful applications and enable solutions in situations where symmetric ciphers fail, symmetric ciphers tend to have considerably faster implementations. In practice, one often combines both

types to benefit as well from the speed of symmetric ciphers as from the versatility of asymmetric ones.

Both symmetric and asymmetric ciphers can be used to achieve the two other characteristics of secure communication: integrity and authenticity. Symmetric ciphers are used here in a construction called message authentication code (MAC). The corresponding construction for asymmetric ciphers is called a digital signature scheme.

In recent decades, the scope of cryptology has been broadened to encompass more diverse applications that center generally around the problem of retaining control over some information while at the same time allowing other parties limited access to this information. A good example for this is Yao's millionaires' problem: a group of millionaires wants to determine the richest among them without revealing their exact wealth to each other. This and related problems can be solved with a cryptographic techniques called secure multi-party computation (MPC). Other examples of such techniques include fully homomorphic encryption (FHE) which allows an external party to do meaningful computation on encrypted data without gaining knowledge of either the data or the unencrypted result of the computation, and verifiable computation where a party can prove to another party that it indeed accurately performed some desired computation.

We will concentrate here on the oldest branch of cryptology: symmetric ciphers. There are two types of symmetric ciphers: stream ciphers and block ciphers. While stream ciphers encrypt messages as a whole, block ciphers only encrypt one fixed-size piece of information at a time. Block ciphers hence require a message to be separated into pieces of this size. To avoid security problems which can arise when every piece of information is encrypted independently of the others — in particular when the same piece of information is always encrypted to the same encrypted piece — block ciphers should always be executed in a so-called mode of operation. This is an embedding structure which connects the encryptions of the single message pieces to create dependencies between these. These modes also allow block ciphers to act like stream ciphers. Here we will only concern ourselves with block ciphers.

0.1 Overview of the thesis

This is a thesis by publication which consists of two parts. In the first part, we give a concise introduction to block ciphers and their cryptanalysis. Topics covered include different notions of block cipher security, some general design strategies, and important cryptanalytic techniques such as differential and linear cryptanalysis. For more details and background references, [21] is a good source.

The second part of the thesis contains a collection of publications which treat various aspects of block cipher security. In the first publication, a variant of the Advanced Encryption Standard is considered in which an integral building block — the S-box — is considered to be unknown to the attacker. It is demonstrated that integral cryptanalysis can still be used to attack up to six rounds of this cipher,

recovering both the key and the secret S-box.

In the second publication, a new family of ciphers is introduced which is optimized for deployment in fully homomorphic encryption or multi-party computation schemes. In such schemes, the non-linear operations of a cipher pose the largest bottleneck. This new cipher family is thus designed from ground up with the goal of minimizing its non-linearity. The publication includes a cryptanalytic evaluation of the scheme as well as an implementation comparison with existing schemes. It should be mentioned that there exist attacks which can break some members of the cipher family with less time complexity than exhaustive search [12]. An updated version of this publication will soon be publicly available.

The third publication analyzes the type of round functions used in the block cipher Simon, published by the NSA. In particular, exact and efficient methods to determine the linear and differential round transition probabilities are derived. Those are then used to find optimal differential and linear trails using a computer-aided approach. Further building up on this, the parameter space of Simon is explored with regard to security against these attack vectors.

The fourth publication introduces a framework which generalizes the methods of differential cryptanalysis. It is shown that impossible transitions in this framework can be used to successfully attack round-reduced versions of the Advanced Encryption Standard and the Data Encryption Standard with low data complexity. Furthermore it is shown that higher-order differentials correspond to the generalized form of truncated differentials in this framework.

Part I

Introduction to block ciphers and their security

1 Block Ciphers

In this chapter, we will introduce the concept of a block cipher, discuss the different notions of block cipher security, and take a look at some basic design strategies.

1.1 Notations and conventions

We will denote the finite field with n elements as $\mathbb{F}_n$ or as $\mathbb{F}_{p^m}$ where p is a prime number and the characteristic of the field. We will denote the logarithm to the base 2 as log. Random variables will be written with bold capital letters, e.g., $\mathbf{X}$. When a probability distribution is not explicitly mentioned, a uniform distribution is assumed. The size (or cardinality) of a set A is denoted as $|A|$. Let $A \subseteq X$. When the superset X is clear from the context we will denote the complement $X \setminus A$ of A in B as $\overline{A}$.

1.2 Basic definitions

Conceptually a block cipher is some method that allows us to encode a fixed-size piece of information — such as a word of fixed length in some alphabet — using a secret piece of information, a key. The hope for any well-designed block cipher is that it should not be possible to extract the original information from the encoded word without knowledge of the secret key. We start be defining the basic concepts of block ciphers.

A *block cipher* is a family of bijective functions Enc_k parametrized by a *key* $k \in \mathcal{K}, |\mathcal{K}| < \infty$ that map a finite set of messages $\mathcal{M}$ to a finite set of encrypted messages $\mathcal{C}$:

$$\mathrm{Enc}_k : \mathcal{M} \to \mathcal{C}.$$

The input to a block cipher is called the *plaintext*, the output is called the *ciphertext*. These are usually denoted as m and c. Accordingly $\mathcal{M}$ and $\mathcal{C}$ are called the *plaintext space* (or *message space*) and the *ciphertext space*. $\mathcal{K}$ is called the *key space*. The function Enc_k is called the *encryption* function, its inverse $\mathrm{Dec}_k := \mathrm{Enc}_k^{-1}$ is called the *decryption* function.

Usually $\mathcal{M}$, $\mathcal{C}$, and $\mathcal{K}$ will have a group structure e.g., $\mathbb{F}_p^n$ or $\mathbb{F}_{p^n}$. The most common choices are the binary representations $\mathbb{F}_2^n$ and $\mathbb{F}_{2^q}^m$. The elements of these can nicely be associated with bit-strings of length n or qm respectively and are thus very amenable for implementation in computers. Due to this, we define the *block size* n of a cipher as $n := \lceil \log |\mathcal{M}| \rceil$ and likewise the *key size* l as $l := \lceil \log |\mathcal{K}| \rceil$. In this thesis we will only be concerned with bit-based ciphers.

1.3 Notions of security

We expect from a secure cipher that it is not possible for an adversary to infer the plaintext from the ciphertext without knowledge of the secret key. Let us assume an attacker has collected some amount of ciphertext. Usually the attack will have some a priori knowledge about the plaintext. This could for example be the language or some format that the plaintext will be in. The attacker could then always try out all possible keys until one of the keys returns a plausible message.

Such an attack, in which the adversary simply tries out all keys, is called an *exhaustive search* or a *brute-force attack*. Unless the adversary has absolutely no knowledge about the plaintext or is given only a very limited amount of ciphertext, such an attack is always possible. Brute-force attacks thus pose a natural reference for the quality of any attack.

Of course to apply an exhaustive search, the attacker must know the cipher and the possible keys. To thwart this type of attack one could simply try to keep the block cipher itself a secret. While this has been done in the past, and sometimes continues to be done, Auguste Kerckhoffs famously formulated in 1883 [17] what is nowadays a generally accepted principle:

Kerkhoffs' principle. *The security of a cipher should solely rely on the secrecy of the key and never on the secrecy of the encryption method.*

Adhering to this principle will ensure that even an adversary with full knowledge of the encryption method, will not be able to break the cipher. Today this principle is often even taken a step further: it is considered best practice to make the design of a block cipher public. While this removes any secrecy of the method, it enables the large community of academic, industrial, governmental, and leisure cryptanalysts to scrutinize the design. This either leads to broken designs, which should not be used in the first place, or to an increased level of trust in the security of the cipher.

We will now start to take a closer look at the components that together define the different notions of block ciphers security.

1.3.1 The adversary

When we want to define what a secure block cipher is, the first question that we have to answer is "Secure against whom?". What are the abilities that we need to assume an adversary can potentially have and use to break an encryption scheme?

For being able to attack the cipher, the adversary needs access to some data. The extent to which the adversary is granted access defines four different main types of attack scenarios.

Definition 1.1 (Ciphertext-only attacks)**.** The adversary is only given access to a number of ciphertexts.

Definition 1.2 (Known-plaintext attacks)**.** The adversary is given access to a number of corresponding plaintext-ciphertext pairs.

Definition 1.3 (Chosen-plaintext attacks)**.** The adversary can choose one set of plaintexts for which she will be given the corresponding ciphertexts.

Definition 1.4 (Adaptively chosen-plaintext attacks)**.** The adversary has access to the encryption of any plaintext of her choice during the whole attack.

Clearly each attack type gives the adversary more power then the previous attack types. While especially the chosen-plaintexts attacks might seem to be giving the adversary unrealistically much access, there are practical scenarios in which attacks of this type are possible. Furthermore, as security against chosen-plaintext attacks is a stronger security notion, we might want our ciphers to be secure against even such an adversary.

For the last two types of attacks, there exist the corresponding ciphertext versions: chosen-ciphertext and adaptively chosen-ciphertext attacks. These types of attacks can also be combined to give the strongest types of access: chosen-plaintext chosen-ciphertext attacks and the corresponding adaptive versions.

Block ciphers are finite objects and as such they can always be attacked just given enough time and data. As mentioned above, it is for example always possible to find the key by exhaustive search. To make meaningful statements about the security of a block cipher, it is hence necessary to state what the maximally allowed time and the maximally allowed amount of data are. The natural upper bound for the allowed time — the so-called time complexity — is the time needed to exhaustively try out the whole key space. The natural upper bound for the maximally allowed knowledge of data — the so-called data complexity — is the whole codebook, i.e., all possible plaintext-ciphertext pairs.

The data complexity can directly be measured in allowed plaintext-ciphertext pairs. For the time complexity it is not so straightforward. The usually taken and arguably most objective method of determining the time complexity of some attack is stating it in relation to the time that the encryption of one plaintext would take. Unfortunately this relation depends on the concrete computing machine (computing model) and the complexity of an attack in relation to the complexity of an encryption can vary depending on this.[1] One can argue though that while the concrete machine used for an attack can make a difference of a small factor the general order of magnitude of an attack's time complexity should not deviate too much.

A third resource that an attacker needs and whose amount is often not stated in concrete attacks is the allowed memory capacity. Whereas a high time complexity can often be handled by parallelizing the attack and distributing it to a large number of computers, a high memory complexity cannot so easily be dealt with and can pose the bottleneck in practical attack scenarios. High memory complexity can also have a large effect on the time complexity as the access time increases considerably when moving from cache to RAM to disk storage.

[1] A good example are modern CPUs that have dedicated processor instructions for encryptions with the block cipher AES, making encryptions relatively cheaper in comparison to older CPUs where AES encryptions could not utilize these instructions.

1.3.2 Attack goals

To define the security of a cipher, we need to both define the power and limitations and the goal that the attacker tries to achieve. We will now introduce a number of different possible attack goals in decreasing order of difficulty for the attacker.

Definition 1.5 (Key-recovery attack)**.** In a key-recovery attack, the attacker's goal is to retrieve the key.

As said before it is (almost)[2] always possible to find the key by exhaustive search. A key-recovery attack would thus only be considered a break of the cipher's security if it has a time complexity below that of a brute-force attack. Of course it is principally possible for the designers of a cipher to explicitly state lower security claims.

Definition 1.6 (Global deduction attack)**.** In a global deduction attack, the goal of the attacker is to find an efficient method for decrypting arbitrary ciphertexts.

While this is a weaker attack goal, as we do not require the attacker to recover the key itself, a method that can decrypt arbitrary ciphertexts is usually equally devastating for the security of a cipher. For a well-designed cipher we would hope that this should not be possible for any ciphertexts which gives rise to the third possible attack goal.

Definition 1.7 (Local deduction attack)**.** In a local deduction attack, the goal of the attacker is to find a correct plaintext-ciphertext pair different from those that she has been given directly.

While it is intuitively clear that we would want a good cipher to be resistant against the above mentioned attack goals, for the next goal it is not initially clear why susceptibility to it can pose a security risk.

Definition 1.8 (Distinguishing attack)**.** In a distinguishing attack, the attacker is given access to both the cipher with a uniformly randomly chosen key and to a function that has been chosen uniformly at random from all invertible mappings from the plaintext space to the ciphertext space. The goal of the attacker is then to determine which of the two is the cipher and which is the random function.

Intuitively it is a desirable property of a cipher to look like a random function. After all this is what we would expect from the ciphertexts: they should look random. Although this type of attack looks like it is mostly of theoretical interest, we will later see that ciphers which fail to be indistinguishable from random permutations are often also vulnerable to the other attack goals (see Section 2.1).

The counterpart to the last type of attack goal, the distinguishing attack, is an idealized block cipher:

[2] Exceptions are scenarios in which the attacker simply lacks enough information to deduce the key, for example due to a lack of data, a lack of knowledge about the plaintext distribution in a ciphertext-only attack, or the existence of equivalent keys.

Definition 1.9 (Ideal cipher)**.** An ideal cipher for a message space $\mathcal{M}$, a ciphertext space $\mathcal{C}$ and a key space $\mathcal{K}$ is a family of functions indexed by $k \in \mathcal{K}$ where each function is chosen independently and uniformly at random from all bijective $\mathcal{M}$ to $\mathcal{C}$.

An ideal cipher tries to capture the intuitive notion of what we would like a cipher to behave like. It is important to note that the ideal cipher is not a block cipher according to our definition. It corresponds rather to the set of all possible block ciphers (for given $\mathcal{M}$, $\mathcal{C}$, and $\mathcal{K}$) endowed with the uniform probability distribution. It thus inherently lives in a probability space — no concrete instantiation can ever be an ideal cipher. The best we can hope for is thus that a good concrete design is indistinguishable from an ideal cipher, i.e., an adversary trying to achieve a distinguishing attack will only succeed with a probability very close to one half.[3]

We now have a range of different types of adversaries and attack goals that together create different security notions. Combining the strongest type of adversary, the adversary with adaptive chosen-ciphertext chosen-plaintext access, with the weakest type of attack, the distinguishing attack, creates the strongest notion of security for a block cipher (among the notions discussed here). Today we expect a good cipher to be secure in any of the notions that one can construct with the adversaries and goals described here.

We should also mention that there exists a range of other types of adversaries and attack goals that give rise to other interesting security notions. It is for example possible to give the adversary different computational restrictions for the offline phase, the phase where she does not yet have access to the cipher, than for the online phase. This distinction is essential for example in the classical time-memory trade-off (see Hellman [15]). It is also possible to give the adversary access to two versions of the cipher with different keys where the adversary can choose the difference between the keys, allowing for so-called related-key attacks (see for example Biham [2] or Knudsen [18]). An attacker that can restrict the cipher to only choosing its key from a subset of all keys, creates the notion of weak-key attacks (see for example [26]). An interesting variation of the attack goal are multi-target attacks where the adversary has access to a collection of block ciphers and the goal is to break one of them.

Shannon's confusion and diffusion

Shannon described in his fundamental work [32] two properties that ciphers should have to help frustrate attacks: confusion and diffusion. By *diffusion*, Shannon referred to the ability of the cipher to make any statistic which is simple to detect in the plaintexts difficult to detect in the ciphertexts and vice versa. Today we might rephrase this as that a good cipher should behave like a

[3] As it is often the case in symmetric cryptography, it is not possible to define "very close to one half" precisely without making an arbitrary choice of constant. In theoretical cryptography, idealized constructs usually have a size parameter that can tend to infinity allowing limit definitions as common in complexity theory. As concrete block ciphers are always finite, this is unfortunately not possible here.

random permutation. By *confusion*, Shannon referred to the feature of a cipher that key recovery remains difficult, even when the adversary is given a large number of known plaintext-ciphertext pairs. Today we might say that a cipher should be secure against key recovery attacks under known-plaintext attacks. While this is an integral requirement for ciphers nowadays, before Shannon the security of many encryption schemes would break down under known-plaintext attacks.

1.4 General block cipher design considerations

From the general notions of block cipher security, let us now move to considerations for practical block cipher constructions. Let us start the discussion with a quote from Claude Shannon from the article that is often credited with founding modern cryptography in which he describes what a good cipher design should ideally achieve:

> It is not enough merely to be sure none of the standard methods of cryptanalysis work — we must be sure that no method whatever will break the system easily. This, in fact, has been the weakness of many systems; designed to resist all the known methods of solution, they later gave rise to new cryptanalytic techniques which rendered them vulnerable to analysis. (Claude E. Shannon, 1949 [32, p. 704])

Despite the fact that almost 70 years have passed since Shannon's work and despite the many considerable advances that have been made in symmetric cryptology, the described situation has not changed: in lack of any method to ensure that an efficient cipher design is secure against all possible attacks our best option of determining a cipher's security is still to ensure that the cipher is secure against all known attack vectors.

Before we discuss such attack techniques on block ciphers just yet, let us review some general design decisions that have proven themselves as good starting points for cipher design. A first property that is certainly desirable is that a cipher should make any simple relationship between plaintexts too complicated to be detected in the ciphertexts. Intuitively we would like the cipher to thoroughly mix the possible messages to conceal such relations.[4]

Without discussing the role of the keys yet, how can we build a function that achieves such a mixing property? Certainly if we randomly picked a permutation, it would be a good mixing function. But any such function would be impossible to describe, let alone to implement. We thus need some method to build such mixing function that is viable. Shannon [32] suggested to take the composition (product) of many simple functions to create a complex one. Intuitively this makes sense: just

[4] While we seem to have some intuitive idea of what such a well-mixing (pseudo-random) function is, it seems extremely difficult to give a sensible, formal definition. As not uncommon in cryptology we will let our intuition guide us.

as with shuffling cards, repeating a large number of mildly random operations, we expect to get quite thoroughly random results.

How then do we get the key into play? To achieve security against known- or chosen-plaintext attacks, we would like the relationship between the key, plaintext and ciphertext to be as complex as possible while maintaining an efficient construction. We can utilize the same approach that we already use to achieve a good mixing: By blending the key with the message during or between application of the simple functions, the key will be intermingled with the message in a complex way. The repeated application of these simple functions can thus not only be used to hide the relationship between the plaintext and the ciphertext but also to hide the relationship between the plaintext, the ciphertext, and the key.

All modern block ciphers follow this concept which gives rise to the following definition: a *product cipher* is a block cipher that consists of the repeated composition of functions:

$$\mathrm{Enc}_k = f^r_{k_r} \circ \cdots \circ f^1_{k_1},\ r > 1,$$

where the keys for the smaller functions are derived from the general key k via a mapping called *key schedule*, $k_i = \mathrm{KeySchedule}_k(i), 1 \leq i \leq r$. The functions $f^i_k, 1 \leq i \leq r$ are called *round functions* and the respective keys k_i are called *round keys*. If the functions f^i are identical (or almost identical), such a cipher is also referred to as an *iterated* block cipher. The intermediate results between rounds are either called intermediate messages or states. As we will only be concerned with product ciphers, we will implicitly assume for the rest of this thesis that block ciphers are of this kind.

The difficulty of designing good product ciphers lies then in determining suitable round functions, a good method of blending the key into the message, and the number of rounds needed to achieve a sufficiently complex relationship between plaintext, ciphertext, and key. In the following, we will introduce the two most prominent design classes.

1.5 Substitution-permutation networks

One possibility of constructing good round functions is this: to avoid the expensive implementation cost of good mixing operations on the whole message, we chop the message into smaller parts, apply good mixing operations on each part, and then use a cheap mixing operation to mingle all parts again. This corresponds to two different mixing layers called substitution layer and permutation layer. A cipher which is constructed as a sequence of substitution and permutation layers is called a *substitution-permutation network* (abbreviated as SPN).

In the substitution layer, the state is separated into smaller segments — usually all of the same size. Each of the segments is then substituted independently of the other segments according to some rule. Usually this is done via a so-called *S-box* ("S" as in substitution) which is simply a look-up table. Depending on the design, the same S-box might be used for all segments, or different S-boxes can be employed. Choosing

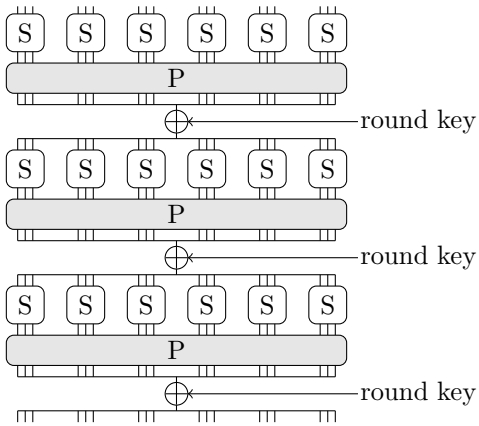

Figure 1.1: A schematic of a substitution-permutation network. S-boxes are denoted by S, the permutation layers are denoted by P.

good S-boxes is a large field of research in itself but as they are often the only source of non-linearity in the cipher, good non-linear properties are usually desirable.

The permutation layer operates on the entire state. As such it is desirable to keep the complexity of this layer low to achieve an efficient implementation. In hardware the cheapest implementation for the permutation layer would be a mere reordering of the state bits which corresponds to a rewiring – hence the name permutation layer. In modern ciphers, especially those designed for software implementation, the permutation layer is often more generally a linear or affine transformation.

To mix a round key into the state, many options such as key-dependent S-boxes are available. The most common technique is though to add a key of the same length as the state to the state using either modular or exclusive-or addition This can be done either before both layers, between them, or after them. See Fig. 1.1 for a schematic of a substitution-permutation network.

By far the most important block cipher based on a substitution-permutation network is the Advanced Encryption Standard (abbreviated AES). Originally named Rijndael, it was standardized by the U.S. National Institute of Standard and Technology (NIST) in 2001 [1] after being selected as finalist among fifteen submissions in a two-round standardization process. Due to the importance of the AES and its influence on many other cipher designs we will shortly describe its general structure. For more details on the specification as well as a good background cipher design decision, on the AES by its designers Daemen and Rijmen [9] is a good source.

The AES encrypts messages of sixteen bytes, or 128 bits. It has either ten, twelve, or fourteen rounds, depending on whether the key size 128, 192, or 256 was chosen. Each round consists of a substitution layer and a permutation layer followed by the exclusive-or addition of a round key to the state. In the substitution layer, each byte is substituted according to an S-box — the same S-box for all bytes and rounds.

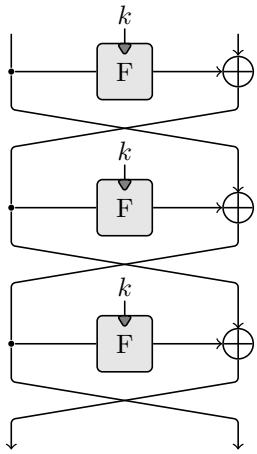

Figure 1.2: A schematic of a Feistel network.

The permutation layer is itself subdivided into two linear layers. To best describe these transformation, we imagine the state of sixteen bytes arranged in a four by four matrix. The first layer simply rotates the rows of the matrix, each row by a different value. The second layer applies a linear transformation to each column of the matrix separately. This linear transformation has been chosen to provide an excellent mixing of the byte values in each column. Together both layers ensure that a very strong mixing is achieved after few rounds of the cipher.

1.6 Feistel ciphers

The basic idea underlying Feistel ciphers, also called Feistel networks, is similar to that of substitution-permutation networks. Namely the idea is to split the state into segments, apply the complex and hence expensive transformations only to segments and use a linear and cheap transformation to mix the segments. In contrast to substitution-permutation networks, Feistel ciphers do not apply the complex transformations to all segments in parallel but apply the transformation only to a subset of segments each round. Furthermore, Feistel ciphers allow non-invertible transformations for the segments.

In the classical Feistel construction, the state is split into two equally sized segments. The first segment is sent through some transformation, usually confusingly also called the (Feistel) round function, and the mixed with the second segment via an exclusive-or addition. Both the result of this addition and the original value of the first segment are kept and used as the two input segments to the next round — only with reversed roles. The key is usually introduced by being added to either the input or the output of the transformation. See Fig. 1.2 for a schematic of a Feistel network.

Some advantages of a Feistel network with respect to a substitution-permutation network are the following: the possibility to use non-invertible round functions giving the designer more freedom of choice, the potentially smaller implementations in hardware both due to the fact that an expensive non-linear function is only applied to a part of the state, and the fact that the same network can be used for decryption, only with an inverted sequence of round keys. But then, as only a part of the state undergoes a non-linear transformation each round, generally more rounds are needed to achieve security than in comparable SPN constructions.

There exist many generalizations of the classical Feistel constructions and Feistel networks can differ in a number of ways. Instead of using same-sized segments, segments of different size may be used. Instead of splitting the state into two segments, more segments can be used. In constructions where more than two segments are used, the number of Feistel round functions used per round of the cipher can differ as well as how these are used to mingle the segments.

The original Feistel cipher

The most influential Feistel cipher is the Data Encryption Standard (abbreviated DES). It is the predecessor of the AES and was standardized in 1977 [10]. The underlying original design was developed by Horst Feistel at IBM [14] from whom the name stems. This first version of the Feistel cipher differed in two ways from the DES and most of todays schemes. Firstly, the switching of segments at the end of each round functions was key-dependent in each bit. So each bit was switched or not switched independently of the other bits depending on the round key. Secondly, the round function, which was build as one round of an SPN, contained a key-dependent choice of S-boxes.

2 Cryptanalysis

In this chapter, we will first give an overview of different types of attack elements and how they can be combined to form more complex attacks. We will then move on to discuss some of the cryptanalytic techniques used in the publications of this thesis.

2.1 Overview of attack types and attack constructions

The simplest and most straight-forward attack which is always available given one or a few pairs of known plaintext and ciphertext is the *exhaustive key search* or *brute-force attack*. To determine the correct key the available ciphertexts are decrypted under all possible keys and any key for which this does not result in the correct plaintexts is discarded. To identify the correct key with high probability, at least around the order of 2^k trial decryptions are needed where k is the size of the key. Because this attack is (almost) always possible, it is used as the reference frame for most other attacks.

To construct an attack which can be executed faster than a brute-force attack, some weakness in the cipher design needs to be used. Central for such cryptanalytic attempts is the fact that block ciphers are round-based. As elaborated in Section 1.4, the whole idea behind using round-based designs is to reach security through the repeated application of round functions that by themselves are not cryptographically secure. Any cryptanalytic attempt will thus start by analyzing the round function and its weaknesses. The difficult part is to determine those weaknesses that create persisting vulnerabilities also in the repeated composition of the round functions.

2.1.1 Attack elements

Successful attacks are often a combination of different attack elements that together can be used to recover the key or distinguish the cipher. We will now briefly describe the major types of such attack elements.

Partial key guessing The importance of a thorough mixing of the whole key with the state is manifested in the partial key guessing technique. Suppose that after some number of rounds there is some part of the output state that, when written as a function of the input bits and the key bits, does not depend on all key bits. We can then determine the value of these key bits by using exhaustive search only on them.

If this technique can successfully be applied, it lends itself to a divide-and-conquer approach to key recovery, strongly reducing the time complexity to determine the full

key. This technique is rarely though applicable to all rounds of a decently designed cipher. Its importance lies much rather in how it can be combined with many other attack elements to construct key recovery attacks.

Deterministic distinguisher A *deterministic distinguisher* is a property of the cipher that holds for any key and would be highly improbable to hold for a random permutation. It is important that such property can be evaluated efficiently to be useful as a distinguisher. Typically such distinguisher could take the form of a linear equation in the ciphertext and plaintext bits. Examples of such distinguishers which we will discuss later are integral properties and truncated differentials.

To every property that always holds, there is the complementary property that never holds. Clearly both are equally good at distinguishing the cipher from a random permutation. It can at times be conceptually easier to work with the impossible property. This is notable for example in impossible differential attacks.

Statistical distinguishers A *statistical distinguisher* is a property that is key-dependent but will for most keys be improbable to hold for a random permutation. A statistical distinguisher can usually be regarded as a random variable over the probability space that results from picking the cipher key uniformly at random. Again it is important that whether or not this property holds must be efficiently determinable.

The important characteristic of a statistical distinguisher is the probability with which it holds. As an exact determination is often infeasible, it is common to make some independence assumptions that allow an easier deduction of this probability. Important examples of statistical distinguishers are differentials, linear approximations, and boomerangs.

2.1.2 Combining attack elements

Powerful attacks are almost always the result of a combination of different attack elements to construct attacks that can cover more rounds or to turn a distinguishing attack into a key recovery attack. We will discuss now some typical combinations of attack elements.

Partial key guessing + a distinguisher To turn a distinguishing attack into a key recovery attack, we can combine a distinguisher with partial key recovery. Let us assume that the distinguishing property is determined from some of the input bits to round 1 and some of the output bits of round r of the cipher. Let us further assume that those output bits of round r can be calculated from the output bits of round $r + s$ and some but not all of the key bits.

An attacker given access to the output bits of round $r + s$ and the corresponding input bits but not the intermediate bits, can now try all key bit combinations needed to determine the intermediate state bits. By discarding all those key bit combinations

that do not give intermediate state bits for which the distinguishing property holds, the attacker can reduce the possible key space, potentially to the point of determining the key exactly.

There are two potential pitfalls in this combination. The first one is, it needs to be made sure that the property is unlikely to hold if the output of round $r+s$ is partially decrypted with a wrong key bit guess. This is commonly known as the *wrong-key randomization hypothesis*. It usually requires that the quality of the distinguishing property does not hold equally well over all rounds but is specific to round r.

The second pitfall is that the constructed attack might not be actually better than an exhaustive key search. Two factors need to be taken into account when determining the time complexity of this combined attack: the time needed to determine the validity of the distinguishing property and the quality of the distinguishing property. The latter determines the probability with which wrong key guesses can be weeded out.

To improve the quality of the attack, the distinguisher can at times be combined with partial key guessing both at the beginning and at the end of the distinguisher, not only at the end.

Partial key guessing + partial key guessing Assume that some part of the state after r rounds only depends on the input bits and some, but not all key bits. Let us furthermore assume that the same partial intermediate state when written as a function of the output bits of round $r+s$ and the key bits, also does not depend on all key bits.

An attacker can now, given access to a number of input texts to round 1 and the corresponding output texts of round $r+s$, mount the following attack. First she creates a list of all possible intermediate states and the corresponding key bit values, determined from the input texts of round 1. Then she creates the corresponding list of intermediate states and key bit values, now determined from the output bits of round $r+s$. To determine potential key candidates, she can look for intermediate states that appear in both lists. To further reduce the number of key candidates, she can test whether the key bits from the first list and the second list are compatible for states that appear in both lists.

This type of attack is called a *meet-in-the-middle attack* and was first applied to 2DES, a concatenation of two DES ciphers using different keys [11]. The time complexity of such an attack is essentially determined by the more expensive one of the two partial key guesses. The downside is that it requires memory to store at least one of the two lists (the other one can be generated and discarded entry by entry). It is often possible though to drastically reduce the memory requirements with a slighty higher time complexity using Floyd's cycle finding method (see for example Morita, Ohta, and Miyaguchi [28]).

Distinguisher + distinguisher At times it is possible to combine two distinguishers to create a longer distinguishing property on a block cipher. For example, if we have two deterministic distinguishers, one on the first r rounds, and another on the

second s rounds, that are contradicting each other in the intermediate states, we can construct an impossible property. This is for example done to construct impossible differentials using two probability-one truncated differentials in the miss-in-the-middle technique.

It is also possible to combine statistical distinguishers. This is for example done in differential-linear attacks [25] or in boomerang attacks [36]. To be able to determine the success probability of these attacks, it is necessary to make additional assumption about independences which are not always satisfied (see for example [29]).

2.2 Differential cryptanalysis

One of the most important cryptanalytic techniques available today is differential cryptanalysis. Here we will give a concise introduction to the basic principles and some important extensions.

2.2.1 Basic differential cryptanalysis

In differential cryptanalysis, the cryptanalyst tries to find a correlation between the difference of two plaintexts and the difference of two ciphertexts. If the correlation is sufficiently stronger than the expected correlation for a random permutation, it may be used in a cryptanalytic attack.

The core idea behind differential cryptanalysis is to choose the difference such that it is unaffected by the way the round keys are introduced into the state. Usually round keys are either added as vectors over $\mathbb{F}_2$ or as elements of $\mathbb{Z}_n$ for some $n \in \mathbb{N}$. More generally we can say that the round key is most often introduced via some group operation.

For the sake of simplicity, we restrict ourselves here to the case where the group operation is addition of vectors over $\mathbb{F}_2$. Most (if not all) definitions can equally be made for a general group addition. This restriction also implies that the texts or states, at which we are looking, lie in $\mathbb{F}_2^n$ for some $n \in \mathbb{N}$.

Definition 2.1 (Differential)**.** A *difference* δ between two messages m_1 and m_2 is defined with respect to the addition over $\mathbb{F}_2^n$ as $m_1 \oplus m_2$. A *differential* is a pair of differences (α, β). These differences denote a difference in the input and a difference in the output for some function.

We write $\alpha \xrightarrow{f} \beta$ to indicate a differential over a function f or just $\alpha \longrightarrow \beta$ when the function is clear from the context. To write the event that a specific pair of texts $(x, x \oplus \alpha)$ follows the differential, i.e. that $f(x \oplus \alpha) \oplus f(x) = \beta$, we write $\alpha \underset{x}{\xrightarrow{f}} \beta$.

Definition 2.2 (Differential probability)**.** Let $\mathbf{X} \in \mathbb{F}_2^n$ be a uniformly distributed random variable. Let $f : \mathbb{F}_2^n \to \mathbb{F}_2^m$. The *differential probability* of $\alpha \xrightarrow{f} \beta$ is defined as

$$\Pr_{\mathbf{X}}\left(\alpha \underset{\mathbf{X}}{\xrightarrow{f}} \beta\right) \tag{2.1}$$

which can also be written as

$$\Pr_{\mathbf{X}}\left(f(\mathbf{X}) \oplus f(\mathbf{X} \oplus \alpha) = \beta\right). \tag{2.2}$$

This probability is also denoted as $\mathrm{DP}_f(\alpha, \beta)$ or $\mathrm{DP}(\alpha, \beta)$ when the respective function is clear from the context.

For a random function (and similarly for a random permutation), the probability of any given differential with non-zero input difference is very low, on average 2^{-n}. If we have for a function f a differential of probability significantly higher than 2^{-n}, we can use this differential to distinguish the function f from a random function (or permutation).

There are two problems that we encounter when we are trying to find a differential suitable for a distinguisher:

1. How do we determine the differential probability when exhaustively trying all text pairs is computationally infeasible?

2. How do we determine the differential probability of functions with secret parameters (e.g. block ciphers with secret keys)?

Let us first discuss a partial solution to the first problem for composed functions.

In a product cipher, it is natural to consider intermediate differences taken between the single rounds. Let now $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be a function generated through the concatenation of r round functions:

$$f = f_r \circ \cdots \circ f_2 \circ f_1. \tag{2.3}$$

Definition 2.3 (Differential trail)**.** An $r+1$-tuple of differences $(\alpha_0, \alpha_1, \ldots, \alpha_r)$ with differences in $\mathbb{F}_2^n$ is called a *differential trail* (or *differential characteristic*).

We will write $\alpha_0 \xrightarrow{f_1} \alpha_1 \xrightarrow{f_2} \cdots \xrightarrow{f_r} \alpha_r$ to indicate the differential trail, or $\alpha_0 \longrightarrow \alpha_1 \longrightarrow \cdots \to \alpha_r$ when the individual functions are clear from the context.

We are now interested in the probability that a pair of texts follows this trail, i.e., takes all the specified intermediate differences.

Definition 2.4 (Differential trail probability)**.** For a function $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$, $f = f_r \circ \cdots \circ f_2 \circ f_1$ and an associated differential trail $(\alpha_0, \alpha_1, \ldots, \alpha_r)$, the *differential trail probability* is defined as

$$\Pr_{\mathbf{X}}\left(\alpha_0 \xrightarrow[\mathbf{X}]{f_1} \alpha_1 \text{ and } \alpha_1 \xrightarrow[f_1(\mathbf{X})]{f_2} \alpha_2 \text{ and } \cdots \text{ and } \alpha_{r-1} \xrightarrow[f_{r-1}\circ\cdots\circ f_1(\mathbf{X})]{f_r} \alpha_r\right) \tag{2.4}$$

where $\mathbf{X}$ is a random variable, uniformly distributed over $\mathbb{F}_2^n$.

We will also write $\mathrm{DTP}_f(\alpha_0, \alpha_1, \ldots, \alpha_r)$ or $\mathrm{DTP}(\alpha_0, \alpha_1, \ldots, \alpha_r)$ to denote this probability.

In most attacks, we have no information about intermediate states of the cipher, so we have no possibility of determining which intermediate differences a pair of texts takes. Why then are we at all interested in differential trails? This is for two reasons. Firstly, it is much easier to determine the probability of a differential trail. Secondly, differential trails allow us to determine a lower bound on the probability of a differential.

When we multiply the probability of a differential with the size of the input space, we obtain the number of ordered text pairs that adhere to the differential. Likewise when we multiply the probability of a differential trail with the size of the input space, we obtain the number of ordered text pairs that adhere to the differential trail. But every pair of texts has to adhere to one and only one differential trail. All differential trails that share the same input and output difference with a differential hence create a partition of the text pairs that adhere to the differential. This allows us to formulate the following proposition:

Proposition 2.5 (Differential composition)**.** *The probability of a differential* $(\alpha_0, \alpha_r$ *over a function* $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$*,* $f = f_r \circ \cdots \circ f_2 \circ f_1$ *is the sum of the probabilities of all differential trails* $(\alpha_0, \alpha_1, \ldots, \alpha_r)$ *that it contains:*

$$\Pr(\alpha_0 \longrightarrow \alpha_r) = \sum_{\alpha_1,\ldots,\alpha_{r-1} \in \mathbb{F}_2^n} \Pr(\alpha_0 \longrightarrow \alpha_1 \longrightarrow \ldots \longrightarrow \alpha_{r-1} \longrightarrow \alpha_r). \tag{2.5}$$

Summing over the probabilities of a subset of all differential trails for a given differential, we thus obtain a lower bound for the probability of this differential.

A note on differentials, trails and characteristics

In the first paper on differential cryptanalysis, Biham and Shamir [4] used the term "characteristic" for what we call a differential trail here. Lai and Massey [24] introduced term differential. The term "differential characteristic" seems to be used first by Langford and Hellman [25]. Daemen [7] introduced the term "differential trail". The phrasing "differential path" is also somtimes used [33]. In this text, we will mostly be using "differential trail" as the most descriptive term.

How do we determine the probability of a single differential trail now? To do this exactly is unfortunately infeasible in most cases. The common way of tackling the problem is to assume that the probability of a differential trail is the product of the single round transitions. Although the single rounds are still key dependent, the round transition probability usually is not as a key addition leaves the difference untouched. While the assumption of round independence is clearly not satisfied in general, experiments suggest that this approximation holds fairly well in many practical cases.

In block cipher cryptanalysis, the assumption of round independence is often deduced in a two-step process. Firstly we assume a model in which the round

independence holds, secondly we explicitly state the assumption that this model is close to the reality. The standard model uses a cipher where the round keys are independent, uniformly distributed random variables. In addition to this, it is required that the cipher has a structural property that ensures that the addition of the round keys is sufficient to make the round transition properties independent. Such ciphers are called *Markov ciphers* (see [24]).

The assumption that the actual differential probabilities of a given cipher with a secret key correspond to the probabilities derived in this model is called the *stochastic equivalence hypothesis*. This hypothesis states that differential probabilities behave under almost all keys as they behave under independent, uniformly random round keys.

In accordance with this, we will implicitly assume here that the probability of a differential trail corresponds to the product of the individual round transition probabilities:

$$\Pr\left(\alpha_0 \xrightarrow{f_1} \alpha_1 \xrightarrow{f_2} \cdots \xrightarrow{f_r \alpha_r}\right) = \prod_{i=1}^{r} \Pr\left(\alpha_{i-1} \xrightarrow{f_i} \alpha_i\right). \tag{2.6}$$

If there now exists a differential $\alpha \longrightarrow \beta$ over r rounds of a cipher that has a probability p which is much larger than 2^{-n}, we can use this differential to distinguish these rounds from a random permutation in a chosen-plaintext attack. By requesting the encryption values of sufficiently many plaintext pairs, we can test whether the output difference corresponds to β as often as predicted by the differential probability. This distinguisher can then be combined with partial key guessing to construct a key recovery attack.

2.2.2 Truncated differentials

Sometimes it can be useful to bundle many differentials together to achieve stronger distinguishers. For example, starting with an input difference α the most probable output difference might not be sufficiently probable to be used in an attack. But by testing for a larger number of the most likely output differences instead of just one, we may be able to find a strong enough distinguisher. The problem hereby is how to efficiently determine whether a difference belongs to this set of most probable output differences. Depending on the size of this set, it might be too difficult to determine membership (or write down this set).

Truncated differentials are such collections of differentials that have a structure that makes it easy to define them and determine membership: a *truncated difference* is defined as an affine subspace of all differences and a *truncated differential* is then defined as a pair of truncated differences. The probability of such a truncated differential is defined as the probability that a pair of texts chosen uniformly at random from all text pairs with a difference in the truncated input difference is mapped to a difference in the output truncated difference.

2.2.3 Impossible differentials

In standard differential cryptanalysis, we try to use differentials that have a sufficiently high probability to distinguish the cipher from a random permutation. Another possibility is to use differentials that have zero probability. Such differentials are termed *impossible differentials*. Clearly it is difficult to use only one impossible differential for distinguishing. But given a slightly larger number, we are already able to distinguish a cipher from a random permutation.

To this end, impossible truncated differentials — truncated differentials of probability zero — form an important class of collections of impossible differentials. This is particularly due to the fact that such impossible truncated differentials can often be constructed using two probability-one truncated differentials in the earlier mentioned miss-in-the-middle technique.

The development of differential cryptanalysis

Differential cryptanalysis was originally introduced at the CRYPTO '90 conference by Biham and Shamir [4]. Lai and Massey [24] introduced the concept of a differential as a collection of differential trails with the same input and output difference. Truncated differentials were devised by Knudsen [20]. Impossible differentials were independently developed by Biham, Biryukov, and Shamir [3] and Knudsen [19].

2.3 Linear cryptanalysis

To introduce the concepts of linear cryptanalysis, we will use an inductive approach which differs somewhat from standard introductions.

Let $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be a bijective mapping on $\mathbb{F}_2^n$. Let $(A, \overline{A})$ and $(B, \overline{B})$ be two partitions of $\mathbb{F}_2^n$ such that $|A| = \left|\overline{A}\right|$ and $|B| = \left|\overline{B}\right|$, i.e., the partitions each split $\mathbb{F}_2^n$ in two parts of equal size. When we choose a random element $\mathbf{X}$ from A, the probability that $f(\mathbf{X})$ is in B clearly is $\frac{1}{2}$ if f is a random permutation. Thus if we choose A and B such that this probability deviates from $\frac{1}{2}$, we can use this to distinguish f from a random permutation by just determining for a sufficient number of random inputs from A the number of their images that are in B. Note that if the probability that a random element from A is mapped to B is p, then the probability that a random element from $\overline{A}$ is mapped to $\overline{B}$ is also p while the respective probabilities for A to $\overline{B}$ and $\overline{A}$ to B are both $p - 1$. Clearly if we know f, it is always possible to find A and B such that this probability is 1.

But now consider the following simple cipher:

$$\mathrm{Enc}_{k_0,k_1}(x) = f(x \oplus k_0) \oplus k_1.$$

For this simple cipher already, our approach fails due to the key additions of k_0 and k_1: we can neither tell when an input to f is in A nor when an output of f is in B.

This is unavoidable. Let us therefore try to do the next best thing: If we choose A and B such that $A + k_0$ is always either A or $\overline{A}$ and $B + k_1$ is always either B or $\overline{B}$, then we know that an element of A is mapped to an element of B always with probability either p or $1 - p$ for a fixed set of keys. If p deviates from $\frac{1}{2}$, it is thus again easy to distinguish Enc_{k_0,k_1} from a random permutation.

As we see in the next lemma, the restriction that $A + k_0$ is always either A or $\overline{A}$ determines the form of the set A to a large degree.

Lemma 2.6. *Let A be a subset of $\mathbb{F}_2^n$ of size 2^{n-1} with the property that for any $k \in \mathbb{F}_2^n$ we have either $A + k = A$ or $A + k = \overline{A}$. Then A must be an affine subspace of dimension $n - 1$.*

Proof. Let $K \subseteq \mathbb{F}_2^n$ be the set of keys k for which $A + k = A$. Clearly 0 is in K. Also for any $k, k' \in K$, $k + k'$ is also in K as $A + k + k' = A + k' = A$. The set K thus forms a linear subspace of $\mathbb{F}_2^n$. Assume without loss of generality that 0 is an element of A (if not we can switch the roles of A and $\overline{A}$). Then $0 + k$ is in A for any $k \in K$ and thus $K \subseteq A$. But also $0 + a \in A$ for any $a \in A$ and hence $A \subseteq K$. Thus $A = K$ and A is a linear subspace of dimension $n - 1$. Then $\overline{A}$ also has to be an affine subspace of dimension $n - 1$. □

From this, we can conclude that the most sensible partition of the space into two equally sized parts is to choose affine subspaces. We can furthermore always set A and B to be the linear spaces (i.e., they include 0 as an element) and set $\overline{A}$ and $\overline{B}$ to the affine halves of the space.

How can we efficiently determine whether a value lies in the space A or $\overline{A}$? We can use the fact that hyperplanes are uniquely described as an orthogonal set with respect to some vector using the inner product. Let $\langle a, b\rangle$ denote the canonical inner product of $\mathbb{F}_2^n$ such that

$$\langle a, b\rangle = \bigoplus_{i=1}^{n} a_i b_i.$$

Then there exists a unique $\alpha \in \mathbb{F}_2^n$ such that $A = \{a \in \mathbb{F}_2^n \mid \langle \alpha, a\rangle = 0\}$. This α is called the *linear mask* associated with A. Knowing α, we can determine whether x is in A or in $\overline{A}$ by evaluating whether $\langle \alpha, x\rangle$ is equal to 0 or 1. A partition of $\mathbb{F}_2^n$ into two affine subspaces of dimension $n - 1$ thus uniquely corresponds to a non-zero linear mask $\alpha \in \mathbb{F}_2^n$ and vice versa.

Let β now be the corresponding vector for the set B. To distinguish the cipher from a random permutation, we have to count how often we have either

$$\langle \alpha, x\rangle = 0 \text{ and } \langle \beta, \mathrm{Enc}(x)\rangle = 0, \tag{2.7}$$

corresponding to an element from A being mapped to B, or

$$\langle \alpha, x\rangle = 1 \text{ and } \langle \beta, \mathrm{Enc}(x)\rangle = 1, \tag{2.8}$$

corresponding to an element from $\overline{A}$ being mapped to $\overline{B}$, when taking random elements x as input to the cipher. If the fraction of x for which this is true is close

to p or $1-p$, we are likely to have been using the cipher. If this fraction is close to $\frac{1}{2}$, we are likely using a random permutation instead. To ease the evaluation, we can combine Eqs. (2.7) and (2.8) to only one equivalent equation:

$$\langle \alpha, x \rangle + \langle \beta, \mathrm{Enc}(x) \rangle = 0.$$

We call such an equation a *linear approximation* of the cipher Enc. The *correlation* of this approximation is defined as $2p-1$. A correlation of 1 then corresponds to all values of A being mapped to B while a correlation of -1 indicates that all values of A are mapped to the complement of B. Our distinguishing ability then only depends on the absolute value of this correlation.

Let us consider a slightly more complex cipher than the previous one:

$$\mathrm{Enc}_{k_0,k_1,k_2}(x) = g(f(x \oplus k_0) \oplus k_1) \oplus k_2.$$

We ignore the key additions for now, i.e., let us assume the keys are equal to 0. Let us suppose that we have three partitions A, B, and C such that the probability from A to B is p over f and the probability from B to C is q. We call such a trail of partitions a *linear trail*. The probability that an element of A is mapped to an element of C corresponds to the sum of the probability that an element of A is mapped first to B and then to C and the probability that it is first mapped to $\overline{B}$ and then to C. As a first guess, we might approximate this probability as $pq+(1-p)(1-q) = 1-p-q+2pq$. This yields a correlation of $2-2p-2q+4pq-1 = (2p-1)(2q-1)$ equal to the product of the correlations of both transitions. Alternatively, if the middle key had switched the transition probabilities now, we would have gotten a probability estimate of $p(1-q)+(1-p)q = p+q-2pq$ which corresponds to a correlation of $2p+2q-4pq-1 = -(2p-1)(2q-1)$. The absolute value of the two-round correlation is thus not influenced by the key additions in this estimate.

Now what assumption did we implicitly make, when we approximated the probability of a particular path being taken as the product of the round transition probabilities? We assumed that those values in B to which an element from A can be mapped are mapped with probability q to C, the same probability that any element from B is mapped to C. So we implicitly assumed that those images from A in B are representative for the whole set, an assumption which seems fair if f were a random function. But then the whole point of the distinguisher is that f is not behaving like a random function.

Imagine we had been using a different middle partition B' with different transition probabilities. Clearly this would have given us a different estimate. So which one is the correct one? As it turns out, when there are no key additions, the exact correlation of the transition A to C is the sum of the correlations of all trails from A to B to C where the middle partitions B can take any value. Taking now key additions into account, the values of the single trail correlations can flip from positive to negative and vice versa. So without knowing the keys, again we are left without knowledge of the two-round correlation.

Similarly to differential cryptanalysis, we have to make an assumption of independence that allows us to estimate the correlation of such a partition. The assumption is that the sign of the correlation of each linear trail is chosen independently of all other trails to be either positive or negative with probability $\frac{1}{2}$. While the expected value of the correlation is then 0, the variance of the correlation scales with the sum of the squares of the correlations of the single trails. It is also this variance that determines the quality of the distinguisher, i.e., the number of needed ciphertext-plaintext pairs needed to distinguish the cipher from a random permutation. As always, whether or not the assumption holds needs to be verified by experiment.

The development of linear cryptanalysis

Linear cryptanalysis was introduced by Matsui [27]. In his attack on DES he only considered linear trails. Nyberg [30] realized that for a correct estimate all linear trails with the same input and output mask have to be taken into account. There exist several extensions of linear cryptanalysis, e.g., multi-dimensional cryptanalysis (see for example [6]). Somewhat corresponding to impossible differentials in differential cryptanalysis is zero-correlation cryptanalysis [5] in linear cryptanalysis.

2.4 Higher order-derivatives and integral cryptanalysis

In this section, we will discuss some techniques based on derivatives over $\mathbb{F}_2$ that can be used to find deterministic distinguishers.

2.4.1 Higher-order derivatives

A function that takes a fixed number of input bits and outputs one bit is called a *Boolean function*. A function that takes a fixed number of bits as input and outputs another fixed number of bits can naturally be represented as a vector of Boolean functions and is hence called a *vectorial Boolean function*. As all relevant block ciphers are bit-based, they can naturally be thought of as vectorial Boolean functions.

One natural way to represent a Boolean function is to write it as a polynomial function in the input bits such as $f(x_1, x_2, x_3) = 1 + x_1 + x_2 + x_2x_3 + x_1x_2x_3$. As the operations are over $\mathbb{F}_2$, we have that $x^2 = x$. So it is common to write the terms such that each variable appears at most once as a factor. The *degree* of a Boolean function is then the largest number of variables in a term that appears in the function.

The *derivative* Δ_α of a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ in the direction of $\alpha \in \mathbb{F}_2^n$ is defined as

$$\Delta_\alpha f(x) := f(x + \alpha) + f(x)$$

where x is the input bit vector. This derivative shares many properties with the standard derivative over the real numbers: it is linear, it satisfies (a variant of) the

product rule and importantly it reduces the degree of the function by at least 1. It is straightforward to extend this definition to vectorial Boolean functions.

A repeated application of the derivative in the direction α_1, α_2, ..., α_t is written as $\Delta_{\alpha_1,\alpha_2,\ldots,\alpha_t} f$. When the α_i are linearly dependent, the derivative always evaluates to 0. When all the α_i are linearly independent, it can be evaluated as

$$\Delta_{\alpha_1,\alpha_2,\ldots,\alpha_t} f = \sum_{\beta \in L(\alpha_1,\alpha_2,\ldots,\alpha_t)} f(x+\beta)$$

where $L(\alpha_1, \alpha_2, \ldots, \alpha_t)$ denotes the linear space spanned by the α_i. To evaluate some derivative of order n of some function at a point x, we thus need the value of the function f at 2^n different points.

The fact that the degree of the function is reduced by 1 for each derivative can be used to construct a distinguisher: if we know that the degree of the cipher that we want to distinguish is d while the degree for a random permutation is much likely higher than d, we can evaluate an order-$(d+1)$ derivative of the function in question and see whether it is equal to 0.

If we write the input bits of a Boolean function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ explicitly out as x_1, x_2, and so forth, we can write the derivative taken in the direction of x_i as Δ_{x_i} which corresponds to the derivative Δ_α where α is all 0s except for one 1 at the ith position. Taking the derivative Δ_{x_i} of a Boolean polynomial, removes all terms from the polynomial that do not contain x_i and removes the variable x_i from all remaining terms. The derivative Δ_{x_1} of the polynomial $1 + x_1 + x_2 + x_2x_3 + x_1x_2x_3$ would thus be $1 + x_2x_3$. We can then easily test whether a given term, say $x_2x_5x_{17}$ is present in a polynomial by taking the derivative for those variables $\Delta_{x_2,x_5,x_{17}}$ and seeing whether a constant 1 is present in the resulting Boolean polynomial. This can easily be done by evaluating the derivative at 0.

For an actual cipher the situation is slightly more complicated. As the cipher takes as input both a plaintext and a key, the value of an output bit is a Boolean function in the plaintext bits and the key bits. Alternatively we could say: whether a particular term in the plaintext bits is present is determined by the key. We could thus write the Boolean function representing an output bit as

$$\sum_{x \in \mathbb{F}_2^n} c_x(k) \cdot x$$

where x represents possible terms in the plaintext bits and $c_x(k)$ is the coefficient that determines whether this term is present or not. This coefficient generally depends on the key k and is by itself again a Boolean function. As said above, we can determine the value of a coefficient function for a cipher with a fixed secret key by evaluating the respective derivative for this term at 0.

For a good cipher, these coefficient functions should be relatively complex in the key. An attack that makes use of coefficient functions that are linear in the key bits is the AIDA [35] or cube attack [13]. If we can find sufficiently many coefficient functions that are linear in the key bits, we can determine the values of these functions

for a cipher with a secret key using derivatives to create a linear system of equations which we can solve for the secret key bits.

Higher-order differentials

Higher-order derivatives were initially developed in 1994 by Lai [23] without a concrete application however. They were then used by Knudsen [20] in 1995 to describe higher-order differentials (which pretty much are higher-order derivatives plus a fixed output value that we are trying to detect) and he demonstrated that these can be used to break ciphers which are secure against standard differential cryptanalysis. Ironically Jakobsen and Knudsen [16] used higher-order differentials again in 1997 to break a cipher designed by Nyberg and Knudsen [31] in 1995.

2.4.2 Integral cryptanalysis

Integral cryptanalysis was initially developed as a dedicated attack against the block cipher Square [8] and later generalized by Knudsen and Wagner [22]. In an integral attack one tries to find a set of input bit positions such that the encrypted values of any set of input values for which the values are fixed in these bit positions and take all possible combinations of values in the other bit positions sums to 0 (at least in some bits).

For the usual case where the sum is taken to be an exclusive-or sum, we can formulate such integral property as follows: there exists a set of input bits and an output bit such that the Boolean function representing the output bit does not possess any term which contains all of these input bits simultaneously. This kind of property is particularly interesting because it will often still exist after a number of rounds for which the degree of the cipher will already be too high for a naive higher-order attack.

The elegance of the original Square attack and the generalized integral attack lies in how the structure of the cipher is utilized to find such property. This approach uses arguments about how certain properties of collections of states propagate through the cipher. Unfortunately this approach is usually only sufficient to attack a few rounds. Recently some progress has been made to find integral properties in cases where the usual toolset of integral cryptanalysis fails [34]. In this new method, a property of a collection of states termed *division property* is used to integrate more structural detail of the cipher into arguing how the property propagates (namely the degree of S-boxes or transitions in general is used).

Bibliography

[1] *Advanced Encryption Standard*. Federal Information Processing Standard (FIPS), Publication 197, U.S. Department of Commerce, Washington D.C. National Institute of Standards and Technology. Nov. 2001.

[2] Eli Biham. "New Types of Cryptanalytic Attacks Using Related Keys". In: *Journal of Cryptology* 7.4 (1994), pp. 229–246.

[3] Eli Biham, Alex Biryukov, and Adi Shamir. "Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials". In: *Journal of Cryptology* 18.4 (2005), pp. 291–311.

[4] Eli Biham and Adi Shamir. "Differential Cryptanalysis of DES-like Cryptosystems". In: *Advances in Cryptology - CRYPTO '90*. Ed. by Alfred Menezes and Scott A. Vanstone. Vol. 537. Lecture Notes in Computer Science. Springer, 1991, pp. 2–21. ISBN: 3-540-54508-5.

[5] Andrey Bogdanov and Vincent Rijmen. "Linear hulls with correlation zero and linear cryptanalysis of block ciphers". In: *Designs, Codes and Cryptography* 70.3 (2014), pp. 369–383.

[6] Joo Yeon Cho, Miia Hermelin, and Kaisa Nyberg. "A New Technique for Multidimensional Linear Cryptanalysis with Applications on Reduced Round Serpent". In: *Information Security and Cryptology - ICISC 2008*. Ed. by Pil Joong Lee and Jung Hee Cheon. Vol. 5461. Lecture Notes in Computer Science. Springer, 2008, pp. 383–398. ISBN: 978-3-642-00729-3.

[7] Joan Daemen. "Cipher and Hash Function Design. Strategies based on linear and differential cryptanalysis". PhD thesis. KU Leuven, Mar. 1995.

[8] Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. "The Block Cipher Square". In: *Fast Software Encryption, FSE '97*. Ed. by Eli Biham. Vol. 1267. Lecture Notes in Computer Science. Springer, 1997, pp. 149–165. ISBN: 3-540-63247-6.

[9] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002. ISBN: 978-3-540-42580-9.

[10] *Data Encryption Standard (DES)*. Federal Information Processing Standard (FIPS), Publication 46, U.S. Department of Commerce, Washington D.C. National Institute of Standards and Technology. Jan. 1977.

[11] Whitfield Diffie and Martin E. Hellman. "Special Feature Exhaustive Cryptanalysis of the NBS Data Encryption Standard". In: *IEEE Computer* 10.6 (1977), pp. 74–84.

[12] Itai Dinur, Yunwen Liu, Willi Meier, and Qingju Wang. "Optimized Interpolation Attacks on LowMC". In: *Advances in Cryptology - ASIACRYPT 2015*. Ed. by Tetsu Iwata and Jung Hee Cheon. Vol. 9453. Lecture Notes in Computer Science. Springer, 2015, pp. 535–560. ISBN: 978-3-662-48799-0.

[13] Itai Dinur and Adi Shamir. "Cube Attacks on Tweakable Black Box Polynomials". In: *Advances in Cryptology - EUROCRYPT 2009*. Ed. by Antoine Joux. Vol. 5479. Lecture Notes in Computer Science. Springer, 2009, pp. 278–299. ISBN: 978-3-642-01000-2.

[14] Horst Feistel. *Cryptographic coding for data-bank privacy*. Research report RC 2827. IBM Research, Mar. 1970.

[15] Martin E. Hellman. "A cryptanalytic time-memory trade-off". In: *IEEE Transactions on Information Theory* 26.4 (1980), pp. 401–406.

[16] Thomas Jakobsen and Lars R. Knudsen. "The Interpolation Attack on Block Ciphers". In: *Fast Software Encryption, FSE '97*. Ed. by Eli Biham. Vol. 1267. Lecture Notes in Computer Science. Springer, 1997, pp. 28–40. ISBN: 3-540-63247-6.

[17] Auguste Kerckhoffs. "La cryptographie militaire". In: *Journal de sciences militaires* IX (1883), pp. 5–38, 161–191.

[18] Lars R. Knudsen. "Cryptanalysis of LOKI91". In: *Advances in Cryptology - AUSCRYPT '92*. Ed. by Jennifer Seberry and Yuliang Zheng. Vol. 718. Lecture Notes in Computer Science. Springer, 1993, pp. 196–208. ISBN: 3-540-57220-1.

[19] Lars R. Knudsen. *DEAL - A 128-bit Block Cipher*. Technical report 151. Submitted as an AES candidate by Richard Outerbridge. Department of Informatics, University of Bergen, Norway, Feb. 1998.

[20] Lars R. Knudsen. "Truncated and Higher Order Differentials". In: *Fast Software Encryption, FSE '94*. Ed. by Bart Preneel. Vol. 1008. Lecture Notes in Computer Science. Springer, 1995, pp. 196–211.

[21] Lars R. Knudsen and Matthew Robshaw. *The Block Cipher Companion*. Information Security and Cryptography. Springer, 2011. ISBN: 978-3-642-17341-7.

[22] Lars R. Knudsen and David Wagner. "Integral Cryptanalysis". In: *Fast Software Encryption, FSE 2002*. Ed. by Joan Daemen and Vincent Rijmen. Vol. 2365. Lecture Notes in Computer Science. Springer, 2002, pp. 112–127. ISBN: 3-540-44009-7.

[23] Xuejia Lai. "Higher Order Derivatives and Differential Cryptanalysis". In: *Communications and Cryptography, Two Sides of One Tapestry*. Ed. by Richard E. Blahut, Jr. Daniel J. Costello, Ueli Maurer, and Thomas Mittelholzer. Kluwer Academic Publishers, 1994, pp. 227–233. ISBN: 978-1-4613-6159-6.

[24] Xuejia Lai and James L. Massey. "Markov Ciphers and Differential Cryptanalysis". In: *Advances in Cryptology - EUROCRYPT '91*. Ed. by Donald W. Davies. Vol. 547. Lecture Notes in Computer Science. Springer, 1991, pp. 17–38. ISBN: 3-540-54620-0.

[25] Susan K. Langford and Martin E. Hellman. "Differential-Linear Cryptanalysis". In: *Advances in Cryptology - CRYPTO '94*. Ed. by Yvo Desmedt. Vol. 839. Lecture Notes in Computer Science. Springer, 1994, pp. 17–25. ISBN: 3-540-58333-5.

[26] Gregor Leander, Mohamed Ahmed Abdelraheem, Hoda AlKhzaimi, and Erik Zenner. "A Cryptanalysis of PRINTcipher: The Invariant Subspace Attack". In: *Advances in Cryptology - CRYPTO 2011*. Ed. by Phillip Rogaway. Vol. 6841. Lecture Notes in Computer Science. Springer, 2011, pp. 206–221. ISBN: 978-3-642-22791-2.

[27] Mitsuru Matsui. "Linear Cryptanalysis Method for DES Cipher". In: *Advances in Cryptology - EUROCRYPT '93*. Ed. by Tor Helleseth. Vol. 765. Lecture Notes in Computer Science. Springer, 1994, pp. 386–397. ISBN: 3-540-57600-2.

[28] Hikaru Morita, Kazuo Ohta, and Shoji Miyaguchi. "A Switching Closure Test to Analyze Cryptosystems". In: *Advances in Cryptology - CRYPTO '91*. Ed. by Joan Feigenbaum. Vol. 576. Lecture Notes in Computer Science. Springer, 1992, pp. 183–193. ISBN: 3-540-55188-3.

[29] Sean Murphy. "The Return of the Cryptographic Boomerang". In: *IEEE Transactions on Information Theory* 57.4 (2011), pp. 2517–2521.

[30] Kaisa Nyberg. "Linear Approximation of Block Ciphers". In: *Advances in Cryptology - EUROCRYPT '94*. Ed. by Alfredo De Santis. Vol. 950. Lecture Notes in Computer Science. Springer, 1995, pp. 439–444. ISBN: 3-540-60176-7.

[31] Kaisa Nyberg and Lars R. Knudsen. "Provable Security Against a Differential Attack". In: *Journal of Cryptology* 8.1 (1995), pp. 27–37.

[32] Claude E. Shannon. "Communication theory of secrecy systems". In: *Bell System Technical Journal* 28.4 (Oct. 1949), pp. 656–715.

[33] Taizo Shirai and Kyoji Shibutani. "Improving Immunity of Feistel Ciphers against Differential Cryptanalysis by Using Multiple MDS Matrices". In: *Fast Software Encryption, FSE 2004*. Ed. by Bimal K. Roy and Willi Meier. Vol. 3017. Lecture Notes in Computer Science. Springer, 2004, pp. 260–278. ISBN: 3-540-22171-9.

[34] Yosuke Todo. "Structural Evaluation by Generalized Integral Property". In: *Advances in Cryptology - EUROCRYPT 2015*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9056. Lecture Notes in Computer Science. Springer, 2015, pp. 287–314. ISBN: 978-3-662-46799-2.

[35] Michael Vielhaber. "Breaking ONE.FIVIUM by AIDA an Algebraic IV Differential Attack". In: *IACR Cryptology ePrint Archive* 2007 (2007), p. 413.

[36] David Wagner. "The Boomerang Attack". In: *Fast Software Encryption, FSE '99*. Ed. by Lars R. Knudsen. Vol. 1636. Lecture Notes in Computer Science. Springer, 1999, pp. 156–170. ISBN: 3-540-66226-X.

Part II

Publications

Security of the AES with a Secret S-box

Publication Information

Tyge Tiessen, Lars R. Knudsen, Stefan Kölbl, and Martin M. Lauridsen. "Security of the AES with a Secret S-Box". In: *Fast Software Encryption, FSE 2015*. Ed. by Gregor Leander. Vol. 9054. Lecture Notes in Computer Science. Springer, 2015, pp. 175–189. ISBN: 978-3-662-48115-8

Contribution

- Main author.

Remarks

This publication has been slightly edited to fit the format.

Security of the AES with a Secret S-box

Tyge Tiessen, Lars R. Knudsen, Stefan Kölbl, and Martin M. Lauridsen
{tyti,lrkn,stek,mmeh}@dtu.dk

DTU Compute, Technical University of Denmark, Denmark

Abstract. How does the security of the AES change when the S-box is replaced by a secret S-box, about which the adversary has no knowledge? Would it be safe to reduce the number of encryption rounds?

In this paper, we demonstrate attacks based on integral cryptanalysis which allow to recover both the secret key and the secret S-box for respectively four, five, and six rounds of the AES. Despite the significantly larger amount of secret information which an adversary needs to recover, the attacks are very efficient with time/data complexities of $2^{17}/2^{16}$, $2^{38}/2^{40}$ and $2^{90}/2^{64}$, respectively.

Another interesting aspect of our attack is that it works both as chosen plaintext and as chosen ciphertext attack. Surprisingly, the chosen ciphertext variant has a significantly lower time complexity in the attacks on four and five round, compared to the respective chosen plaintext attacks.

Keywords: AES, integral cryptanalysis, secret S-box

1 Introduction

The Advanced Encryption Standard (AES) [10] is an iterated block cipher using 10, 12, or 14 rounds depending on the key size of 128, 192, or 256 bits. These variants are named AES-128, AES-192, and AES-256.

In this paper we consider the cipher that is derived from the AES by replacing the S-box with a secret 8-bit S-box while keeping everything else unchanged. If the choice of S-box is made uniformly at random from all 8-bit S-boxes, the size of the secret information increases from $128 - 256$ bits, the key size in the AES, to $1812 - 1940$ bits. Clearly the security level of such a cipher could be very high, thus the question is: Could the number of rounds of this cipher be reduced to fewer than 10 rounds (as in AES-128)?

The AES was designed in order to achieve good resistance against differential and linear cryptanalysis, and this includes the choice of the S-box. Nonetheless a randomly chosen S-box is very likely to be highly resistant against these attacks as well.

The method that is most successful in attacking AES for up to 6 rounds is integral cryptanalysis. Somewhat surprisingly, a variant of this attack also applies to the AES variant with a secret S-box with up to 6 rounds, and although the complexity of the attack is larger than for the attack on the original AES, the time complexity is still less than exhaustive search of a 128-bit key.

Related Work.

The idea of integral cryptanalysis was conceived as a dedicated attack against the block cipher Square [3]. This attack is able to break up to six rounds of AES-128. Biryukov and Shamir applied integral cryptanalysis to a generalised SPN structure denoted SASAS [1], which consists of three substitution layers separated by two affine layers. In their paper, the attacker is assumed not to have any knowledge about the linear layer or the S-boxes which are all allowed to be chosen independently at random. The SASAS attack recovers an equivalent representation of this SPN and thus allows decryption of any ciphertext. The attack allows to break the equivalent of three rounds of AES. It does *not*, however, recover neither the key nor the S-box.

The case of the AES with a secret S-box, which we consider in this paper, lies in between two cases: The original Square attack on one hand can not be directly applied to the case with the secret S-box as it requires knowledge of the S-box to peel off the last layer after guessing some key bits. The SASAS attack, on the other hand, can be used to attack three rounds of this cipher. However, it is not very effective, as the extra knowledge of the linear layer and the equality of all S-boxes remains unused.

The security of PRESENT with a secret S-box was studied by Borghoff et al. in [2] and allows an attack on 28 out of 31 rounds using slightly less than 2^{64} plaintexts. This attack was further improved by Liu et al. in [8]. As the attack depends on the weakness of some randomly chosen 4-bit S-boxes, it seems hard to apply it to the 8-bit S-boxes used in the AES.

Furthermore there are various block cipher designs based on using a secret, key-dependent substitutions like Khufu [9], Blowfish [14], Twofish [15] or Maya [7]. The attack also bears some resemblance to so-called SCARE (Side-Channel Analysis for Reverse Engineering) attacks in which side-channel information is used to recover unknown parts of cipher implementations (see for example [13]).

Our Contributions.

We demonstrate that despite the increased size of the secret information in the cipher, we are able to recover both the secret key and the S-box for the 4-round, 5-round and 6-round versions of AES-128 by building up on techniques from integral cryptanalysis. Our attacks on four and five rounds are practical and achieve almost the same complexity as previous attacks which do not need to recover a secret S-box. The 6-round attack has a complexity of 2^{90} which is already much less than exhaustive search of the key, let alone of the S-box.

Table 1: Results of integral cryptanalysis on AES-128 with a secret S-box, AES-128 and SASAS with AES-like parameters. The time complexity is given in encryption equivalents, the data complexity is given in number of plaintexts/ciphertexts (16 bytes), the memory complexity is given in bytes. We assume that one round of encryption corresponds to 2^5 table lookups.

		Complexity			
Cipher	Rounds	Time	Data	Memory	Reference
SASAS	3	2^{21}	2^{16}	2^{20}	[1]
AES-128 (secret S-box)	4	2^{17}	2^{16}	2^{16}	*This work*
AES-128	4	2^{14}	2^{9}	–	[4]
AES-128 (secret S-box)	5	2^{38}	2^{40}	2^{40}	*This work*
AES-128	5	2^{38}	2^{33}	–	[4]
AES-128 (secret S-box)	6	2^{90}	2^{64}	2^{69}	*This work*
AES-128	6	2^{44}	2^{34}	2^{36}	[6]

Table 1 compares the complexities for our attacks with those of previous integral attacks on AES-128 and the SASAS attack. Interestingly, the time complexities of the 4-round and 5-round attacks are lower by a factor of 2^{11} and 2^{16} respectively in the chosen ciphertext variant as compared to the chosen plaintext variant.

Organisation.

This paper is organised as follows. In §2 the notation and a specification of the AES is given. In §3 we analyse the security of the AES with a secret S-box with respect to statistical and integral attacks. §4 holds the concluding remarks.

2 AES Specification

The AES [10] is an iterated block cipher that operates on 128-bit blocks and comes in three variants: AES-128, AES-192, and AES-256, which have key sizes of 128, 192 and 256 bits, respectively. The number of rounds T is 10, 12, and 14 respectively. The AES uses the four operations `SubBytes`, `ShiftRows`, `MixColumns`, and `AddRoundKey` which are detailed below. We use R_i, $1 \leq i \leq T$, to denote the round function which takes a 128-bit block as input and provides a 128-bit block as output. The ith round is defined as

$$R_i = \begin{cases} \texttt{AddRoundKey}_i \circ \texttt{MixColumns} \circ \texttt{ShiftRows} \circ \texttt{SubBytes} & , \ i < T \\ \texttt{AddRoundKey}_i \circ \texttt{ShiftRows} \circ \texttt{SubBytes} & , \ i = T \end{cases} .$$

Before the first round, a pre-whitening key is used in a step $\texttt{AddRoundKey}_0$, so the T-round encryption with master key K is denoted as

$$E_K = R_T \circ \cdots \circ R_1 \circ \texttt{AddRoundKey}_0.$$

Each of the four operations operate on a 128-bit block arranged in a 4×4 byte matrix:

$$\begin{pmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \end{pmatrix}.$$

The bytes are regarded as elements of what is called the *Rijndael finite field* $\mathbb{F}_{256} = \mathbb{F}_2[x]/(x^8 + x^4 + x^3 + x + 1)$. In the Rijndael finite field, an element is represented by a single byte $a = (a_7 a_6 \cdots a_1 a_0)$ with $a_i \in \mathbb{F}_2$, which in turn represents the field element

$$a(x) = a_7 x^7 + a_6 x^6 + \cdots + a_1 x + a_0.$$

We use hexadecimal notation in typewriter font to write byte values. As such $a = \texttt{01}$ represents $a(x) = 1$, $a = \texttt{02}$ represents $a(x) = x$, and so on. In the following, we briefly describe the four operations used in AES.

2.1 SubBytes

In the `SubBytes` operation, each of the 16 bytes in the state matrix is replaced by another value according to an 8-bit S-box. In the standard AES, the AES S-box is used whose full description is available to the adversary. However, in our analysis we will assume that the S-box is secret and thus unknown to the adversary.

2.2 ShiftRows

In the `ShiftRows` step, the ith row of the state, $0 \leq i \leq 3$, is rotated to the left by i positions. As such,

$$\texttt{ShiftRows}\left(\begin{pmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \end{pmatrix}\right) = \left(\begin{pmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_5 & s_9 & s_{13} & s_1 \\ s_{10} & s_{14} & s_2 & s_6 \\ s_{15} & s_3 & s_7 & s_{11} \end{pmatrix}\right).$$

2.3 MixColumns

In this step, each of the four columns of the state matrix are multiplied from the right onto an invertible matrix M over the Rijndael finite field. The matrix M and

its inverse are

$$M = \begin{pmatrix} \texttt{02} & \texttt{03} & \texttt{01} & \texttt{01} \\ \texttt{01} & \texttt{02} & \texttt{03} & \texttt{01} \\ \texttt{01} & \texttt{01} & \texttt{02} & \texttt{03} \\ \texttt{03} & \texttt{01} & \texttt{01} & \texttt{02} \end{pmatrix} \quad \text{and} \quad M^{-1} = \begin{pmatrix} \texttt{0e} & \texttt{0b} & \texttt{0d} & \texttt{09} \\ \texttt{09} & \texttt{0e} & \texttt{0b} & \texttt{0d} \\ \texttt{0d} & \texttt{09} & \texttt{0e} & \texttt{0b} \\ \texttt{0b} & \texttt{0d} & \texttt{09} & \texttt{0e} \end{pmatrix}.$$

2.4 `AddRoundKey`

In this step, a 128-bit round key is added to the state using the XOR operation. The $T+1$ round keys, denoted $RK_0, \ldots, RK_T$ are generated using the AES key schedule. A brief description of the AES key schedule can be found in Appendix A.

3 Cryptanalysis of the AES with a Secret S-box

3.1 Differential and Linear Cryptanalysis

First, we consider the security of the AES with a secret S-box which is chosen uniformly at random against the two most commonly used attacks vectors for block ciphers: differential cryptanalysis and linear cryptanalysis. The original AES was designed to resist these two attacks.

It has been shown that for mappings chosen uniformly at random from the set of all m-bit bijective mappings, the expected value of the highest probability of a (non-trivial) differential characteristic is at most $\frac{2m}{2^m}$ [11]. In our case where $m = 8$, this means that for a randomly chosen 8-bit S-box the expected maximum probability of a differential characteristic is $\frac{16}{2^8} = 2^{-4}$.

Since the number of active S-boxes for four rounds of the AES is at least 25 [4], one has an upper bound of the probability for any 4-round differential characteristic of 2^{-100}, and thus an upper bound for any 8-round differential characteristic of 2^{-200}. This is sufficient to conclude that differential cryptanalysis will not pose a threat to variants of the AES where the S-box is replaced by a randomly chosen 8-bit S-box.

It is possible to prove a similar result for linear cryptanalysis using the bounds of linear characteristics from [12].

3.2 Integral Cryptanalysis on Four Rounds

Summary.

Before we go into the details of the attack, let us summarize it shortly. The attack splits the task of determining the secret S-box into consecutive steps that find increasingly better.

First we use the fact that we can create balanced sets of intermediate texts right after the first `SubBytes` step in round 1 by applying the SQUARE attack as a chosen

ciphertext attack[1]. These balanced sets can be used to set up a system of linear equations which can be used to determine the secret S-box up to affine equivalence over $\mathbb{F}_2^8$ as is similarly done in the SASAS attack [1]. A representative from this equivalence class is already sufficient to determine the whitening key up to 256 variants.

The knowledge about the whitening key and the representative of the S-box equivalence class allow us now to determine the intermediate texts right before the `MixColumns` step in round 1 up to affine equivalence over $\mathbb{F}_2^8$. As a result of the Square attack, the intermediate texts *after* the `MixColumns` step should take on each byte value in each byte position exactly once. This can be used to determine the secret S-box up to affine equivalence over $\mathbb{F}_{256}$. Finally, the secret S-box can be determined using knowledge of the key schedule.

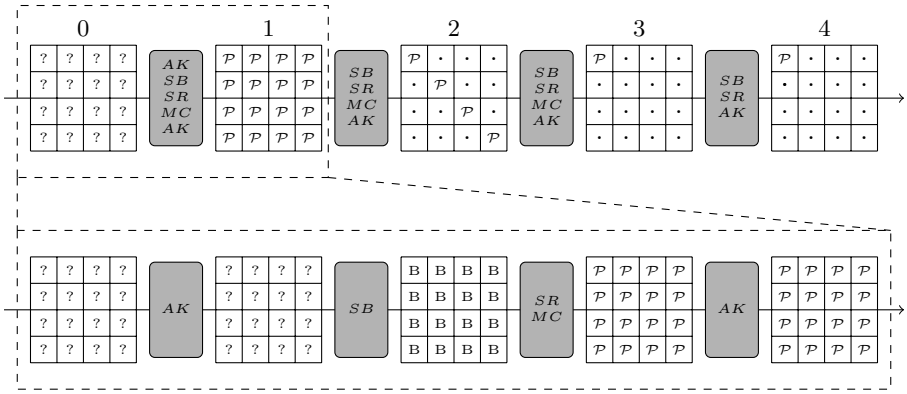

Figure 1: Outline of the 4-round integral attack. The following notation is used: $\mathcal{P}$ takes each of the 256 values once, $\cdot$ is constant, B is balanced and the values ? are unknown.

Prerequisites.

Before we start with the attack, let us clarify the notation. We assume that the last round, the fourth in this case, does not contain a `MixColumns` operation, as is the case for the last round of standard AES.

By a Λ-set, we mean a set of 256 messages that differ only in one byte but take for this byte all possible 256 values. Just as in the standard Square [3] attack, when we decrypt a Λ-set with 4-round AES, we get intermediate texts right after the `SubBytes` step of round 1 that are balanced, i.e. the sum of all texts is equal to the text containing only zeroes, In particular, this set of messages is balanced in every byte.

[1]The reason for using a chosen ciphertext instead of a chosen plaintext attack will be explained later.

Finding an Affine Equivalent of the Secret S-box over $\mathbb{F}_2^8$.

Let $p_i, 0 \leq i < 256$, be the list of the first bytes of the 256 plaintexts, generated from the Λ-set of ciphertexts. Let k_0 be the first byte of the whitening key. We can now write the fact that the intermediate texts are balanced right after the first `SubBytes` step as

$$\bigoplus_{i=0}^{255} S(p_i \oplus k_0) = 0$$

where S is the secret S-box. Let $z_i := S(k_0 \oplus i)$. The above equation is then linear in the z_{p_i} and can be written as

$$z_{p_0} \oplus z_{p_1} \oplus \cdots \oplus z_{p_{255}} = 0. \tag{1}$$

As duplicate values in the p_i values will cancel out, only those p_i need to be taken into account that appear an odd number of times in the list.

Taking different Λ-sets of ciphertexts, we can now try to generate enough linear equations to be able to determine S uniquely. Unfortunately, we encounter two problems now. Firstly, we do not know the value of k_0. We can thus only hope to determine $S(k_0 \oplus \cdot)$. Secondly, the above equations are invariant under affine transformations: Let A be an affine transformation from $\mathbb{F}_2^8$ to $\mathbb{F}_2^8$. Then

$$A(z_{p_0}) \oplus A(z_{p_1}) \oplus \cdots \oplus A(z_{c_{255}}) = 0$$

is also true for any set of p_i that fulfills equation (1) and has an even number of summands. We can thus at best determine $S(k_0 \oplus \cdot)$ up to 2^{72} affine equivalent variants. Using the fact that the affine mapping needs to be invertible, we can thus at best determine the set

$$\{A \circ S(k_0 \oplus \cdot) \mid A : \mathbb{F}_2^8 \to \mathbb{F}_2^8 \text{ is invertible}\}$$

which is of size $2^{70.2}$.

As each linear equation like equation 1 gives us one byte of information and as we can only determine the S-box up to $2^{72} = 2^{9 \cdot 8}$ variants, there can at most be $256 - 9 = 247$ linearly independent equations like equation (1). We found that using 256 different Λ-sets suffices in most cases to generate a set of equations with rank 247.

Given such a set of equations, it is now easy to determine one representative from the set of affine equivalents to $S(k_0 \oplus \cdot)$. Let this representative be denoted as S', i.e. $S' = A \circ S(k_0 \oplus \cdot)$ for some invertible affine $A : \mathbb{F}_2^8 \to \mathbb{F}_2^8$ and unknown k_0.

Determining the Whitening Key.

Let now $p_{i,j}$ with $0 \leq i < 256$ and $0 \leq j < 8$ be byte j of the plaintext i in one of the Λ-sets and let k_j be byte j of the whitening key. We then have for $a \in \mathbb{F}_2^8$:

$$a = k_j \quad \Rightarrow \quad 0 = \bigoplus_{i=0}^{255} S(a \oplus p_{i,j}),$$

which is generally not true for $a \neq k_j$, a fact the standard Square attack is based on as well. For invertible affine $A : \mathbb{F}_2^8 \to \mathbb{F}_2^8$, we also have the equivalence

$$0 = \bigoplus_{i=0}^{255} S(a \oplus p_{i,j}) \quad \Leftrightarrow \quad 0 = \bigoplus_{i=0}^{255} A \circ S(a \oplus p_{i,j}).$$

We can thus for each byte j with $1 \leq j < 8$ find $k_j \oplus k_0$ by trying out for which of the 256 possible values of a we have

$$\bigoplus_{i=0}^{255} S'(a \oplus p_{i,j}) = 0$$

for all Λ-sets. This allows us to determine the whitening key up to 256 variants, depending on the value of k_0. Let us set $k' = (0, k_1 \oplus k_0, k_2 \oplus k_0, \ldots, k_{15} \oplus k_0)$. Then when using k' as the whitening key and S' as the S-box for encryption, the intermediate texts after the `ShiftRows` step in round 1 will correspond to the correct intermediate texts up to a fixed affine transformation on each byte.

Finding an Affine Equivalent of the Secret S-box over $\mathbb{F}_{256}$.

When we decrypt a Λ-set, the set of intermediate texts that we get after the `MixColumns` step in round 1 will take all 256 possible values in each of the 16 state bytes (see Figure 1). The key idea here is to use this property to filter out wrong candidates for the secret S-box.

For a set of 256 bytes, we say that it has the $\mathcal{P}$ property if it contains every possible value exactly once. Let V be a set of 256 byte vectors. We will likewise say that V has the $\mathcal{P}$ property if V has this property in every byte position.

If V is now the set of intermediate texts after the `MixColumns` operation in round 1, that is the result of the decryption of a Λ-set, we know from the Square attack that V has the $\mathcal{P}$ property. Let now D be the corresponding set of intermediate texts directly before the `MixColumns` step. We can test our candidate S' for S, by constructing the corresponding candidate set D' for the intermediate texts after the `ShiftRows` step in round 1 with our acquired knowledge of the whitening key, and applying the `MixColumns` operation on this set D' to see whether we obtain a set with the $\mathcal{P}$ property.

For how many of the 2^{72} candidates for S' do we expect this to hold? Let A be the affine transformation by which S' deviates from S. Then the byte vectors in D' also

deviate by this transformation from the true set D. Clearly, if A consists only of an addition, the $\mathcal{P}$ property of MD' is preserved where M is the `MixColumns` matrix. We can thus restrict A to linear transformations.

In the case, that A corresponds to an invertible linear mapping over $\mathbb{F}_{256}$, i.e. a multiplication with some element from $\mathbb{F}_{256}^*$, the set of intermediate texts after the `MixColumns` step will still have the $\mathcal{P}$ property as well since the linear transformation commutes with the multiplication within the `MixColumns` matrix M and the application of the invertible linear transformation A on the set MD leaves the $\mathcal{P}$ property untouched:

$$MD' = MAD = AMD.$$

Opposed to this, when A does not commute with the multiplication in $\mathbb{F}_{256}$, the $\mathcal{P}$ property of MD' is in general not preserved. As is shown in Appendix B, if A commutes with a primitive element of $\mathbb{F}_{256}$, A corresponds to multiplication with an element of $\mathbb{F}_{256}$. As `03` is a primitive element of the Rijndael field and is an entry in every row and column of M, the only class of affine transformations that preserve the $\mathcal{P}$ property of MD' is exactly the affine transformations over $\mathbb{F}_{256}$.

Checking whether the $\mathcal{P}$ property holds for MD' allows us thus to find the correct S up to affine transformations over $\mathbb{F}_{256}$. Nevertheless, still $2^{72-16} = 2^{56}$ candidates need to be tested.

Complexity Reduction: Finding the Affine Equivalent Over $\mathbb{F}_{256}$.

The specific structure of the `MixColumns` matrix M allows us to reduce the computational complexity of finding the correct affine representative amongst the 2^{56} possible candidates.

Let us define that a set of $2l$ vectors over $\mathbb{F}_2^n$ has the $\mathcal{R}$ property if both 1 and 0 appear in every bit position exactly l times. Note that the $\mathcal{P}$ property implies the $\mathcal{R}$ property and that the $\mathcal{R}$ property implies that the set of vectors is balanced but the opposite direction of implications is in generally false. As the $\mathcal{R}$ property, like the $\mathcal{P}$ property, is not preserved by the `MixColumns` layer, we still expect to find the correct representative by testing for the $\mathcal{R}$ property instead of the $\mathcal{P}$ property[2].

Let us take a closer look at the specific form of matrix M. When written as a linear function from F_{256}^4 to F_{256}^4, it has the form

$$M = \begin{pmatrix} \texttt{02} & \texttt{03} & \texttt{01} & \texttt{01} \\ \texttt{01} & \texttt{02} & \texttt{03} & \texttt{01} \\ \texttt{01} & \texttt{01} & \texttt{02} & \texttt{03} \\ \texttt{03} & \texttt{01} & \texttt{01} & \texttt{02} \end{pmatrix}. \tag{2}$$

If we associate the multiplication with `01`, `02`, and `03` with their respective linear

[2]This was indeed the case for all our test runs.

mappings from $\mathbb{F}_2^8$ to $\mathbb{F}_2^8$, we get the following representations:

$$\texttt{01} = \begin{pmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0\\ 0&0&0&0&0&0&0&1 \end{pmatrix} \quad \texttt{02} = \begin{pmatrix} 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 1&0&0&0&1&0&0&0\\ 1&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&1&0\\ 1&0&0&0&0&0&0&1\\ 1&0&0&0&0&0&0&0 \end{pmatrix} \quad \texttt{03} = \begin{pmatrix} 1&1&0&0&0&0&0&0\\ 0&1&1&0&0&0&0&0\\ 0&0&1&1&0&0&0&0\\ 1&0&0&1&1&0&0&0\\ 1&0&0&0&1&1&0&0\\ 0&0&0&0&0&1&1&0\\ 1&0&0&0&0&0&1&1\\ 1&0&0&0&0&0&0&1 \end{pmatrix}. \tag{3}$$

If we now write $a_0, a_1, \ldots, a_7$ for the rows of A we can write the first row of the 32×32 matrix MA over $\mathbb{F}_2$ as

$$v := (a_1, \quad a_0 \oplus a_1, \quad a_0, \quad a_0)\,.$$

We see now that whether or not the first bit in the set MD' satisfies the $\mathcal{R}$ property relies solely on the rows a_0 and a_1 of the matrix A. As we only need to test matrices A that are not linearly equivalent over $\mathbb{F}_{256}$, we can fix one row of A to a non-zero constant. Let a_0 be fixed. Then we only need to try out all 2^8 possible values for a_2 to see which one gives us the $\mathcal{R}$ property in this bit.

After having determined a_1 (and fixed a_0), we can use the second row of MA to determine a_2 and continue on to determine A uniquely. In each step, we only need to test 2^8 possible values. We can thus split the task of trying of out all 2^{56} candidates for A, to trying out row by row which reduces the complexity to $7 \cdot 2^8 \approx 2^{11}$ steps.

Determining the Secret S-box.

Without assuming anything about the key schedule, we can only determine the secret S-box up to an additive constant before and after the S-box, i.e. $S'(x) \sim a \oplus S(b \oplus x)$ since any additive constants can also be seen as part of the round keys. When not assuming anything about the key schedule, one can for example require that the first byte of the whitening key and the first round key is zero. It is straightforward then to find the correct representative for S out of the 2^{16} options under these constraints. Using knowledge about the key schedule, one can also easily determine the correct variants for the round keys and adjust the representative for the S-box accordingly.

The Complexity of the Attack.

The needed data consists of the decryption of 256 Λ-sets which corresponds to a data complexity of 2^{16} chosen ciphertexts. As most of these texts are only used to generate the linear system of equations in the first plaintext byte, most plaintext pairs can be discarded after the corresponding equation has been extracted. The memory complexity is thus $2^{8+8} = 2^{16}$ bytes.

Let us go through the steps to see what the time complexity is. Determining S' up to affine equivalence over $\mathbb{F}_2^8$ requires solving a system of linear equations in 2^8 variables. This requires $2^{3 \cdot 8} = 2^{24}$ steps where each step is comparable to a table lookup. Finding the whitening key requires trying out for each of the 16 key bytes all 2^8 possible solutions with one Λ-set of 2^8 values. It thus takes about $16 \cdot 2^8 \cdot 2^8 = 2^{20}$ table lookups.

To determining S' up to affine equivalence over $\mathbb{F}_{256}$ using the $\mathcal{R}$ property, for each of the seven rows of A that have not been fixed we have to test 2^8 values, each with a Λ sets. Thus the total complexity of this step is $7 \cdot 2^8 \cdot 2^8 \approx 2^{19}$. A step here has about the same complexity as a table lookup.

The complexity of the attack is dominated by solving the linear system of equations, namely 2^{24} steps, which corresponds to 2^{17} encryptions when assuming a complexity of 2^5 table lookups per encryption round. We ran the attack 1000 times on the single core of an Intel Core i7-4600M CPU at 2.90GHz. It found both the correct S-box and the correct key each time and always ran in less than a second (including reading the input data).

3.3 Integral Cryptanalysis on Five Rounds

The attack on four rounds can be extended to five rounds using a technique by Ferguson et al. [6] that allowed to improve the SQUARE attack on six rounds. The underlying idea is to create sets of ciphertexts that form a Λ-set right before the `MixColumns` step of round 4. Unfortunately, even with key guessing, it is not possible to determine such a set without knowledge of the secret S-box. However, by taking all 2^{32} possible values for four bytes that are in the same column during the `MixColumns` step of round 4 and keeping all other bytes constant, we can generate a set of ciphertexts that will take all 2^{32} values in that column. This set can now be viewed as the union of 2^{24} Λ-sets (see Figure 2).

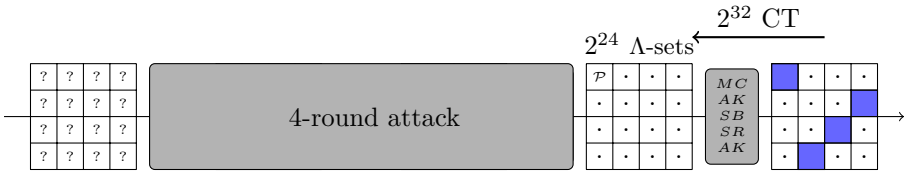

Figure 2: The 2^{32} ciphertexts take all possible combinations in the blue bytes but constant values in the rest. The state before the `MixColumns` step in the round before can be seen as the union of 2^{24} Λ-sets as depicted here. It is then possible to apply the 4-round attack again.

A set of ciphertexts that gives us a Λ-set in the `MixColumns` step of round 4 will generate a balanced set right after the `SubBytes` step of round 1. As a sum of balanced sets remains balanced, decrypting our 2^{32} ciphertexts, we get a balanced set of size 2^{32} after the `SubBytes` step of round 1. This set can now be used to mount the four round attack on five rounds as well.

Just as in the four round version, we use the fact when such a set is balanced, we can, by using 256 of them, create a system of linear equations that can be solved to find an S-box S' that is an affine equivalent to S over $\mathbb{F}_2^8$. We can use the knowledge of S' again to determine the whitening key up to 256 variants. We can then again generate the corresponding intermediate texts after the first `SubBytes` step that are

affinely equivalent over $\mathbb{F}_2^8$ to the true texts. With these texts we can now determine S up to affine equivalence over $\mathbb{F}_{256}$ by using the $\mathcal{R}$ property. Note that when using the $\mathcal{R}$ property here, we expect the correct set of texts to take in each bit the values 0 and 1 each exactly 2^{31} times as we are now working with the union of 2^{24} Λ-sets. Again to determine S exactly and finding the correct master key is straightforward from this point.

How do the complexities of the attack change as compared to the 4-round variant? As we need 256 sets of ciphertexts, each of size 2^{32}, this leaves us with a data complexity of 2^{40}, an increase by a factor of 2^{24} in comparison to the four round attack. The time complexity of solving the linear system of equations does not change (it is still a system of 256 equations in 256 variables). The complexity of the whitening key recovery increases with the size of the balanced sets, i.e. again by a factor of 2^{24}, leaving us with a complexity of 2^{44} table lookups. Likewise is the complexity of checking the $\mathcal{R}$ property increased by a factor of 2^{24} to a total complexity of 2^{43} steps of the same complexity as a table lookup. This leaves the total time complexity at roughly 2^{45} steps which corresponds to 2^{38} encryptions when assuming a complexity of 2^5 table lookups per encryption round.

The data complexity of 2^{40} chosen ciphertexts corresponds to 18 terabyte of data. But as most of the sets of 2^{32} plaintexts are each only used to generate one linear equation (in the 256 variables), apart from a few (16 suffice), most can be discarded during the generation of the linear system of equations, leaving us with at most 2^{40} bytes that need to be stored in memory at any point in time.

3.4 Integral Cryptanalysis on Six Rounds

The standard way of extending the Square attack to six rounds (in the case of a chosen ciphertext attack) is by guessing four bytes of the whitening key and peeling of the first round of encryption for one byte of intermediate text, thereby increasing the time complexity of the attack by a factor of 2^{32}. Unfortunately, this does not extend to the AES with a secret S-box as knowledge of the S-box is required to strip off the first round.

There is nonetheless a way to extend the five round attack to six rounds. Over one round of the AES, the four output bytes of one column only depend on four of the input bytes. Thus, it is possible to describe two rounds of AES with a secret S-box as the parallel application of four Super-boxes (see also [5]) with a linear transformation before and after. Such a Super-box consists of the parallel application of four S-boxes, a key addition a multiplication of the four bytes with the `MixColumns` matrix, again an application of four S-boxes in parallel and a final key addition.

Just as in the 5-round attack, we can generate sets of texts that are balanced right after the `SubBytes` step in round 2 and we can hence use these texts to generate a system of linear equations that lets us determine the Super-boxes, just as it allowed us to determine the usual S-boxes in the attacks before. Unfortunately, the system of linear equations for one Super-box involves now not 2^8 variables but 2^{32} variables. This means that both the computational complexity as well as the data complexity

increase. For the data complexity, when using the round extension as in the five round attack, we need now 2^{32} sets of each 2^{32} texts, leaving us with a data complexity of 2^{64} chosen ciphertexts. Just as with the attack on the normally sized S-box, the set of equations is not of full rank and lets us determine the Super-box only up to $2^{32 \cdot 32+32} = 2^{1056}$ affine equivalents – only slightly less when taking the necessary bijectivity of the affine transform into account.

The Super-box that we obtain will thus be of the form

$$A \circ \texttt{SubBytes} \circ \texttt{KeyAddition} \circ \texttt{MixColumns} \circ \texttt{SubBytes} \circ \texttt{KeyAddition}$$

where A is an unknown invertible affine mapping over $\mathbb{F}_2^{32}$ and where the other standard AES steps are truncated to operate on four bytes only. Despite our lack of knowledge of A, this form is already enough to extract from it the secret S-box and the involved key bytes up to 2^{16} variants, i.e. up to two additive constants applied before and after the S-box. After this, it is straightforward to uniquely determine the secret S-box and the key e.g. by guessing the two additive constants and applying standard 6-round Square attack.

If we decrypt a Λ-set with our affinely transformed Super-box, we get a set that is balanced right after the first `SubBytes` step of the Super-box as described in the SASAS paper [1]. Note that it is necessary to assume that A distributes the 8 bits that are being varied in the Λ-set to at least two S-Boxes, an assumption that is true for almost all possible A. At this point we can thus simply apply again the same techniques as we did for the four round attack to determine the secret S-box and the involved key bytes, only that we mount the attack on the affine equivalent of the Super-box now instead of the whole cipher.

What is the complexity of this attack? As already mentioned above, the data complexity is 2^{64} chosen ciphertexts. The time complexity is dominated by the first step of solving the system of 2^{32} linear equations over 2^{32} variables. Using Gaussian elimination, this step consists of 2^{96} operations, each comparable in complexity to a table lookup. Thus, the time complexity corresponds to 2^{90} encryptions when assuming a complexity of 2^5 table lookups per encryption round. The memory complexity of $2^{32} \cdot 2^{32} \cdot 32 = 2^{69}$ bytes is also dominated by the size of the system of equations.

3.5 A Note on Chosen Ciphertext vs. Chosen Plaintext

Due to the symmetry of the AES regarding encryption and decryption, the attacks described here principally work in both directions. Interestingly though, for the attacks on four and five rounds, the chosen ciphertext variant is considerably more effective than the chosen plaintext attack. This is because the `MixColumns` matrix is sufficiently sparser than its inverse, creating a difference of 2^{16} in the number of steps when applying the $\mathcal{R}$ property. This changes the time complexities of the 4-round and 5-round attacks to 2^{28} and 2^{54}. As the complexity of the 6-round attack is dominated by the solving of the linear system of equations, it does not make a difference in that attack scenario.

4 Conclusion

In this work, we studied the impact of replacing the S-box in the AES by a secret S-box unknown to the adversary. Despite the expected increase in difficulty of recovering the secret information, we were able to mount efficient attacks based on integral cryptanalysis combined with dedicated techniques.

We were able to show that AES-128 with a secret S-box, reduced to 4 and 5 rounds, is susceptible to attacks with practical complexity that successfully recover both the secret S-box and the key. Furthermore, we have shown an attack on a variant with 6 rounds with a time complexity of 2^{90}, which is much less effort than the time required to do exhaustive search of the key, let alone of the S-box.

Similarly to standard AES, it seems difficult to extend our attacks to more than 6 rounds. Also, the gap between the time complexities of integral attacks on standard AES and the AES with a secret S-box increases dramatically for the attack on 6 rounds. It is an open question whether this complexity can be further reduced.

Acknowledgements

The work in this paper has partially been funded by the Nasjonal sikkerhetsmyndighet (NSM).

References

[1] Alex Biryukov and Adi Shamir. “Structural Cryptanalysis of SASAS”. In: *EUROCRYPT 2001*. Ed. by Birgit Pfitzmann. Vol. 2045. LNCS. 2001, pp. 394–405.

[2] Julia Borghoff, Lars R. Knudsen, Gregor Leander, and Søren S. Thomsen. “Cryptanalysis of PRESENT-Like Ciphers with Secret S-Boxes”. In: *Fast Software Encryption, FSE 2011*. Ed. by Antoine Joux. Vol. 6733. LNCS. 2011, pp. 270–289.

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

[4] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002.

[5] Joan Daemen and Vincent Rijmen. “Understanding Two-Round Differentials in AES”. In: *Security and Cryptography for Networks (SCN) 2006*. Ed. by Roberto De Prisco and Moti Yung. Vol. 4116. LNCS. 2006, pp. 78–94.

[6] Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Michael Stay, David Wagner, and Doug Whiting. "Improved Cryptanalysis of Rijndael". In: *Fast Software Encryption, FSE 2000*. Ed. by Bruce Schneier. Vol. 1978. LNCS. 2000, pp. 213–230.

[7] Mahadevan Gomathisankaran and Ruby B. Lee. "Maya: A Novel Block Encryption Function". In: *International Workshop on Coding and Cryptography 2009*. 2009.

[8] Guo-Qiang Liu, Chen-Hui Jin, and Chuan-Da Qi. "Improved Slender-set Linear Cryptanalysis". In: *Fast Software Encryption, FSE 2014*. 2014.

[9] Ralph C. Merkle. "Fast Software Encryption Functions". In: *CRYPTO '90*. Ed. by Alfred Menezes and Scott A. Vanstone. Vol. 537. LNCS. 1991, pp. 476–501.

[10] National Institute of Standards and Technology. "Advanced Encryption Standard". Federal Information Processing Standard (FIPS), Publication 197, U.S. Department of Commerce, Washington D.C. Nov. 2001.

[11] Luke O'Connor. "On the Distribution of Characteristics in Bijective Mappings". In: *EUROCRYPT '93*. Ed. by Tor Helleseth. Vol. 765. LNCS. Springer, 1994, pp. 360–370.

[12] Luke O'Connor. "Properties of Linear Approximation Tables". In: *Fast Software Encryption, FSE '94*. Ed. by Bart Preneel. Vol. 1008. LNCS. Springer, 1995, pp. 131–136.

[13] Matthieu Rivain and Thomas Roche. "SCARE of Secret Ciphers with SPN Structures". In: *Advances in Cryptology - ASIACRYPT 2013*. Ed. by Kazue Sako and Palash Sarkar. Vol. 8269. LNCS. 2013, pp. 526–544.

[14] Bruce Schneier. "Description of a New Variable-Length Key, 64-bit Block Cipher (Blowfish)". In: *Fast Software Encryption, FSE '93*. Ed. by Ross J. Anderson. Vol. 809. 1994, pp. 191–204.

[15] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson. "Twofish: A 128-Bit Block Cipher". In: ().

A The AES Key Schedule

In the AES, we think of the round keys as matrices over the Rijndael finite field, just as the state matrix. The first pre-whitening key RK_0 is the n-bit master key itself, so $RK_0 = K$. The key schedule varies slightly across the three AES variants. Here, we describe it for AES-128 and refer to [4] for the other two cases. We consider the 4 columns of the two round keys as $RK_i = (RK_i^0 \| RK_i^1 \| RK_i^2 \| RK_i^3)$ and $RK_{i+1} = (RK_{i+1}^0 \| RK_{i+1}^1 \| RK_{i+1}^2 \| RK_{i+1}^3)$. To derive RK_{i+1} from RK_i, $0 \leq i < T$, we do the following

1. Let $RK_{i+1}^j = RK_i^j$ for $j = 0, 1, 2, 3$,

2. Rotate RK^3_{i+1} such that the byte in the first row is moved to the bottom,
3. Substitute each byte in RK^3_{i+1} by using the S-box from the `SubBytes` operation,
4. Update the byte in the first row of RK^3_{i+1} by adding $\texttt{02}^{i-1}$ from the Rijndael finite field, and
5. Let $RK^j_{i+1} = RK^j_{i+1} \oplus RK^{j-1 \mod 4}_{i+1}$ for $j = 0, 1, 2, 3$.

This procedure is repeated for $i = 1, \ldots, T$ to obtain the round keys RK_0 to RK_T.

B Lemma

Let $m \in \mathbb{N}^*$. As $\mathbb{F}_{2^m}$ is an m-dimensional $\mathbb{F}_2$-vector space, its elements can be represented as m-dimensional $\mathbb{F}_2$-vectors. But as the multiplication in $\mathbb{F}_{2^m}$ obeys the distributive law, the multiplication with an element of $\mathbb{F}_{2^m}$ corresponds to a linear mapping from $\mathbb{F}_2^m$ to $\mathbb{F}_2^m$, that is an $m \times m$ matrix over $\mathbb{F}_2$. For an element $a \in \mathbb{F}_{2^m}$, let L_a denote the corresponding $m \times m$ matrix. For $b \in \mathbb{F}_{2^m}$, we then have $a \cdot b = L_a b$.

Lemma 1. *Let a be a primitive element of $\mathbb{F}_{2^m}$. Let B be an $m \times m$ matrix over $\mathbb{F}_2$ which commutes with L_a. Then there exists $b \in \mathbb{F}_{2^m}$ such that $L_b = B$.*

Proof. Let c be any element from $\mathbb{F}^*_{2^m}$. As a is primitive, there exists $k \in \mathbb{N}^*$ such that $c = a^k$ and likewise $L_c = L_a^k$. As B commutes with L_a, by induction B also commutes with L_c. Clearly, B also commutes with L_0, so B commutes with all elements of $\mathbb{F}_{2^m}$.

Let now $b \in \mathbb{F}_{2^m}$ be the image of 1 under B, $b = B1$. We then have for any $c \in \mathbb{F}^*_{2^m}$:

$$Bc = L_1 Bc = L_c L_{c^{-1}} Bc = L_c B L_{c^{-1}} c = L_c B 1 = L_c b = c \cdot b = b \cdot c = L_b c.$$

As this is true for any $c \in \mathbb{F}^*_{2^m}$ and clearly also for 0, we have $B = L_b$. $\square$

Ciphers for MPC and FHE

Publication Information

Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. "Ciphers for MPC and FHE". in: *Advances in Cryptology - EUROCRYPT 2015*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9056. Lecture Notes in Computer Science. Springer, 2015, pp. 430–454. ISBN: 978-3-662-46799-2

Contribution

- All authors contributed equally.
- Main contribution is the cryptanalysis of the cipher scheme.
- Further contributions were made in design decisions and writing the specification of the scheme.

Remarks

This publication has been slightly edited to fit the format.

Ciphers for MPC and FHE

Martin Albrecht[1], Christian Rechberger[2], Thomas Schneider[3], Tyge Tiessen[2], and Michael Zohner[3]

[1] Royal Holloway, University of London, UK
martinralbrecht@googlemail.com
[2] Technical University of Denmark, Copenhagen, Denmark
{crec,tyti}@dtu.dk
[3] TU Darmstadt, Darmstadt, Germany
{thomas.schneider,michael.zohner}@ec-spride.de

Abstract. Designing an efficient cipher was always a delicate balance between linear and non-linear operations. This goes back to the design of DES, and in fact all the way back to the seminal work of Shannon.

Here we focus, for the first time, on an extreme corner of the design space and initiate a study of symmetric-key primitives that minimize the multiplicative size and depth of their descriptions. This is motivated by recent progress in practical instantiations of secure multi-party computation (MPC), fully homomorphic encryption (FHE), and zero-knowledge proofs (ZK) where linear computations are, compared to non-linear operations, essentially "free".

We focus on the case of a block cipher, and propose the family of block ciphers "LowMC", beating all existing proposals with respect to these metrics by far. We sketch several applications for such ciphers and give implementation comparisons suggesting that when encrypting larger amounts of data the new design strategy translates into improvements in computation and communication complexity by up to a factor of 5 compared to AES-128, which incidentally is one of the most competitive classical designs. Furthermore, we identify cases where "free XORs" can no longer be regarded as such but represent a bottleneck, hence refuting this commonly held belief with a practical example.

Keywords: block cipher, multiplicative complexity, multiplicative depth, secure multiparty computation, fully homomorphic encryption

1 Introduction

Modern cryptography developed many techniques that go well beyond solving traditional confidentiality and authenticity problems in two-party communication. Secure

multi-party computation (MPC), zero-knowledge proofs (ZK) and fully homomorphic encryption (FHE) are some of the most striking examples.

In recent years, especially the area of secure multi-party computation has moved from a science that largely concerned itself with the mere existence of solutions towards considerations of a more practical nature, such as costs of actual implementations for proposed protocols in terms of computational time, memory, and communication.

Despite important progress and existing proof-of-concept implementations, e.g. [30], [40], [47], [54], [58], [59], [63], there exists a *huge cost gap* between employing cryptographic primitives in a traditional way and using them in the more versatile MPC context. As an example, consider implementations of the AES block cipher, a global standard for the bulk encryption of data. Modern processors achieve a single execution of the block cipher within a few hundred clock cycles (or even less than 100 clock cycles using AES-NI). However, realizing the same cipher execution in the context of an MPC protocol takes many billions of clock cycles and high communication volumes between the participating parties, e.g. several hundreds of Megabytes for two-party AES with security against malicious adversaries [21], [22], [30], [47], [48], [58], [59], [63].

While our design approach is not specific to block ciphers but can be equally applied to e.g. hash functions, in this work, we propose block ciphers that are specifically designed for application in MPC and similar contexts. Traditionally, ciphers are built from linear and non-linear building blocks. These two have roughly similar costs in hardware and software implementations. In CMOS hardware, the smallest linear gate (XOR) is about 2-3 times larger than the smallest non-linear gate (typically, NAND). When implemented in an MPC protocol or a homomorphic encryption scheme, however, the situation is radically different: linear operations come almost for free, since they only incur local computation (resp. do not increase the noise much), whereas the bottleneck are non-linear operations that involve symmetric cryptographic operations and communication between parties (resp. increase the noise considerably). Our motivation hence comes from implementations of ciphers in the context of MPC, ZK, or FHE schemes where linear parts are much cheaper than non-linear parts.

This cost metric suggests a new way of designing a cipher where most of the cryptographically relevant work would be performed as linear operations and the use of non-linear operations is minimized. This design philosophy is related to the fundamental theoretical question of the minimal multiplicative complexity (MC) [13] of certain tasks. Such extreme trade-offs were not studied before, as all earlier designs – due to their target platforms – faired better with obtaining a balance between linear and non-linear operations.

In this work we propose to start studying symmetric cryptography primitives with low multiplicative complexity in earnest. Earlier tender steps in this direction [33], [35], [60] were aimed at good cost and performance when implemented with side-channel attack countermeasures, and are not extreme enough for our purpose. Our question hence is: what is the minimum number of multiplications for building a secure block cipher? We limit ourselves to multiplications in GF(2) and motivate

this as follows:

- By using Boolean circuits we decouple the underlying protocol / primitive (MPC protocol / ZK protocol / FHE scheme) from that of the cipher. Hence, the same cipher can be used for multiple applications.
- GF(2) is a natural choice for MPC protocols based on Yao or GMW (in the semi-honest setting, but also for their extensions to stronger adversaries), ZK protocols, as well as for fully or somewhat homomorphic encryption schemes (cf. Section 2 for details).

By nature of the problem, we are interested in two different metrics. One metric refers to what is commonly called multiplicative complexity (MC), which is simply the number of multiplications (AND gates) in a circuit, see e.g. [13]. The second metric refers to the multiplicative depth of the circuit, which we will subsequently call ANDdepth. We note that already in [26] it was observed that using ciphers with low ANDdepth is of central importance for efficient evaluations within homomorphic encryption schemes. Therefore, the authors of [26] suggest to study block cipher designs that are optimized for low ANDdepth, a task to which we provide a first answer. Our work is somehow orthogonal to Applebaum et. al [2], where the question of what can in principle be achieved in cryptography with shallow circuits was addressed.

This all motivates the following guiding hypothesis which we will test in this paper: "When implemented in practice, a block cipher design with lower MC and lower ANDdepth will result in lower executing times". We note that the relatively low execution times often reported in the literature are *amortized* times, i.e. averaged over many calls of a cipher (in parallel). This, however, neglects the often important *latency*. Hence, another design goal in this work is to reduce this latency.

Outline and contribution.

In Section 2 we describe several schemes with "free XORs". Then, in Section 3, we focus on an extreme corner of the design space of block ciphers and propose a new block-cipher design strategy that minimizes the multiplicative size and depth of the circuit describing it, beating all existing candidates by far with respect to these metrics. In terms of ANDdepth, the closest competitor is PRINCE. In terms of MC, the closest competitor turns out to be Simon. We give a high-level overview over a larger field of competing designs in Section 4. We analyse the security of our constructions in Section 5 and provide experimental evidence for the soundness of our approach in Section 6. In particular, our implementations outperform previously reported results in the literature, often by more than a factor 5 in MPC and FHE implementation settings. They also indicate that in the design space we consider, "free XORs" can no longer be regarded as free but significantly contribute to the overall cost, hence refuting this commonly held belief with a practical example. Finally, we describe our optimisation strategies for implementing our designs in the MPC and FHE case, which might be of independent interest.

Main features and advantages of LowMC.

- Low ANDdepth, and low MC, which positively impacts the latency and throughput of the FHE, MPC, or ZK evaluation of the cipher.
- Partial Sbox layer.
- Security arguments against large classes of statistical attacks like differential attacks, similar to other state-of-the-art designs are given in Section 5. Zorro [33] is the first SPN cipher in the literature that uses a non-full Sbox layer and is related to LowMC in this respect. However, recent attacks on Zorro that exploit this particular property [5], [36], [61], [67], highlight the need to be very careful with this design strategy. In our analysis of LowMC in Section 5 we are able to take these into account.
- In contrast to other constructions, it is easy to obtain tight bounds on the MC and ANDdepth.
- The design is very flexible and allows for a unified description regardless of the blocksize.
- We explicitly de-couple the security claim of a block cipher from the block size.

2 Schemes

In this section we list several schemesfor MPC, FHE, and ZK that benefit from evaluating our cipher. We give a list of example applications for LowMC in the full version of the paper.

2.1 Multi-Party Computation (MPC)

There are two classes of practically efficient secure multi-party computation (MPC) protocols for securely evaluating Boolean circuits where XOR gates are considerably cheaper (no communication and less computation) than AND gates.

The first class of MPC protocols has a constant number of rounds and their total amount of communication depends on the MC of the circuit (each AND gate requires communication). Examples are protocols based on Yao's garbled circuits [68] with the free XOR technique [46]. To achieve security against stronger (i.e., malicious or covert) adversaries, garbled circuit-based protocols apply the cut-and-choose technique where multiple garbled circuits are evaluated, e.g., [4], [28], [30], [41], [42], [47], [49], [50], [51], [52], [53], [59], [63], [64]; also MiniLEGO [29] falls into this class.

The second class of MPC protocols has a round complexity that is linear in the ANDdepth of the evaluated circuit (each AND gate requires interaction) and hence the performance depends on both, the MC and ANDdepth of the circuit. Examples are the semi-honest secure version of the GMW protocol [34] implemented in [18], [62], and tiny-OT [58] with security against malicious adversaries.

2.2 Fully homomorphic encryption (FHE)

In all somewhat and fully homomorphic encryption schemes known so far XOR (addition) gates are considerably cheaper than AND (multiplication) gates. Moreover, XOR gates do not increase the noise much, whereas AND gates increase the noise considerably (cf. [37]). Hence, as in somewhat homomorphic encryption schemes the parameters must be chosen such that the noise of the result is low enough to permit decryption, the overall complexity depends on the ANDdepth.

2.3 Zero-Knowledge proof of knowledge (ZK)

In several zero-knowledge proof protocols XOR relations can be proven for free and the complexity essentially depends on the number of AND gates of the relation to be proven. Examples for such protocols are [10], [15] and the recently proposed highly efficient protocol of [43] that requires only one evaluation of a garbled circuit [68] and can make use of the free XOR technique [46].

3 Description of LowMC

LowMC is a flexible block cipher based on an SPN structure where the block size n, the key size k, the number of Sboxes m in the substitution layer and the allowed data complexity d of attacks can independently be chosen[4]. The number of rounds needed to reach the security claims is then derived from these parameters.

To reduce the MC, the number of Sboxes applied in parallel can be reduced, leaving part of the substitution layer as the identity mapping. Despite concerns raised regarding this strategy [67], we will show that security is viable. To reach security in spite of a low MC, pseudorandomly generated binary matrices are used in the linear layer to introduce a very high degree of diffusion. A method to accountably instantiate LowMC is given in Section 3.3.

Encryption with LowMC starts with a key whitening, followed by several rounds of encryption where the exact number of rounds depends on the chosen parameter set. A single round is composed as follows:

$$\begin{gathered}\textsc{LowMCRound}(i) = \\ \textsc{KeyAddition}(i) \circ \textsc{ConstantAddition}(i) \circ \textsc{LinearLayer}(i) \circ \textsc{SboxLayer}\end{gathered}$$

In the following we give a detailed description of the individual steps.

SboxLayer is an m-fold parallel application of the same 3-bit Sbox on the first $3m$ bits of the state. If $n > 3m$ then for the remaining $n - 3m$ bits, the SboxLayer is the identity. The selection criteria for the Sbox were as follows:

- Maximum differential probability: 2^{-2}

[4]The number of Sboxes is limited though by the block size as the Sboxes need to fit into a block.

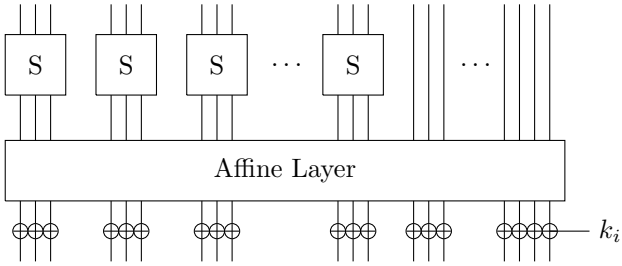

Figure 1: Depiction of one round of encryption with LowMC.

- Maximum linear probability: 2^{-2}
- Simple circuit description involving MC = 3 AND gates, with ANDdepth=1
- Each of the 8 non-zero component functions has algebraic degree 2

The Sbox is specified in 2, and coincides with the Sbox used for PRINTcipher [45]. Other representations of the Sbox can be found in the full version of this paper. LinearLayer(i) is the multiplication in GF(2) of the state with the binary $n \times n$ matrix `Lmatrix[i]`. The matrices are chosen independently and uniformly at random from all invertible binary $n \times n$ matrices.

ConstantAddition(i) is the addition in GF(2) of `roundconstant[i]` to the state. The constants are chosen independently and uniformly at random from all binary vectors of length n.

KeyAddition(i) is the addition in GF(2) of `roundkey[i]` to the state. To generate `roundkey[i]`, the master key `key` is multiplied in GF(2) with the binary $n \times k$ matrix `Kmatrix[i]`. The matrices are chosen independently and uniformly at random from all binary $n \times k$ matrices of rank $\min(n, k)$.

Decryption is done in the straightforward manner by an inversion of these steps.

$$\boxed{S(a, b, c) = (a \oplus bc, a \oplus b \oplus ac, a \oplus b \oplus c \oplus ab)}$$

Figure 2: Specification of the 3-bit Sbox.

3.1 Pseudocode

`plaintext` and `state` are n-bit quantities. `key` is a k-bit quantity, which can both be larger or smaller than n. `r` is the number of rounds.

blocksize n	sboxes m	keysize k	data d	rounds r	ANDdepth	ANDs per bit
256	49	80	64	11	11	6.3
256	63	128	128	12	12	8.86

Table 1: Parameter sets of LowMC instantiations. One first set has PRESENT-like security parameters, the second set has AES-like security parameters.

```
ciphertext = encrypt (plaintext,key)
  //initial whitening
  state = plaintext + MultiplyWithGF2Matrix(KMatrix(0),key)

  for (i = 1 to r)
    //m computations of 3-bit sbox,
    //remaining n-3m bits remain the same
    state = Sboxlayer (state)

    //affine layer
    state = MultiplyWithGF2Matrix(LMatrix(i),state)
    state = state + Constants(i)

    //generate round key and add to the state
    state = state + MultiplyWithGF2Matrix(KMatrix(i),state)
  end
  ciphertext = state
```

3.2 Parameters

Our security analysis against differential, linear, higher-order, meet-in-the-middle, algebraic, and slide attacks suggests that, except with negligible probability, any uniformly randomly chosen set of matrices leads to a secure construction for the parameters given in Table 1. For a larger selection of parameters bundled with security bounds, see the full version of this paper.

3.3 Instantiation of LowMC

To maximize the amount of diffusion done by the linear layer, we rely on randomly generated, invertible binary matrices. As there exist no binary matrices of size larger than 1×1 that are MDS, and as it is generally an NP-complete problem to determine the branching number of a binary matrix [7], there is no obviously better method to reach this goal. The problem in the instantiation of LowMC is to find an accountable way of constructing the random matrices and vectors that leaves no room for the designer to plant backdoors.

Our recommended instantiation is a compromise between randomness, accountability and ease of implementation. It uses the Grain LSFR as a self-shrinking generator (see [39] and [56]) as a source of random bits. The exact procedure can be found in the full version of this paper.

It must be mentioned though that it is principally possible to use any sufficiently random source to generate the matrices and constants. It is also not necessary that the source is cryptographically secure.

4 Comparison with other ciphers

In the following we survey a larger number of existing cipher designs and study their ANDdepth and MC per encrypted bit which we summarize in Table 2. We both choose representative candidates from various design strategies, as well as the designs that are most competitive in terms of our metrics. We do this in two distinct categories: AES-like security (with key sizes of 128-bits and more and data security and block size of 128-bits and more), and lightweight security (data security and block size of 96 bits or below). Note that data security refers to the log_2 of the allowable data complexity up to which a cipher is expected to give the claimed securtiy against shortcut attacks. For LowMC we explicitly de-couple the data security from the block size of the cipher as the proposed design strategy favour larger block sizes but we don't see a new for larger data security than 128. For size-optimized variants we instantiate ℓ-bit adders using a ripple-carry adder which has $\ell - 1$ ANDs and ANDdepth $\ell - 1$; for depth-optimized variants we instantiate them with a Ladner-Fischer adder that has $\ell + 1.25\ell \log_2 \ell$ ANDs and ANDdepth $1 + 2\log_2 \ell$, cf. [62].

We first survey AES versions and then ciphers with related security properties. The Sbox construction of [12] has 34 AND gates and ANDdepth 4 (the size optimized Sbox construction of [11] has only 32 AND gates, but higher ANDdepth 6). See also Canright [17]. To encrypt a 128-bit block, AES-128 has 10 rounds and uses 160 calls to the Sbox (40 for key schedule), hence 5 440 AND gates, or 42.5 AND gates per encrypted bit. To encrypt a 128-bit block, AES-192 has 12 rounds and uses 192 calls to the Sbox (32 for key schedule), hence 6 528 AND gates, or 51 AND gates per encrypted bit. To encrypt a 128-bit block, AES-256 has 14 rounds and uses 224 calls to the Sbox (56 for key schedule), hence 7 616 AND gates, or 59.5 AND gates per encrypted bit.

AES is actually comparatively efficient. Other ciphers with a different design strategy can have very different properties. Threefish [27] is a cipher with large block size. Threefish with its 512-bit block size has 72 rounds with 4 additions modulo 2^{64} each resulting in 35.438 AND gates per encrypted bit and ANDdepth=4 536 (63 per round). Threefish with its 1 024-bit block size has 80 rounds with 8 additions each resulting in 39.375 AND gates per bit and ANDdepth=5 040 (63 per round). The recently proposed NSA cipher Simon [6] is also a good candidate to be of low multiplicative complexity. If b is the block size, it does $b/2$ AND gates per round, and

ANDdepth is equal to the number of rounds. For a key size of 128 bit (comparable to AES) and block size 128 bit, it needs 68 rounds. This means, 4 352 AND gates, or 34 AND gates per bit.

In the lightweight category, we consider Present, but also Simon. The Present Sbox can be implemented with as little as 4 AND gates which is optimal [19] and has ANDdepth 3. With $16 \cdot 31 = 496$ Sbox applications per 64 bit block we arrive at 31 AND gates per bit. A depth-optimized version of the Present Sbox with ANDdepth 2 and 8 ANDs is given in the full version of this paper. The 128bit secure version of Present differs only in the key schedule. Simon-64/96 has a 96 bit key, block size 64 bit and 42 rounds and Simon-32/64 has a 64 bit key, block size 32 bit and 32 rounds; see above for MC and ANDdepth. As another data point, the DES circuit of [65] has 18 175 AND gates and ANDdepth 261. KATAN [16] has 254 rounds. In KATAN32, the ANDdepth increases by two every 8 rounds resulting in an ANDdepth of 64; with 3 AND gates per round and a block size of 32 bit this results in 23.81 ANDs per bit, but similar to Simon-32/64 applications are limited due to the small block size. In KATAN48 and KATAN64 the ANDdepth increases by 2 every 7 rounds resulting in an ANDdepth of 74. KATAN48 has 6 ANDs per round and a block size of 48 bit resulting in 31.75 ANDs per bit. KATAN64 has 9 ANDs per round and a block size of 64 bit resulting in 35.72 ANDs per bit. Prince [8] has 12 rounds and each round can be implemented with 10 AND gates and ANDdepth 2, cf. [26]. NOEKEON [20] is a competitive block cipher with 16 rounds and each round applies 32 S-boxes consisting of 4 AND gates with ANDdepth 2 each.

LowMC is easily parameterizable to all these settings, see also Table 1 in Section 3. It has at most (if $3m = n$) one AND gate per bit per round which results, together with a moderate number of rounds to make it secure, in the lowest ANDdepth and lowest MC per encrypted bit, cf. Table 2.

5 Resistance against cryptanalytic attacks

The number of rounds r equals ANDdepth, and is hence a crucial factor to minimize. For this we evaluate the security of the construction against an array of known attack vectors. Below we especially discuss differential, linear and high-order attacks, as their analysis is a relevant technical contribution in itself. For a short discussion of other attack vectors, we refer to the full version of this paper.

We aim to prove the LowMC designs secure against classes of known attacks. However, due to the choice of random linear layers it is not immediately clear how to bound the probability of differential or linear characteristics. This is something we will investigate and resolve in Section 5.1. Due to the extremely simple description of the Sbox, higher order [44] and cube attacks [24] that exploit a relatively slow growth in the algebraic degree appear to be the most promising attack vector, and are studied in Section 5.4. The quality of these bounds is tested on small versions of LowMC. This all will allow us to formulate in Section 5.6 a relatively simple expression for deriving a lower bound for the number of rounds given other parameters like the

Cipher	Key size	Block size	Data sec.	ANDdepth	ANDs/bit	Sbox representation
AES-like security						
AES-128	128	128	128	40 (60)	43 (40)	[12] ([11])
AES-192	192	128	128	48 (72)	51 (48)	[12] ([11])
AES-256	256	128	128	56 (84)	60 (56)	[12] ([11])
Simon	128	128	128	68	34	[6]
Simon	192	128	128	69	35	[6]
Simon	256	128	128	72	36	[6]
Noekeon	128	128	128	32	16	[20]
Robin	128	128	128	96	24	[35]
Fantomas	128	128	128	48	16.5	[35]
Threefish	512	512	512	936 (4 536)	306 (36)	[27]
Threefish	512	1 024	1024	1 040 (5 040)	340 (40)	[27]
LowMC	128	256	128	**12**	8.85	full version
Lightweight security						
PrintCipher-96	160	96	96	96	96	full version
PrintCipher-48	80	48	48	48	48	full version
Present	80 or 128	64	64	62 (93)	62 (31)	full version ([19])
Simon	96	64	64	42	21	[6]
Simon	64	32	32	32	16	[6]
Prince	128	64	64	24	30	[26]
KATAN64	80	64	64	74	36	[16]
KATAN48	80	48	48	74	32	[16]
KATAN32	80	32	32	64	24	[16]
DES	56	64	56	261	284	[65]
LowMC	80	256	64	11	**6.31**	full version

Table 2: Comparison of ciphers (excluding key schedule). We list the depth-optimized variants; size-optimized variants are given in () if available. Best in class are marked in bold.

desired security level in terms of time and data, and block size.

5.1 Differential characteristics

In differential attacks, the principal goal is to find a pair (α, β) of an input difference α and an output difference β for the cipher such that pairs of input texts with difference α have an unusual high probability to produce output texts with difference β. Such a pair of differences is called a *differential*. A good differential can be used to mount distinguishing attacks as well as key recovery attacks on the cipher. For this it suffices if the differential does not cover the whole cipher but all except one or a few rounds.

As it is infeasible to calculate the probability of differentials for most ciphers, the cryptanalyst often has to be content with finding good *differential characteristics* i.e., paths of differences through the cipher for which the probability can directly be calculated. Note that a differential is made up of all differential characteristics that have the same input and output difference as the differential. The probability of a good differential characteristic is thus a lower bound for the related differential.

Allowing parts of the state to go unchanged through the Sbox layer clearly increases the chance of good differential characteristics. It is for example always possible to find a one round characteristic of probability 1. In fact, it is even possible to find $\lceil \frac{l}{3m} \rceil$-round characteristics of probability 1 where l is the width of the identity part

and m the number of 3-bit Sboxes. Nonetheless, as we will prove in the following, this poses no threat. This is because of the randomness of the linear layer which maps a fixed subspace to a random subspace of the same dimension: Most "good" difference i.e., differences that activate none or only few Sboxes, are mapped to "bad" differences that activate most of the Sboxes per layer. This causes the number of characteristics that only use "good" differences to decay exponentially with the number of rounds. In the case of a $\lceil \frac{l}{3m} \rceil$-round characteristic of probability 1, this means that the output difference is fixed to very few options, which makes it then already in the next round extremely unlikely that any one of the options is mapped onto a "good" difference.

We will now prove that good differential characteristics exist only with negligible probability in LowMC. The basic idea behind the proof is the following. We calculate for each possible good differential characteristic the probability that it is realized in an instantiation of LowMC under the assumption that the binary matrices of the linear layer were chosen independently and uniformly at random. We then show that the sum of these probabilities, which is an upper bound for the probability that any good characteristic exists, is negligible.

Recall that m is the number of Sboxes in one Sbox layer in LowMC and that l is the bit-length of the identity part of the Sbox layer. We thus have $n = 3m + l$. Let $V(i)$ be the number of bit vectors of length n that correspond to a difference that activates i Sboxes. As we can choose i out of the m Sboxes, as for each active 3-bit Sbox there are 7 possible non-zero input differences and as the bits of the identity part can be chosen freely, we have

$$V(i) = \binom{m}{i} \cdot 7^i \cdot 2^l \quad . \tag{1}$$

Let α_0 be an input difference and let α_1 be an output difference for one round of LowMC. Let a_0 be the number of Sboxes activated by α_0. As an active Sbox maps its non-zero input difference to four possible output differences each with probability $\frac{1}{4}$, and as a uniformly randomly chosen invertible binary $n \times n$ matrix maps a given non-zero n-bit vector with probability $\frac{1}{2^n-1}$ to another given non-zero output vector, the probability that the one-round characteristic (α_0, α_1) has a probability larger than 0 is

$$\frac{4^{a_0}}{2^n - 1} \quad . \tag{2}$$

Let $(\alpha_0, \alpha_1, \ldots, \alpha_r)$ now be a given characteristic over r rounds where the differences α_i are at the end of round i and α_0 is the starting difference. Let $(a_0, a_1, \ldots, a_{r-1})$ be the numbers of Sboxes activated by each α_0, α_1,..., and α_{r-1}. We can now calculate the probability that this characteristic has a probability larger than 0 in a random instantiation of LowMC as

$$\frac{4^{a_0}}{2^n - 1} \cdot \frac{4^{a_1}}{2^n - 1} \cdots \frac{4^{a_{r-1}}}{2^n - 1} = \frac{4^{a_0 + a_1 + \cdots + a_{r-1}}}{(2^n - 1)^r} \quad . \tag{3}$$

Rounds	1 - 6	7	8	9	10	11	12	13	14	15
$n = 256$	1.0	2^{-100}	2^{-212}	2^{-326}	2^{-442}	2^{-558}	2^{-676}	2^{-794}	2^{-913}	-
$n = 1024$	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2^{-26}	2^{-145}	2^{-264}

Table 3: Example of how the probability bound p_{stat}, for the existence of differential or linear characteristic of probability at least 2^{-d}, evolves. The parameters are here $m = 42$, $d = 128$.

Summing now over all possible characteristics over r rounds that activate at most d Sboxes, we can calculate an upper bound for the probability that there exists an r-round characteristic with d or fewer active Sboxes as

$$\sum_{\substack{0 \leq a_0, a_1, \ldots, a_{r-1} \leq m \\ a_0 + a_1 + \cdots + a_{r-1} \leq d}} V(a_0) \cdot V(a_1) \cdots V(a_{r-1}) \cdot (2^n - 1) \cdot \frac{4^{a_0 + a_1 + \cdots + a_{r-1}}}{(2^n - 1)^r} \tag{4}$$

where the factor $(2^n - 1)$ is the number of choices for the last difference α_r that can take any non-zero value.

With the knowledge that each active Sbox reduces the probability of a characteristic by a factor of 2^{-2}, we can now calculate for each parameter set of LowMC the number of rounds after which no good differentials are present except for a negligible probability. We consider as good differential characteristics those with a probability higher than 2^{-d}, where d is the allowed data complexity in the respective parameter set. We call a negligible probability a probability lower than 2^{-100}. Note that this probability only comes into play once when fixing an instantiation of LowMC. The calculated bound for our choice of parameters can be found in Table 4.

5.2 Linear characteristics

In linear cryptanalysis [55], the goal of the cryptanalyst is to find affine approximations of the cipher that hold sufficiently well. As with differential cryptanalysis, these can be used to mount distinguishing and key recovery attacks. The approximation is done by finding so-called *linear characteristics*, a concatenation of linear approximations for the consecutive rounds of the cipher. Similar to differential characteristics, linear characteristics activate Sboxes that are involved in the approximations.

The proof for the absence of good differential characteristics is directly transferable to linear characteristics because of two facts. Firstly, the maximal linear probability of the Sbox is 2^{-2}, just the same as the maximal differential probability. Secondly, the transpose of a uniformly randomly chosen invertible binary matrix is still a uniformly randomly chosen invertible binary matrix. Thus we can use equation 4 to calculate the bounds for good linear characteristics as well.

5.3 Boomerang attacks

In boomerang attacks [66], good partial differential characteristics that cover only part of the cipher can be combined to attack ciphers that might be immune to standard differential cryptanalysis. In these attacks, two differential characteristics are combined, one that covers the first half of the cipher and another that covers the second half. If both have about the same probability, the complexity corresponds roughly to the inverse of the fourth power of this probability [66]. Thus to calculate the number of rounds sufficient to make sure that no boomerang exists, we calculate the number of rounds after which differential characteristics of probability $2^{-d/4}$ exist only with negligible probability and then double this number.

5.4 Higher order attacks

Due to its small size, the degree of the Sbox in its algebraic representation is only two. Since in one round the Sboxes are applied in parallel and since the affine layer does not change the algebraic degree, the algebraic degree of one round is two as well. As a low degree could be used as a lever for a high-order attack, let us take a look at how the algebraic degree of LowMC develops over several rounds.

Clearly the algebraic degree of the cipher after r rounds is bounded from above by 2^r. It is furthermore generally bounded from above by $n-1$ since the cipher is a permutation. A second upper bound, that is better suited and certainly more realistic for the later rounds, was found by Boura et al. [9]. In our case it is stated as following: If the cipher has degree d_r after r rounds, the degree after round $r+1$ is at most $\frac{n}{2}+\frac{d_r}{2}$. Differing from Boura et al. [9], in LowMC the Sbox layer only partially consists of Sboxes and partially of the identity mapping. This must be accounted for and requires a third bound: If the cipher has degree d_r after r rounds, the degree after round $r+1$ is at most $m+d_r$. A proof of this can be found in the full version of this paper. This can be summarized as follows:

Lemma 1. *If the algebraic degree of LowMC with m Sboxes and length l of the identity part in the Sbox layer is d_r after r rounds, the degree in round $r+1$ is at most*

$$\min\left(2d_r, m+d_r, \frac{n}{2}+\frac{d_r}{2}\right) \tag{5}$$

where $n=3m+l$ is the block width of LowMC.

Combining these three bounds, we can easily calculate lower bounds for the number of rounds r needed for different parameter sets l and m of LowMC to reach a degree that is at least as large as the allowed data complexity d minus 1. The results of this for LowMC's parameters are displayed in Table 4.

5.5 Experimental Cryptanalysis

We proved that no good differential or linear characteristic can cover sufficiently many rounds to be usable as an attack vector in LowMC. This does not exclude

Sboxes	blocksize	data complexity	r_{stat}	r_{bmrg}	r_{deg}
49	256	64	5	6	6
63	256	128	5	6	7

Table 4: For the different sets of LowMC parameters, bounds are given for the number of rounds for which no good differential or linear characteristics exist (r_{stat}), to avoid good boomerangs (r_{bmrg}), and the number of rounds needed to have a sufficiently high algebraic degree (r_{deg}). The bounds were calculated using equations 4 and 5.

though the possibility of good differentials or linear hulls for which a large number of characteristics combine. Given the highly diffusive, random linear layers, this seems very unlikely.

Likewise we were able to find lower bounds on the number of rounds needed for the algebraic degree of LowMC to be sufficiently high. Even though this is state-of-the art also for traditional designs to date, this gives us no guarantee that it will indeed be high. Unfortunately it is not possible to directly calculate the algebraic degree for any large block size.

To nevertheless strengthen our confidence in the design, we numerically examined the properties of small-scale versions of LowMC. In table 5, we find the results for a 24-bit wide version with 4 Sboxes. For testing its resistance against differential cryptanalysis, we calculated the full codebook under 100 randomly chosen keys and used the distribution of differences to estimate the probabilities of the differentials. To reduce the computational complexity, we restricted the search space to differentials with one active bit in the input difference.

It can clearly be seen that the probability of differentials quickly saturates to values too low to allow an attack. Clearly, the bound calculated with equation 4 (p_{stat} in the table) overestimates the probability of good characteristics. Even though we were not able to search the whole space of differentials there is little reason to assume that there are other differentials that fare considerably better. It is important to note that the number of impossible differentials goes to 0 after only few rounds. Thus impossible differentials cannot be used to attack any relevant number of rounds. At the same time this assures the absence of any truncated differentials of probability 1.

The minimal algebraic degree[5] is tight for this version when compared with the theoretic upper bound as determined with equation 5. More experimental cryptanalysis can be found in the full version of this paper.

[5]That is the minimum of the algebraic degrees of the 24 output bit when written as Boolean functions.

Rounds	p_{best}	p_{worst}	n_{imposs}	$\deg_{\text{exp}}$	$\deg_{\text{theor}}$	p_{stat}
2	$2^{-8.64}$	0	$2^{28.58}$	4	4	-
3	$2^{-12.64}$	0	$2^{28.00}$	8	8	-
4	$2^{-14.64}$	0	$2^{4.25}$	12	12	-
5	$2^{-18.60}$	$2^{-26.06}$	0	16	16	-
6	$2^{-20.49}$	$2^{-25.84}$	0	20	20	-
7	$2^{-23.03}$	$2^{-25.74}$	0	22	22	-
8	$2^{-23.06}$	$2^{-25.74}$	0	23	23	-
10	-	-	-	-	-	$2^{-5.91}$
11	-	-	-	-	-	$2^{-16.00}$
12	-	-	-	-	-	$2^{-26.28}$
19	-	-	-	-	-	$2^{-101.5}$

(a) $n = 24$, $m = 4$, $k = 12$, $d = 12$

Table 5: Experimental results of full codebook encryption over 100 random keys for a set of small parameters are given. p_{best} and p_{worst} are the best and the worst approximate differential probability of any differential with one active bit in the input difference. n_{imposs} is the number of impossible differentials with one active bit in the input difference. $\deg_{\text{exp}}$ is the minimal algebraic degree in any of the output bits. $\deg_{\text{theor}}$ is the upper bound for the algebraic degree as determined from equation 5. p_{stat} is the probability that a differential or linear characteristic of probability at least 2^{-12} exists (see eq. 4).

5.6 Fixing the number of rounds

We base our recommendation for the number of rounds on the following:

$$r \geq \max(r_{\text{stat}}, r_{\text{bmrg}}, r_{\text{deg}}) + r_{\text{outer}}$$

where r_{stat} is a bound for statistical attack vectors such as differentials and linear characteristics as discussed in Section 5.1, r_{bmrng} is the bound for boomerang attacks as discussed in Section 5.3, and where r_{deg} indicates the number of rounds needed for the cipher to have sufficient degree as discussed in Section 5.4. Values of these for the parameters of LowMC can be found in Table 4. For the number of rounds which can be peeled off at the beginning and end of the cipher by key guessing and other strategies, we use the ad-hoc formular $r_{\text{outer}} = r_{\text{stat}}$.

6 Comparison of Implementations

In the following we report on experiments when evaluating LowMC with MPC protocols in Section 6.1 and with FHE in Section 6.2. The performance of both implementations is independent of the specific choice of the random matrices and

vectors used in LowMC (cf. Section 3.3) as we do not use any optimizations that are based on their specific structure.

In both the FHE and MPC settings, for more efficient matrix multiplication, we use a method that is generically better than a naive approach: the "method of the four Russians" [1]. This method reduces the complexity of the matrix-vector product from $O(n^2)$ to $O(n^2/log(n))$, i.e. it's an asymptotically faster algorithm and is also fast in practice for the dimensions we face in LowMC. Asymptotically faster methods like the Strassen-Winograd method method make no sense however, for the dimensions we are considering.

It turns out that considering design-optimizations of the linear layer by introducing structure and thereby lowering the density of the matricies and in turn reducing the number of XOR computations will not improve performance of all these implementations. On the contrary, as the application of the security analysis suggests, the number of rounds would need to be increased in such a case.

6.1 MPC Setting

As an example for both classes of MPC protocols described in Section 2.1 we use the GMW protocol [34] in the semi-honest setting. As described in [18], this protocol can be partitioned into 1) a *setup phase* with a constant number of rounds and communication linear in the MC of the circuit (2κ bits per AND gate for κ-bit security), and 2) an *online phase* whose round complexity is linear in the ANDdepth of the circuit. Hence, we expect that the setup time grows linearly in the MC while the online time grows mostly with increasing ANDdepth when network latency is high.

Benchmark Settings.

For our MPC experiments we compare LowMC against other ciphers with a comparable level of security. We compare LowMC with the two standardized ciphers Present and AES and also with the NSA cipher Simon which previously had the lowest number of ANDs per encrypted bit (cf. Table 2). More specifically, for lightweight security with at least $\kappa = 80$ bit security we compare LowMC with 80 bit keys against Present with 80 bit key (using the Sbox of [19]) and Simon with 96 bit keys (the Simon specification does not include a variant with 80 bit keys); for long-term security with $\kappa = 128$ bit security we compare LowMC with 128 bit keys against AES-128 (using the Sbox of [12]) and Simon with 128 bit key; we set the security parameters for the underlying MPC protocol to $\kappa = 80$ bit for lightweight security and to $\kappa = 128$ bit for long-term security. We exclude the key schedule and directly input the pre-computed round keys. We use the GMW implementation that is available in the ABY-framework [23] which uses the efficient oblivious transfer extensions of [3][6]. We run our MPC experiments on two desktop PCs, each equipped

[6] Our the MPC implementations of the benchmarked block-ciphers are available online as part of the ABY-framework `https://github.com/encryptogroup/ABY`.

Lightweight Security						
Cipher	Present		Simon		LowMC	
Communication [kB]	39		**26**		51	
Runtime	LAN	WAN	LAN	WAN	LAN	WAN
Setup [s]	0.003	0.21	**0.002**	0.21	**0.002**	**0.14**
Online [s]	**0.05**	13.86	**0.05**	5.34	0.06	**1.46**
Total [s]	**0.05**	14.07	**0.05**	5.45	0.06	**1.61**
Long-Term Security						
Cipher	AES		Simon		LowMC	
Communication [kB]	170		136		**72**	
Runtime	LAN	WAN	LAN	WAN	LAN	WAN
Setup [s]	0.01	0.27	0.009	0.23	**0.002**	**0.15**
Online [s]	**0.04**	4.08	0.05	6.95	0.07	**1.87**
Total [s]	**0.05**	4.35	0.06	7.18	0.07	**2.02**

Table 6: GMW benchmarking results for single block. Best in class marked in bold.

with an Intel Haswell i7-4770K CPU with 3.5 GHz and 16GB of RAM, that are connected by Gigabit LAN. To see the impact of the reduced ANDdepth in the online phase, we measured the times in a LAN scenario (0.2 ms latency) and also a trans-atlantic WAN scenario (50 ms latency) which we simulated using the Linux command `tc`.

In our first experiment depicted in Table 6 we encrypt a single block, whereas in our second experiment depicted in Table 7 we encrypt multiple blocks in parallel to encrypt 12.8 Mbit of data.

Single-Block Results.

From our single-block experiments in Table 6 we see that the communication of LowMC is higher by factor 2 compared to the lightweight security ciphers but lower by factor 2 compared to the long-term security ciphers. In terms of total runtime, for lightweight security LowMC performs similar to Present and Simon in the LAN setting and outperforms both by factor 3 to 9 in the WAN setting. For long-term security AES is slightly faster than LowMC in the LAN setting, but slower than LowMC in the WAN setting by factor 2. These results can be explained by the high number of XOR gates of LowMC compared to AES, which impact the run-time higher than the communication for the AND gates. In the WAN setting, the higher ANDdepth of AES outweighs the local overhead of the XOR gates for LowMC, yielding a faster run-time for LowMC.

Multi-Block Results.

From our multi-block experiments in Table 7 we see that LowMC needs less communication than all other ciphers: at least factor 2 for lightweight security and factor 4 for long-term security. Also the total runtime of LowMC is the lowest among all ciphers, ranging from factor 6 when compared to Simon for lightweight security to factor 9 when compared to AES for long-term security.

Lightweight Security						
Cipher	Present		Simon		LowMC	
Comm. [GB]	7.4		5.0		**2.5**	
Runtime	LAN	WAN	LAN	WAN	LAN	WAN
Setup [s]	214.17	453.89	268.93	568.35	**43.33**	**138.63**
Online [s]	2.71	34.35	3.29	37.06	**2.02**	**17.12**
Total [s]	216.88	488.24	272.22	605.41	**45.36**	**155.75**
Long-Term Security						
Cipher	AES		Simon		LowMC	
Comm. [GB]	16		13		**3.5**	
Runtime	LAN	WAN	LAN	WAN	LAN	WAN
Setup [s]	553.41	914.27	444.30	727.48	**62.01**	**193.90**
Online [s]	2.50	33.52	2.97	34.42	**2.36**	**21.11**
Total [s]	555.91	947.79	447.27	761.90	**64.37**	**215.01**

Table 7: GMW benchmarking results for multiple blocks to encrypt 12.8 Mbit of data. Best in class marked in bold.

Summary of the Results.

To summarize our MPC experiments, the benefits of LowMC w.r.t. the *online time* depend on the network latency: over the low-latency LAN network existing ciphers achieve comparable or even slightly faster online runtimes than LowMC, whereas in the higher latency WAN network LowMC achieves the fastest online runtime. W.r.t. the *total runtime*, LowMC's benefit in the single-block application again depends on the latency (comparable or slightly less efficient over LAN, but more efficient over WAN), whereas in the multi-block application LowMC significantly improves over existing ciphers by factor 6 to 9. For secure computation protocols with security against malicious adversaries, the benefit of using LowMC would be even more significant, since there the costs per AND gate are at least an order of magnitude higher than in the semi-honest GMW protocol, cf. [48], [58].

6.2 FHE Setting

We implemented LowMC using the homomorphic encryption library HELib [37], [38], which implements the BGV homomorphic encryption scheme [14] and which was also used to evaluate AES-128 [31], [32]. Our implementation represents each plaintext, ciphertext and key bits as individual HE ciphertexts on which XOR and AND operations are performed. Due to the nature of the BGV system this means that we can evaluate many such instances in parallel, typically a few hundred. We found this representation to be more efficient than our other "compact" implementation which packs these bits into the slots of HE ciphertexts.

In the homomorphic encryption setting the number of AND gates is not the main determinant of complexity. Instead, the ANDdepth of the circuit largely determines the cost of XOR and AND, where AND is more expensive than XOR. However, due to the high number of XORs in LowMC, the cost of the linear layer is not negligible. In our implementation we use the "method of the four Russians" [1] to reduce the number of HE ciphertext additions from $\mathcal{O}(n^2)$ to $\mathcal{O}(n^2/\log(n))$.

d	m	r	n	#blocks	t_{setup}	t_{eval}	t_{sbox}	t_{key}	t_{block}	t_{bit}	Memory	Comment
128	63	12	256	600	11.6	506.1	353.2	1.6	0.8434	0.0033	1.58GB	main
128	86	11	512	600	11.7	847.6	451.5	3.2	1.4127	0.0028	2.62GB	perf
128	86	12	512	600	11.7	893.9	480.1	3.2	1.4898	0.0029	2.62GB	cons
64	49	11	256	600	11.0	383.0	206.3	0.9	0.6383	0.0025	1.52GB	main
64	49	10	256	600	11.5	305.6	255.6	1.1	0.5093	0.0020	1.37GB	perf
64	34	11	128	600	13.0	260.7	204.0	0.7	0.4345	0.0034	1.08GB	smll

Table 8: LowMC (commit `f6a086e`) in HElib [38] (commit `e9d3785e`) on Intel i7-4850HQ CPU @ 2.30GHz; d is the allowed data complexity, m is the number of Sboxes, n is the blocksize, r is the number of rounds, # blocks is the number of blocks computed in parallel, t_{setup} is the total setup time, t_{eval} is the total running time of the encryption in seconds, t_{sbox} the total time spent in the S-Box layer in seconds, t_{key} the total time spent in the key schedule in seconds, $t_{block} = t_{eval}/\#\text{blocks}$ and $t_{bit} = t_{block}/n$. The rows marked as "main" contain the main parameter proposals. The rows marked as "perf", "cons" or "smll" contain alternative parameter sets being conservative, performance oriented or relatively small respectively.

In our experiments we chose the depth for the homomorphic encryption scheme such that the "base level" of fresh ciphertexts is at least the number of rounds, i.e. we consume one level per round. Our implementation also does not precompute round keys in advance, but deriving round keys is considered part of the evaluation (cost).

We consider LowMC instances for Present-80 and AES-128 like security. We always choose a homomorphic encryption security level of 80 for compatibility with [31]. Our results are given in Table 8. Our implementation is available at `https://bitbucket.org/malb/lowmc-helib`.

For comparison with previous results in the literature we reproduce those results in Table 9 which demonstrates the benefit of a dedicated block cipher for homomorphic evaluation.

7 Conclusions, lessons learned, and open problems

We proposed block ciphers with an extremely small number of AND gates and an extremely shallow AND depth, demonstrated the soundness of our design through experimental evidence and provided a security analysis of these constructions. Of course, as with any other block cipher, more security analysis is needed to firmly establish the security provided by this new design. Furthermore, with the proposal of the LowMC familiy, we bring together the areas of symmetric cryptographic design and analysis research with new developments around MPC and FHE. Finally, in contrast to current folklore belief, in some implementation scenarios, we identified practical cases where "free XORs" can no longer be considered free and where local computations in an MPC protocol represent a considerable bottleneck.

d	ANDdepth	#blocks	t_{eval}	t_{block}	t_{bit}	Cipher	Reference	Key Schedule
128	40	120	3m	1.5s	0.0119s	AES-128	[31]	excluded
128	40	2048	31h	55s	0.2580s	AES-128	[25]	excluded
128	40	1	22m	22m	10.313s	AES-128	[57]	excluded
128	40	12	2h47m	14m	6.562s	AES-128	[57]	excluded
128	12	600	8m	0.8s	0.0033s	LowMC	this work	included
64	24	1024	57m	3.3s	0.0520s	PRINCE	[26]	excluded
64	11	600	6.4m	0.64s	0.0025s	LowMC	this work	included

Table 9: Comparison of various block cipher evaluations in the literature and this work; Notation as in Table 8. Memory requirements are not listed as they are usually not provided in the literature. The first row is based on experimental data obtained on the same machine and the same instance of HELib as in Table 8.

To finish, we highlight a number of open problems related to the LowMC family of ciphers. Is it possible to reduce the number of rounds in LowMC further, which in turn would further reduce MC and ANDdepth? Analyzing such an extreme corner of the design space for a symmetric cipher is an interesting endavor in itself. Can we add more structure into the linear layers in order to reduce the necessary computational effort in those cases where the number of AND gates is no longer the bottleneck? Do such approaches beat applying asymptotically faster linear algebra techniques for applying linear layers as done in Section 6? As we argue in the paper, simply lowering the density of the matrices by several factors of two will not be enough.

Currently, the MC and ANDdepth of various cipher constructions is poorly understood. For example, it would be interesting to find efficient algorithms along the lines of [11] for the various ciphers including the recent lightweight cipher proposals in the literature. While our choice for GF(2) is well motivated, there are scenarios where larger fields might be beneficial. What designs minimize MC and ANDdepth under such constraints?

Acknowledgements.

We thank Dmitry Khovratovich for pointing to us out that an earlier version of our parameter sets for LowMC instantiations are too optimistic. See the full version of this paper for details. We thank Orr Dunkelman for helpful clarifications on the cipher KATAN. We thank Gaëtan Leurent and François-Xavier Standaert for helpful clarifications regarding the ciphers Fantomas and Robin.

The work of Albrecht was supported by EPSRC grant EP/L018543/1 “Multilinear Maps in Cryptography”. The work of co-authors from TU Darmstadt was supported by the European Union’s 7th Framework Program (FP7/2007-2013) under grant agreement n. 609611 (PRACTICE), by the DFG as part of project E3 within the CRC 1119 CROSSING, by the German Federal Ministry of Education and Research (BMBF) within EC SPRIDE, and by the Hessian LOEWE excellence initiative within

CASED.

References

[1] Martin R. Albrecht, Gregory V. Bard, and William Hart. "Algorithm 898: Efficient multiplication of dense matrices over GF(2)". In: *ACM Transactions on Mathematical Software* 37.1 (2010). URL: `http://dblp.uni-trier.de/db/journals/toms/toms37.html#AlbrechtBH10`.

[2] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. "Cryptography in NC^0". In: *SIAM Journal on Computing* 36.4 (2006), pp. 845–888.

[3] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. "More Efficient Oblivious Transfer and Extensions for Faster Secure Computation". In: *Computer and Communications Security (CCS)*. Code: `http://github.com/MichaelZohner/OTExtension`. ACM, 2013, pp. 535–548.

[4] Yonatan Aumann and Yehuda Lindell. "Security Against Covert Adversaries: Efficient Protocols for Realistic Adversaries". In: *Theory of Cryptography Conference (TCC)*. Vol. 4392. LNCS. Springer, 2007, pp. 137–156.

[5] Achiya Bar-On, Itai Dinur, Orr Dunkelman, Virginie Lallemand, and Boaz Tsaban. *Cryptanalysis of SP Networks with Partial Non-Linear Layers*. Cryptology ePrint Archive, Report 2014/228. `http://eprint.iacr.org/2014/228`. 2014.

[6] 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. `http://eprint.iacr.org/2013/404`. 2013.

[7] Elwyn R. Berlekamp, Robert J. McEliece, and Henk C. A. van Tilborg. "On the inherent intractability of certain coding problems (Corresp.)" In: *IEEE Transactions on Information Theory* 24.3 (1978), pp. 384–386.

[8] 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: *Advances in Cryptology – ASIACRYPT*. Vol. 7658. LNCS. Springer, 2012, pp. 208–225.

[9] Christina Boura, Anne Canteaut, and Christophe De Cannière. "Higher-Order Differential Properties of Keccak and Luffa". In: *Fast Software Encryption (FSE)*. Vol. 6733. LNCS. Springer, 2011, pp. 252–269.

[10] Joan Boyar, Ivan Damgård, and René Peralta. "Short Non-Interactive Cryptographic Proofs". In: *Journal of Cryptology* 13.4 (Dec. 2000), pp. 449–472.

[11] Joan Boyar, Philip Matthews, and René Peralta. "Logic Minimization Techniques with Applications to Cryptology". In: *Journal of Cryptology* 26.2 (2013), pp. 280–312.

[12] Joan Boyar and René Peralta. "A Small Depth-16 Circuit for the AES S-Box". In: *Information Security and Privacy Conference (SEC)*. Vol. 376. IFIP Advances in Information and Communication Technology. Springer, 2012, pp. 287–298.

[13] Joan Boyar, René Peralta, and Denis Pochuev. "On the multiplicative complexity of Boolean functions over the basis $(\wedge, \oplus, 1)$". In: *Theoretical Computer Science* 235.1 (2000), pp. 43–57.

[14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. "Fully Homomorphic Encryption without Bootstrapping". In: *Electronic Colloquium on Computational Complexity (ECCC)* 18 (2011), p. 111.

[15] Gilles Brassard and Claude Crépeau. "Zero-Knowledge Simulation of Boolean Circuits". In: *Advances in Cryptology – CRYPTO*. Vol. 263. LNCS. Sringer, 1986, pp. 223–233.

[16] Christophe De Cannière, Orr Dunkelman, and Miroslav Knezevic. "KATAN and KTANTAN - A Family of Small and Efficient Hardware-Oriented Block Ciphers". In: *Cryptographic Hardware and Embedded Systems (CHES)*. Vol. 5747. LNCS. Springer, 2009, pp. 272–288.

[17] David Canright. "A Very Compact S-Box for AES". In: *Cryptographic Hardware and Embedded Systems (CHES)*. Vol. 3659. LNCS. Springer, 2005, pp. 441–455.

[18] Seung G. Choi, Kyung-Wook Hwang, Jonathan Katz, Tal Malkin, and Dan Rubenstein. "Secure Multi-Party Computation of Boolean Circuits with Applications to Privacy in On-Line Marketplaces". In: *Cryptographers' Track at the RSA Conference (CT-RSA)*. Vol. 7178. LNCS. Code: `http://www.ee.columbia.edu/~kwhwang/projects/gmw.html`. Springer, 2012, pp. 416–432.

[19] Nicolas T. Courtois, Daniel Hulme, and Theodosis Mourouzis. *Solving Circuit Optimisation Problems in Cryptography and Cryptanalysis*. Cryptology ePrint Archive, Report 2011/475. `http://eprint.iacr.org/2011/475`. 2011.

[20] Joan Daemen, Michaël Peeters, Gilles Van Assche, and Vincent Rijmen. "Nessie proposal: NOEKEON". In: *First Open NESSIE Workshop*. 2000.

[21] Ivan Damgård, Rasmus Lauritsen, and Tomas Toft. "An Empirical Study and Some Improvements of the MiniMac Protocol for Secure Computation". In: *Security and Cryptography for Networks (SCN)*. Vol. 8642. LNCS. Springer, 2014, pp. 398–415.

[22] Ivan Damgård and Sarah Zakarias. "Constant-Overhead Secure Computation of Boolean Circuits using Preprocessing". In: *Theory of Cryptology Conference (TCC)*. Vol. 7785. LNCS. Springer, 2013, pp. 621–641.

[23] Daniel Demmler, Thomas Schneider, and Michael Zohner. "ABY - A Framework for Efficient Mixed-Protocol Secure Two-Party Computation". In: *Network and Distributed System Security (NDSS'15)*. Code: `https://github.com/encryptogroup/ABY`. The Internet Society, 2015.

[24] Itai Dinur and Adi Shamir. "Cube Attacks on Tweakable Black Box Polynomials". In: *Advances in Cryptology – EUROCRYPT*. Vol. 5479. LNCS. Springer, 2009, pp. 278–299.

[25] Yarkin Doroz, Yin Hu, and Berk Sunar. *Homomorphic AES Evaluation using NTRU*. Cryptology ePrint Archive, Report 2014/039. `http://eprint.iacr.org/2014/039`. 2014.

[26] Yarkın Doröz, Aria Shahverdi, Thomas Eisenbarth, and Berk Sunar. *Toward Practical Homomorphic Evaluation of Block Ciphers Using Prince*. Cryptology ePrint Archive, Report 2014/233. `http://eprint.iacr.org/2014/233`, presented at Workshop on Applied Homomorphic Cryptography and Encrypted Computing (WAHC'14). 2014.

[27] Niels Ferguson, Stefan Lucks, Bruce Schneier, Doug Whiting, Mihir Bellare, Tadayoshi Kohno, Jon Callas, and Jesse Walker. *The Skein Hash Function Family*. Submission to NIST (Round 3). 2010. URL: `http://www.skein-hash.info/sites/default/files/skein1.3.pdf`.

[28] Tore K. Frederiksen, Thomas P. Jakobsen, and Jesper B. Nielsen. "Faster Maliciously Secure Two-Party Computation Using the GPU". In: *Security and Cryptography for Networks (SCN)*. Vol. 8642. LNCS. Springer, 2014, pp. 358–379.

[29] Tore K. Frederiksen, Thomas P. Jakobsen, Jesper B. Nielsen, Peter S. Nordholt, and Claudio Orlandi. "MiniLEGO: Efficient Secure Two-Party Computation from General Assumptions". In: *Advances in Cryptology – EUROCRYPT*. Vol. 7881. LNCS. Springer, 2013, pp. 537–556.

[30] Tore K. Frederiksen and Jesper B. Nielsen. "Fast and Maliciously Secure Two-Party Computation Using the GPU". In: *Applied Cryptography and Network Security (ACNS)*. Vol. 7954. LNCS. Springer, 2013, pp. 339–356.

[31] Craig Gentry, Shai Halevi, and Nigel P. Smart. *Homomorphic Evaluation of the AES Circuit*. Cryptology ePrint Archive, Report 2012/099. `http://eprint.iacr.org/`. 2012.

[32] Craig Gentry, Shai Halevi, and Nigel P. Smart. "Homomorphic Evaluation of the AES Circuit". In: *Advances in Cryptology – CRYPTO*. Vol. 7417. LNCS. Springer, 2012, pp. 850–867.

[33] Benoît Gérard, Vincent Grosso, María Naya-Plasencia, and François-Xavier Standaert. "Block Ciphers That Are Easier to Mask: How Far Can We Go?" In: *Cryptographic Hardware and Embedded Systems (CHES)*. Vol. 8086. LNCS. Springer, 2013, pp. 383–399.

[34] Oded Goldreich, Silvio Micali, and Avi Wigderson. "How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority". In: *Symposium on Theory of Computing (STOC)*. ACM, 1987, pp. 218–229.

[35] Vicente Grosso, Gaëtan Leurent, François-Xavier Standaert, and Kerem Varici. "LS-Designs: Bitslice Encryption for Efficient Masked Software Implementations". In: *Fast Software Encryption (FSE)*. LNCS. To appear. Springer, 2014.

[36] Jian Guo, Ivica Nikolic, Thomas Peyrin, and Lei Wang. *Cryptanalysis of Zorro*. Cryptology ePrint Archive, Report 2013/713. `http://eprint.iacr.org/2013/713`. 2013.

[37] Shai Halevi and Victor Shoup. "Algorithms in HElib". In: *Advances in Cryptology – CRYPTO*. Vol. 8616. LNCS. Springer, 2014, pp. 554–571.

[38] Shai Halevi and Victor Shoup. *Design and Implementation of a Homomorphic-Encryption Library*. `https://github.com/shaih/HElib/`. 2013.

[39] Martin Hell, Thomas Johansson, Alexander Maximov, and Willi Meier. "The Grain Family of Stream Ciphers". In: *New Stream Cipher Designs - The eSTREAM Finalists*. Ed. by Matthew J. B. Robshaw and Olivier Billet. Vol. 4986. Lecture Notes in Computer Science. Springer, 2008, pp. 179–190. ISBN: 978-3-540-68350-6. DOI: `10.1007/978-3-540-68351-3_14`. URL: `http://dx.doi.org/10.1007/978-3-540-68351-3_14`.

[40] Yan Huang, David Evans, Jonathan Katz, and Lior Malka. "Faster secure two-party computation using garbled circuits". In: *USENIX Security*. USENIX, 2011.

[41] Yan Huang, Jonathan Katz, and David Evans. "Efficient Secure Two-Party Computation Using Symmetric Cut-and-Choose". In: *Advances in Cryptology – CRYPTO*. Vol. 8043. LNCS. Springer, 2013, pp. 18–35.

[42] Yan Huang, Jonathan Katz, Vladimir Kolesnikov, Ranjit Kumaresan, and Alex J. Malozemoff. "Amortizing Garbled Circuits". In: *Advances in Cryptology – CRYPTO*. Vol. 8617. LNCS. Springer, 2014, pp. 458–475.

[43] Marek Jawurek, Florian Kerschbaum, and Claudio Orlandi. "Zero-knowledge Using Garbled Circuits: How to Prove Non-algebraic Statements Efficiently". In: *Computer and Communications Security (CCS)*. ACM, 2013, pp. 955–966.

[44] Lars R. Knudsen. "Truncated and Higher Order Differentials". In: *Fast Software Encryption (FSE)*. Vol. 1008. LNCS. Springer, 1994, pp. 196–211.

[45] Lars R. Knudsen, Gregor Leander, Axel Poschmann, and Matthew J. B. Robshaw. "PRINTcipher: A Block Cipher for IC-Printing". In: *Cryptographic Hardware and Embedded Systems (CHES)*. Vol. 6225. LNCS. Springer, 2010, pp. 16–32.

[46] Vladimir Kolesnikov and Thomas Schneider. "Improved Garbled Circuit: Free XOR Gates and Applications". In: *International Colloquium on Automata, Languages and Programming (ICALP)*. Vol. 5126. LNCS. Springer, 2008, pp. 486–498.

[47] Benjamin Kreuter, Abhi Shelat, and Chih-Hao Shen. "Billion-gate Secure Computation with Malicious Adversaries". In: *USENIX Security*. USENIX, 2012.

[48] Enrique Larraia, Emmanuela Orsini, and Nigel P. Smart. "Dishonest Majority Multi-Party Computation for Binary Circuits". In: *Advances in Cryptology – CRYPTO*. Vol. 8617. LNCS. Springer, 2014, pp. 495–512.

[49] Yehuda Lindell. "Fast Cut-and-Choose Based Protocols for Malicious and Covert Adversaries". In: *Advances in Cryptology – CRYPTO*. Vol. 8043. LNCS. Springer, 2013, pp. 1–17.

[50] Yehuda Lindell and Benny Pinkas. "An Efficient Protocol for Secure Two-Party Computation in the Presence of Malicious Adversaries". In: *Advances in Cryptology – EUROCRYPT*. Vol. 4515. LNCS. Springer, 2007, pp. 52–78.

[51] Yehuda Lindell and Benny Pinkas. "Secure Two-Party Computation via Cut-and-Choose Oblivious Transfer". In: *Theory of Cryptography Conference (TCC)*. Vol. 6597. LNCS. Springer, 2011, pp. 329–346.

[52] Yehuda Lindell, Benny Pinkas, and Nigel P. Smart. "Implementing two-party computation efficiently with security against malicious adversaries". In: *Security and Cryptography for Networks (SCN)*. Vol. 5229. LNCS. Springer, 2008, pp. 2–20.

[53] Yehuda Lindell and Ben Riva. "Cut-and-Choose Yao-Based Secure Computation in the Online/Offline and Batch Settings". In: *Advances in Cryptology – CRYPTO*. Vol. 8617. LNCS. Springer, 2014, pp. 476–494.

[54] Dahlia Malkhi, Noam Nisan, Benny Pinkas, and Yaron Sella. "Fairplay — a secure two-party computation system". In: *USENIX Security*. USENIX, 2004, pp. 287–302.

[55] Mitsuru Matsui. "Linear Cryptoanalysis Method for DES Cipher". In: *Advances in Cryptology – EUROCRYPT*. Vol. 765. LNCS. Springer, 1993, pp. 386–397.

[56] Willi Meier and Othmar Staffelbach. "The Self-Shrinking Generator". In: *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings*. Ed. by Alfredo De Santis. Vol. 950. Lecture Notes in Computer Science. Springer, 1994, pp. 205–214. ISBN: 3-540-60176-7. DOI: `10.1007/BFb0053436`. URL: `http://dx.doi.org/10.1007/BFb0053436`.

[57] Silvia Mella and Ruggero Susella. "On the Homomorphic Computation of Symmetric Cryptographic Primitives". In: *Cryptography and Coding*. Ed. by Martijn Stam. Vol. 8308. LNCS. Springer Berlin Heidelberg, 2013, pp. 28–44.

[58] Jesper B. Nielsen, Peter S. Nordholt, Claudio Orlandi, and Sai Sheshank Burra. "A New Approach to Practical Active-Secure Two-Party Computation". In: *Advances in Cryptology – CRYPTO*. Vol. 7417. LNCS. Springer, 2012, pp. 681–700.

[59] Benny Pinkas, Thomas Schneider, Nigel P. Smart, and Stephen C. Williams. "Secure Two-Party Computation Is Practical". In: *Advances in Cryptology – ASIACRYPT*. Vol. 5912. LNCS. Springer, 2009, pp. 250–267.

[60] Gilles Piret, Thomas Roche, and Claude Carlet. "PICARO - A Block Cipher Allowing Efficient Higher-Order Side-Channel Resistance". In: *Applied Cryptography and Network Security (ACNS)*. Vol. 7341. LNCS. Springer, 2012, pp. 311–328.

[61] Shahram Rasoolzadeh, Zahra Ahmadian, Mahmood Salmasizadeh, and Mohammad Reza Aref. *Total Break of Zorro using Linear and Differential Attacks*. Cryptology ePrint Archive, Report 2014/220. `http://eprint.iacr.org/2014/220`. 2014.

[62] Thomas Schneider and Michael Zohner. "GMW vs. Yao? Efficient Secure Two-Party Computation with Low Depth Circuits". In: *Financial Cryptography (FC)*. LNCS. Springer, 2013, pp. 275–292.

[63] Abhi Shelat and Chih-Hao Shen. "Fast two-party secure computation with minimal assumptions". In: *Computer and Communications Security (CCS)*. ACM, 2013, pp. 523–534.

[64] Abhi Shelat and Chih-Hao Shen. "Two-Output Secure Computation with Malicious Adversaries". In: *Advances in Cryptology – EUROCRYPT*. Vol. 6632. LNCS. Springer, 2011, pp. 386–405.

[65] Stefan Tillich and Nigel Smart. *Circuits of Basic Functions Suitable For MPC and FHE*. `http://www.cs.bris.ac.uk/Research/CryptographySecurity/MPC/`.

[66] David Wagner. "The Boomerang Attack". In: *Fast Software Encryption (FSE)*. Vol. 1636. LNCS. Springer, 1999, pp. 156–170.

[67] Yanfeng Wang, Wenling Wu, Zhiyuan Guo, and Xiaoli Yu. *Differential Cryptanalysis and Linear Distinguisher of Full-Round Zorro*. Cryptology ePrint Archive, Report 2013/775. `http://eprint.iacr.org/2013/775`. 2013.

[68] Andrew C.-C. Yao. "How to Generate and Exchange Secrets". In: *IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE, 1986, pp. 162–167.

Observations on the SIMON block cipher family

Publication Information

Stefan Kölbl, Gregor Leander, and Tyge Tiessen. "Observations on the SIMON Block Cipher Family". In: *Advances in Cryptology - CRYPTO 2015*. Ed. by Rosario Gennaro and Matthew Robshaw. Vol. 9215. Lecture Notes in Computer Science. Springer, 2015, pp. 161–185. ISBN: 978-3-662-47988-9

Contribution

- All authors contributed equally.
- Main contributions are subsection 2.4, Theorems 3, 4, and 6 in section 3 and 4 as well as Appendix A and B.

Remarks

This publication has been slightly edited to fit the format.

Observations on the SIMON block cipher family

Stefan Kölbl[1], Gregor Leander[2], and Tyge Tiessen[1]
stek@dtu.dk, gregor.leander@rub.de, tyti@dtu.dk

[1] DTU Compute, Technical University of Denmark, Denmark
[2] Horst Görtz Institute for IT Security, Ruhr-Universität Bochum, Germany

Abstract. In this paper we analyse the general class of functions underlying the Simon block cipher. In particular, we derive efficiently computable and easily implementable expressions for the exact differential and linear behaviour of Simon-like round functions.

Following up on this, we use those expressions for a computer aided approach based on SAT/SMT solvers to find both optimal differential and linear characteristics for Simon. Furthermore, we are able to find all characteristics contributing to the probability of a differential for Simon32 and give better estimates for the probability for other variants.

Finally, we investigate a large set of Simon variants using different rotation constants with respect to their resistance against differential and linear cryptanalysis. Interestingly, the default parameters seem to be not always optimal.

Keywords: SIMON, differential cryptanalysis, linear cryptanalysis, block cipher, Boolean functions

1 Introduction

Lightweight cryptography studies the deployment of cryptographic primitives in resource-constrained environments. This research direction is driven by a demand for cost-effective, small-scale communicating devices such as RFID tags that are a cornerstone in the Internet of Things. Most often the constrained resource is taken to be the chip-area but other performance metrics such as latency [7], code-size [2] and ease of side-channel protection [12]) have been considered as well. Some of these criteria were already treated in Noekeon [9].

The increased importance of lightweight cryptography and its applications has lately been reflected in the NSA publishing two dedicated lightweight cipher families: Simon and Speck [5]. Considering that this is only the third time within four decades that the NSA has published a block cipher, this is quite remarkable. Especially as NIST has started shortly after this publication to investigate the possibilities to standardise lightweight primitives, Simon and Speck certainly deserve a thorough investigation. This is emphasised by the fact that, in contrast to common practice,

neither a security analysis nor a justification of the design choices were published by the NSA. This lack of openness necessarily gives rise to curiosity and caution.

In this paper we focus on the SIMON family of block ciphers; an elegant, innovative and very efficient set of block ciphers. There exists already a large variety of papers, mainly focussed on evaluating SIMON's security with regard to linear and differential cryptanalysis. Most of the methods used therein are rather ad-hoc, often only using approximative values for the differential round probability and in particular for the linear square correlation of one round.

Our Contribution

With this study, we complement the existing work threefold. Firstly we develop an exact closed form expression for the differential probability and a $\log(n)$ algorithm for determining the square correlation over one round. Their accuracy is proven rigorously. Secondly we use these expressions to implement a model of differential and linear characteristics for SAT/SMT solvers which allows us to find the provably best characteristics for different instantiations of SIMON. Furthermore we are able to shed light on how differentials in SIMON profit from the collapse of many differential characteristics. Thirdly by generalising the probability expressions and the SAT/SMT model, we are able to compare the quality of different parameter sets with respect to differential and linear cryptanalysis.

As a basis for our goal to understand both the security of SIMON as well as the choice of its parameter set, we rigorously derive formulas for the differential probabilities and the linear square correlations of the SIMON-like round function that can be evaluated in constant time and time linear in the word size respectively. More precisely, we study differential probabilities and linear correlations of functions of the form

$$S^a(x) \odot S^b(x) + S^c(x)$$

where $S^i(x)$ corresponds to a cyclic left shift of x and $\odot$ denotes the bitwise AND operation.

We achieve this goal by first simplifying this question by considering equivalent descriptions both of the round function as well as the whole cipher (cf. Section 2.4). These simplifications, together with the theory of quadratic boolean functions, result in a clearer analysis of linear and differential properties (cf. Sections 3 and 4). Importantly, the derived simple equations for computing the probabilities of the SIMON round function can be evaluated efficiently and, more importantly maybe, are conceptually very easy. This allows them to be easily used in computer-aided investigations of differential and linear properties over more rounds. It should be noted here that the expression for linear approximations is more complex than the expression for the differential case. However, with respect to the running time of the computer-aided investigations this difference is negligible.

We used this to implement a framework based on SAT/SMT solvers to find the provably best differential and linear characteristics for various instantiations of SIMON

(cf. Section 5, in particular Table 1). Furthermore we are able to shed light on how differentials in Simon profit from the collapse of many differential characteristics by giving exact distributions of the probabilities of these characteristics for chosen differentials. The framework is open source and publicly available to encourage further research [13].

In Section 6 we apply the developed theory and tools to investigate the design space of Simon-like functions. In particular, using the computer-aided approach, we find that the standard Simon parameters are not optimal with regard to the best differential and linear characteristics.

As a side result, we improve the probabilities for the best known differentials for several variants and rounds of Simon. While this might well lead to (slightly) improved attacks, those improved attacks are out of the scope of our work.

Interestingly, at least for Simon32 our findings indicate that the choices made by the NSA are good but not optimal under our metrics, leaving room for further investigations and questions. To encourage further research, we propose several alternative parameter choices for Simon32. Here, we are using the parameters that are optimal when restricting the criteria to linear, differential and dependency properties. We encourage further research on those alternative choices to shed more light on the undisclosed design criteria.

We also like to point out that the Simon key-scheduling was not part of our investigations. Its influence on the security of Simon is left as an important open question for further investigations. In line with this, whenever we investigate multi-round properties of Simon in our work, we implicitly assume independent round keys in the computation of probabilities.

Finally, we note that most of our results can be applied to more general constructions, where the involved operations are restricted to AND, XOR, and rotations.

Related Work

There are various papers published on the cryptanalysis of Simon [1], [3], [6], [16], [18], [19]. The most promising attacks so far are based on differential and linear cryptanalysis, however a clear methodology of how to derive the differential probabilities and square correlations seems to miss in most cases. Biryukov, Roy and Velichkov [6] derive a correct, but rather involved method to find the differential probabilities. Abed, List, Lucks and Wenzel [1] state an algorithm for the calculation of the differential probabilities but without further explanation. For the calculation of the square correlations an algorithm seems to be missing all together.

Previous work also identifies various properties like the strong differential effect and give estimate of the probability of differentials.

The concept behind our framework was previously also applied on the ARX cipher Salsa20 [14] and the CAESAR candidate NORX [4]. In addition to the applications proposed in previous work we extend it for linear cryptanalysis, examine the influence of rotation constants and use it to compute the distribution of characteristics corresponding to a differential.

2 Preliminaries

In this section, we start by defining our notation and giving a short description of the round function. We recall suitable notions of equivalence of Boolean functions that allow us to simplify our investigations of SIMON-like round functions. Most of this section is generally applicable to AND-RX constructions, i.e. constructions that only make use of the bitwise operations AND, XOR, and rotations.

2.1 Notation

We denote by $\mathbb{F}_2$ the field with two elements and by $\mathbb{F}_2^n$ the n-dimensional vector space over $\mathbb{F}_2$. By $\mathbf{0}$ and $\mathbf{1}$ we denote the vectors of $\mathbb{F}_2^n$ with all 0s and all 1s respectively. The Hamming weight of a vector $a \in \mathbb{F}_2^n$ is denoted as $\mathrm{wt}(a)$. By $\mathbb{Z}_n$ we denote the integers modulo n.

The addition in $\mathbb{F}_2^n$, i.e. bit-wise XOR, is denoted by $+$. By $\odot$ we denote the AND operation in $\mathbb{F}_2^n$, i.e. multiplication over $\mathbb{F}_2$ in each coordinate:

$$x \odot y = (x_i y_i)_i.$$

By $\vee$ we denote the bitwise OR operation. By $\overline{x}$ we denote the bitwise negation of x, i.e. $\overline{x} := (x + \mathbf{1})$. We denote by $S^i : \mathbb{F}_2^n \to \mathbb{F}_2^n$ the left circular shift by i positions. We also note that any arithmetic of bit indices is always done modulo the word size n.

In this paper we are mainly concerned with functions of the form

$$f_{a,b,c}(x) = S^a(x) \odot S^b(x) + S^c(x) \tag{1}$$

and we identify such functions with its triple (a, b, c) of parameters.

For a vectorial Boolean function on n bits, $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$, we denote by

$$\widehat{f}(\alpha, \beta) = \sum_x \mu\left(\langle \beta, f \rangle + \langle \alpha, x \rangle\right)$$

the Walsh (or Fourier) coefficient with input mask α and output mask β. Here we use $\mu(x) = (-1)^x$ to simplify the notation.

The corresponding squared correlation of f is given by

$$C^2(\alpha \to \beta) = \left(\frac{\widehat{f}(\alpha, \beta)}{2^n}\right)^2.$$

For differentials we similarly denote by $\Pr(\alpha \longrightarrow \beta)$ the probability that a given input difference α results in a given output difference β, i.e.

$$\Pr(\alpha \longrightarrow \beta) = \frac{|\{x \mid f(x) + f(x + \alpha) = \beta\}|}{2^n}.$$

Furthermore, $\mathsf{Dom}(f)$ is the domain of a function f, $\mathsf{Img}(f)$ is its image.

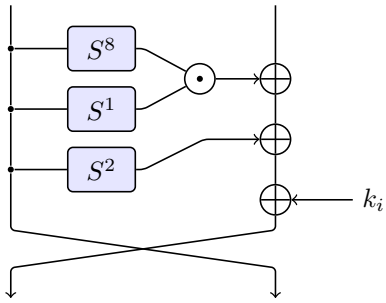

Figure 1: The round function of SIMON.

2.2 Description of SIMON

SIMON is a family of lightweight block ciphers with block sizes 32, 48, 64, 96, and 128 bits. The constructions are Feistel ciphers using a word size n of 16, 24, 32, 48 or 64 bits, respectively. We will denote the variants as SIMON$2n$. The key size varies between of 2, 3, and 4 n-bit words. The round function of SIMON is composed of AND, rotation, and XOR operations on the complete word (see figure 1). More precisely, the round function in SIMON corresponds to

$$S^8(x) \odot S^1(x) + S^2(x),$$

that is to the parameters $(8, 1, 2)$ for f as given in Equation (1). As we are not only interested in the original SIMON parameters, but in investigating the entire design space of SIMON-like functions, we denote by

$$\text{SIMON}[a, b, c]$$

the variant of SIMON where the original round function is replaced by $f_{a,b,c}$ (cf. Equation (1)).

As it is out of scope for our purpose, we refer to [5] for the description of the key-scheduling.

2.3 Affine equivalence of Boolean Functions

Given two (vectorial) Boolean functions f_1 and f_2 on $\mathbb{F}_2^n$ related by

$$f_1(x) = (A \circ f_2 \circ B)(x) + C(x)$$

where A and B are affine permutations and C is an arbitrary affine mapping on $\mathbb{F}_2^n$ we say that f_1 and f_2 are *extended affine equivalent* (cf. [8] for a comprehensive survey).

With respect to differential cryptanalysis, if f_1 and f_2 are extended affine equivalent then the differential $\alpha \xrightarrow{f_1} \beta$ over f_1 has probability p if and only if the differential

$$B(\alpha) \xrightarrow{f_2} A^{-1}(\beta + C(\alpha))$$

over f_2 has probability p as well.

For linear cryptanalysis, a similar relation holds for the linear correlation. If f_1 and f_2 are related as defined above, it holds that

$$\widehat{f_1}(\alpha, \beta) = \widehat{f_2}\left(\left(C \circ B^{-1}\right)^T \beta + \left(B^{-1}\right)^T \alpha, A^T \beta\right).$$

Thus up to linear changes we can study f_2 instead of f_1 directly. Note that, for an actual attack, these changes are usually critical and can certainly not be ignored. However, tracing the changes is, again, simple linear algebra.

For differential and linear properties of Simon-like functions of the form

$$f_{a,b,c}(x) = S^a(x) \odot S^b(x) + S^c(x)$$

this implies that it is sufficient to find the differential and linear properties of the simplified variant

$$f(x) = x \odot S^d(x)$$

and then transfer the results back by simply using linear algebra.[3]

2.4 Structural Equivalence Classes in AND-RX Constructions

AND-RX constructions, i.e. constructions that make only use of the operations AND ($\odot$), XOR ($+$), and rotations (S^r), exhibit a high degree of symmetry. Not only are they invariant under rotation of all input words, output words and constants, they are furthermore structurally invariant under any affine transformation of the bit-indices. As a consequence of this, several equivalent representations of the Simon variants exist.

Let T be a permutation of the bits of an n-bit word that corresponds to an affine transformation of the bit-indices. Thus there are $s \in \mathbb{Z}_n^*$ and $t \in \mathbb{Z}_n$ such that bit i is renamed to $s \cdot i + t$. As the AND and XOR operations are bitwise, T clearly commutes with these:

$$Tv \odot Tw = T(v \odot w)$$
$$Tv + Tw = T(v + w)$$

where v and w are n-bit words. A rotation to the left by r can be written bitwise as $S^r(v)_i = v_{i-r}$. For a rotation, we thus get the following bitwise relation after transformation with T

$$S^r(v)_{s \cdot i + t} = v_{s \cdot (i-r) + t} = v_{s \cdot i + t - s \cdot r} \quad .$$

[3] Note that we can transform the equation $f(x) = S^a(x) \odot S_b(x) + S^c(x)$ to the equation $S^{-a}(f(x)) + S^{c-a}(x)) = x \odot S^{b-a}(x)$.

Substituting $s \cdot i + t$ with j this is the same as

$$S^r(v)_j = v_{j-s\cdot r} \ .$$

Thus the rotation by r has been changed to a rotation by $s \cdot r$. Thus we can write

$$TS^r v = S^{s\cdot r} T v.$$

Commuting the linear transformation of the bit-indices with a rotation thus only changes the rotation constant by a factor. In the special case where all input words, output words and constants are rotated, which corresponds to the case $s = 1$, the rotation constant are left untouched.

To summarise the above, when applying such a transformation T to all input words, output words and constants in an AND-RX construction, the structure of the constructions remains untouched apart from a multiplication of the rotation constants by the factor s.

This means for example for Simon32 that changing the rotation constants from $(8, 1, 2)$ to $(3 \cdot 8, 3 \cdot 1, 3 \cdot 2) = (8, 3, 6)$ and adapting the key schedule accordingly gives us the same cipher apart from a bit permutation. As s has to be coprime to n, all s with $\gcd(s, n) = 1$ are allowed, giving $\varphi(n)$ equivalent tuples of rotation constants in each equivalence class where φ is Euler's phi function.

Together with the result from section 2.3, this implies the following lemma.

Lemma 1. *Any function $f_{a,b,c}$ as defined in Equation (1) is extended affine equivalent to a function*

$$f(x) = x \odot S^d(x)$$

where $d|n$ or $d = 0$.

When looking at differential and square correlations of Simon-like round functions this means that it is sufficient to investigate this restricted set of functions. The results for these functions can then simply be transferred to the general case.

3 Differential Probabilities of SIMON-like round functions

In this section, we derive a closed expression for the differential probability for all Simon-like round functions, i.e. all functions as described in Equation (1). The main ingredients here are the derived equivalences and the observation that any such function is quadratic. Being quadratic immediately implies that its derivative is linear and thus the computation of differential probabilities basically boils down to linear algebra (cf. Theorem 2). However, to be able to efficiently study multiple-round properties and in particular differential characteristics, it is important to have a simple expression for the differential probabilities. Those expressions are given for $f(x) = x \odot S^1(x)$ in Theorem 3 and for the general case in Theorem 4.

3.1 A closed expression for the differential probability

The following statement summarises the differential properties of the f function.

Theorem 2. *Given an input difference α and an output difference β the probability p of the corresponding differential (characteristic) for the function $f(x) = x \odot S^a(x)$ is given by*

$$p_{\alpha,\beta} = \begin{cases} 2^{-(n-d)} & \text{if } \beta + \alpha \odot S^a(\alpha) \in \mathsf{Img}(L_\alpha) \\ 0 & \text{else} \end{cases}$$

where

$$d = \dim \ker(L_\alpha)$$

and

$$L_\alpha(x) = x \odot S^a(\alpha) + \alpha \odot S^a(x)$$

Proof. We have to count the number of solutions to the equation

$$f(x) + f(x + \alpha) = \beta.$$

This simplifies to

$$L_\alpha(x) = x \odot S^a(\alpha) + \alpha \odot S^a(x) = \beta + \alpha \odot S^a(\alpha)$$

As this is an affine equation, it either has zero solutions or the number of solutions equals the kernel size, i.e. the number of elements in the subspace

$$\{x \mid x \odot S^a(\alpha) + \alpha \odot S^a(x) = \mathbf{0}\}.$$

Clearly, the equation has solutions if and only if $\beta + \alpha \odot S^a(\alpha)$ is in the image of L_α. □

Next we present a closed formula to calculate the differential probability in the case where $a = 1$. Furthermore we restrict ourselves to the case where n is even.

Theorem 3. *Let*

$$\texttt{varibits} = S^1(\alpha) \vee \alpha$$

and

$$\texttt{doublebits} = \alpha \odot \overline{S^1(\alpha)} \odot S^2(\alpha).$$

Then the probability that difference α goes to difference β is

$$P(\alpha \to \beta) = \begin{cases} 2^{-n+1} & \text{if } \alpha = \mathbf{1} \text{ and } \mathrm{wt}(\beta) \equiv 0 \mod 2 \\ 2^{-\mathrm{wt}(\texttt{varibits}+\texttt{doublebits})} & \text{if } \alpha \neq \mathbf{1} \text{ and } \beta \odot \overline{\texttt{varibits}} = \mathbf{0} \\ & \text{and } (\beta + S^1(\beta)) \odot \texttt{doublebits} = \mathbf{0} \\ 0 & \text{else} \end{cases}$$

Proof. According to theorem 2, we need to prove two things. Firstly we need to prove that the rank of L_α (i.e. $n - \dim\ker L_\alpha$) is $n-1$ when $\alpha = \mathbf{1}$, and $\mathrm{wt}(\texttt{varibits} + \texttt{doublebits}))$ otherwise. Secondly we need to prove that $\beta + \alpha \odot S^1(\alpha) \in \mathsf{Img}(L_\alpha)$ iff $\mathrm{wt}(\beta) \equiv 0 \mod 2$ when $\alpha = \mathbf{1}$, and that $\beta + \alpha \odot S^1(\alpha) \in \mathsf{Img}(L_\alpha)$ iff $\beta \odot \texttt{varibits} = \mathbf{0}$ and $(\beta + S^1(\beta)) \odot \texttt{doublebits} = \mathbf{0}$ when $\alpha \neq \mathbf{1}$.

We first consider the first part. Let us write $L_\alpha(x)$ in matrix form and let us take x to be a column vector. $S^1(\alpha) \odot x$ can be written as $M_{S^1(\alpha)\odot} x$ with

$$M_{S^1(\alpha)\odot} = \begin{pmatrix} \alpha_{n-1} & \dots & \dots & 0 \\ \vdots & \alpha_0 & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \dots & \dots & \alpha_{n-2} \end{pmatrix} . \tag{1}$$

Equivalently we can write $\alpha \odot x$ and $S^1(x)$ with matrices as $M_{\alpha\odot} x$ and $M_{S^1} x$ respectively where

$$M_{\alpha\odot} = \begin{pmatrix} \alpha_0 & \dots & \dots & 0 \\ \vdots & \alpha_1 & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \dots & \dots & \alpha_{n-1} \end{pmatrix} \text{ and } M_{S^1} = \begin{pmatrix} 0_{1,n-1} & I_{1,1} \\ I_{n-1,n-1} & 0_{n-1,1} \end{pmatrix} , \tag{2}$$

i.e. M_{S^1} consists of two identity and two zero submatrices. The result of $M_{S^1(\alpha)\odot} + M_{\alpha\odot} M_{S^1}$ can now be written as

$$\begin{pmatrix} \alpha_{n-1} & 0 & 0 & \dots & \alpha_0 \\ \alpha_1 & \alpha_0 & 0 & \dots & 0 \\ 0 & \alpha_2 & \alpha_1 & & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \alpha_{n-1} & \alpha_{n-2} \end{pmatrix} \tag{3}$$

Clearly the rank of the matrix is $n-1$ when all α_i are 1. Suppose now that not all α_i are 1. In that case, a set of non-zero rows is linearly dependent iff there exist two identical rows in the set. Thus to calculate the rank of the matrix, we need to calculate the number of unique non-zero rows.

By associating the rows in the above matrix with the bits in `varibits`, we can clearly see that the number of non-zero rows in the matrices corresponds to the number of 1s in $\texttt{varibits} = S^1(\alpha) \vee \alpha$.

To count the number of non-unique rows, first notice that a nonzero row can only be identical to the row exactly above or below. Suppose now that a non-zero row i is identical to the row $(i-1)$ above. Then α_{i-1} has to be 0 while α_i and α_{i-2} have to be 1. But then row i cannot simultaneously be identical to row $(i+1)$ below. Thus it is sufficient to calculate the number of non-zero rows minus the number of rows

that are identical to the row above it to find the rank of the matrix. Noting that row i is non-zero iff $\alpha_i\alpha_{i-1}$ and that $\alpha_i\overline{\alpha_{i-1}}\alpha_{i-2}$ is only equal 1 when row i is non-zero and equal to the row above it. Thus calculating the number of i for which

$$\alpha_i\alpha_{i-1} + \alpha_i\overline{\alpha_{i-1}}\alpha_{i-2}$$

is equal 1 gives us the rank of L_α. This corresponds to calculating $\mathrm{wt}(\texttt{varibits} + \texttt{doublebits})$.

For the second part of the proof, we need to prove the conditions that check whether $\beta + \alpha \odot S^1(\alpha) \in \mathsf{Img}(L_\alpha)$. First notice that $\alpha \odot S^1(\alpha)$ is in the image of L_α (consider for x the vector with bits alternately set to 0 and 1). Thus it is sufficient to test whether β is in $\mathsf{Img}\, L_\alpha$. Let $y = L_\alpha(x)$. Let us first look at the the case of $\alpha = \mathbf{1}$. Then $L_\alpha(x) = x + S^1(x)$. We can thus deduce from bit y_i whether $x_i = x_{i-1}$ or $x_i \neq x_{i-1}$. Thus the bits in y create a chain of equalities/inequalities in the bits of x which can only be fulfilled if there the number of inequalities is even. Hence in that case $\beta \in \mathsf{Img}\, L_\alpha$ iff $\mathrm{wt}(\beta) \equiv 0 \mod 2$.

For the case that $\alpha \neq \mathbf{1}$, we first note that y_i has to be zero if the corresponding row i in the matrix of equation (3) is all zeroes. Furthermore following our discussion of this matrix earlier, we see that y_i is independent of the rest of y if the corresponding row is linearly independent of the other rows and that y_i has to be the same as y_{i-1} if the corresponding rows are identical. Thus we only need to check that the zero-rows of the matrix correspond to zero bits in β and that the bits in β which correspond to identical rows in the matrix are equal. Thus β is in the image of L_α iff $\beta \odot \overline{\texttt{varibits}} = \mathbf{0}$ and $(\beta + S^1(\beta)) \odot \texttt{doublebits} = \mathbf{0}$. □

3.2 The full formula for differentials.

Above we treated only the case for the simplified function $f(x) = x \cdot S^1(x)$. As mentioned earlier, the general case where $\gcd(a-b, n) = 1$ can be deduced from this with linear algebra. When $\gcd(d, n) \neq 1$ though, the function $f(x) = x \odot S^d(x)$ partitions the output bits into independent classes. This not only raises differential probabilities (worst case $d = 0$), it also makes the notation for the formulas more complex and cumbersome, though not difficult. We thus restrict ourselves to the most important case when $\gcd(a-b, n) = 1$. The general formulas are then

Theorem 4. *Let $f(x) = S^a(x) \odot S^b(x) + S^c(x)$, where $\gcd(n, a-b) = 1$, n even, and $a > b$ and let α and β be an input and an output difference. Then with*

$$\texttt{varibits} = S^a(\alpha) \vee S^b(\alpha)$$

and

$$\texttt{doublebits} = S^b(\alpha) \odot \overline{S^a(\alpha)} \odot S^{2a-b}(\alpha)$$

and

$$\gamma = \beta + S^c(\alpha)$$

we have that the probability that difference α goes to difference β is

$$P(\alpha \to \beta) = \begin{cases} 2^{-n+1} & \textit{if } \alpha = \mathbf{1} \textit{ and } \mathrm{wt}(\gamma) \equiv 0 \mod 2 \\ 2^{-\,\mathrm{wt}(\texttt{varibits}+\texttt{doublebits})} & \textit{if } \alpha \neq \mathbf{1} \textit{ and } \gamma \odot \overline{\texttt{varibits}} = \mathbf{0} \\ & \textit{and } (\gamma + S^{a-b}(\gamma)) \odot \texttt{doublebits} = \mathbf{0} \\ 0 & \textit{else} \ . \end{cases}$$

For a more intuitive approach and some elaboration on the differential probabilities, we refer to the ePrint version of this paper.

4 Linear Correlations of SIMON-like round functions

As in the differential case, for the study of linear approximations, we also build up on the results from subsections 2.3 and 2.4. We will thus start with studying linear approximations for the function $f(x) = x \odot S^a(x)$. Again, the key point here is that all those functions are quadratic and thus their Fourier coefficient, or equivalently their correlation, can be computed by linear algebra (cf. Theorem 5). Theorem 6 is then, in analogy to the differential case, the explicit expression for the linear correlations. It basically corresponds to an explicit formula for the dimension of the involved subspace.

The first result is the following:

Theorem 5.

$$\widehat{f}(\alpha,\beta)^2 = \begin{cases} 2^{n+d} & \textit{if } \alpha \in U_\beta^\perp \\ 0 & \textit{else} \end{cases}$$

where

$$d = \dim U_\beta$$

and

$$U_\beta = \{y \mid \beta \odot S^a(y) + S^{-a}(\beta \odot y) = \mathbf{0}\}$$

Proof. We compute

$$\begin{aligned} \widehat{f}(\alpha,\beta)^2 &= \sum_{x,y} \mu\left(\langle \beta, f(x) + f(y)\rangle + \langle \alpha, x+y\rangle\right) \\ &= \sum_{x,y} \mu\left(\langle \beta, f(x) + f(x+y)\rangle + \langle \alpha, y\rangle\right) \\ &= \sum_{x,y} \mu\left(\langle \beta, x \odot S^a(x) + (x+y) \odot S^a(x+y)\rangle + \langle \alpha, y\rangle\right) \\ &= \sum_{y} \mu\left(\langle \beta, f(y)\rangle + \langle \alpha, y\rangle\right) \sum_{x} \mu\left(\langle \beta, x \odot S^a(y) + y \odot S^a(x)\rangle\right) \\ &= \sum_{y} \mu\left(\langle \beta, f(y)\rangle + \langle \alpha, y\rangle\right) \sum_{x} \mu\left(\langle x, \beta \odot S^a(y) + S^{-a}(\beta \odot y)\rangle\right) \ . \end{aligned}$$

Now for the sum over x only two outcomes are possible, 2^n or zero. More precisely, it holds that

$$\sum_x \mu\left(\langle x, \beta \odot S^a(y) + S^{-a}(\beta \odot y)\rangle\right) = \begin{cases} 2^n & \text{if } \beta \odot S^a(y) + S^{-a}(\beta \odot y) = \mathbf{0} \\ 0 & \text{else} \quad . \end{cases}$$

Thus, defining

$$U_\beta = \{y \mid \beta \odot S^a(y) + S^{-a}(\beta \odot y) = \mathbf{0}\}$$

we get

$$\widehat{f}(\alpha, \beta)^2 = 2^n \sum_{y \in U_\beta} \mu\left(\langle \beta, f(y)\rangle + \langle \alpha, y\rangle\right).$$

Now as

$$\begin{aligned} \langle \beta, f(y)\rangle =& \langle \beta, y \odot S^a(y)\rangle && (1) \\ =& \langle \mathbf{1}, y \odot \beta \odot S^a(y)\rangle && (2) \\ =& \langle \mathbf{1}, y \odot S^{-a}(\beta \odot y)\rangle && (3) \\ & && (4) \end{aligned}$$

Now, the function $f_\beta := \langle \beta, f(y)\rangle$ is linear over U_β as can be easily seen by the definition of U_β. Moreover, as f_β is unbalanced for all β, it follows that actually f_β is constant zero on U_β. We thus conclude that

$$\widehat{f}(\alpha, \beta)^2 = 2^n \sum_{y \in U_\beta} \mu\left(\langle \alpha, y\rangle\right).$$

With a similar argument as above, it follows that $\widehat{f}(\alpha, \beta)^2$ is non-zero if and only if α is contained in $U_\beta^\perp$. □

Let us now restrict ourselves to the case where $f(x) = x \odot S^1(x)$. The general case can be deduced analogously to the differential probabilities. For simplicity we also restrict ourselves to the case where n is even.

First we need to introduce some notation. Let $x \in \mathbb{F}_2^n$ with not all bits equal to 1. We now look at blocks of consecutive 1s in x, including potentially a block that "wraps around" the ends of x. Let the lengths of these blocks, measured in bits, be denoted as $c_0, \ldots, c_m$. For example, the bitstring 100101111011 has blocks of length 1, 3, and 4. With this notation define $\theta(x) := \sum_{i=0}^{m} \lceil \frac{c_i}{2} \rceil$.

Noting that the linear square correlation of f is $\frac{\widehat{f}(\alpha,\beta)^2}{2^{2n}}$, we then have the following theorem:

Theorem 6. *With the notation from above it holds that the linear square correlation of $\alpha \xrightarrow{f} \beta$ can be calculated as*

$$C(\alpha \to \beta) = \begin{cases} 2^{-n+2} & \textit{if } \beta = \mathbf{1} \textit{ and } \alpha \in U_\beta^\perp \\ 2^{-\theta(\beta))} & \textit{if } \beta \neq \mathbf{1} \textit{ and } \alpha \in U_\beta^\perp \\ 0 & \textit{else.} \end{cases}$$

Proof. Define $L_\beta(x) := \beta \odot S^1(x) + S^{-1}(\beta \odot x)$. Clearly L_β is linear. Also $U_\beta = \ker L_\beta(x)$. Let us determine the rank of this mapping. Define the matrices $M_{\beta\cdot}$, M_{S^1}, and $M_{S^{-1}}$ as

$$M_{\beta\cdot} = \begin{pmatrix} \beta_0 & \dots & \dots & 0 \\ \vdots & \beta_1 & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \dots & \dots & \beta_{n-1} \end{pmatrix} \qquad \begin{aligned} M_{S^1} &= \begin{pmatrix} 0_{1,n-1} & I_{1,1} \\ I_{n-1,n-1} & 0_{n-1,1} \end{pmatrix} \\ M_{S^{-1}} &= \begin{pmatrix} 0_{n-1,1} & I_{n-1,n-1} \\ I_{1,1} & 0_{1,n-1} \end{pmatrix} \end{aligned} \tag{5}$$

We can then write L_β in matrix form as

$$\begin{pmatrix} 0 & \beta_1 & 0 & \dots & 0 & \beta_0 \\ \beta_1 & 0 & \beta_2 & 0 & \dots & 0 \\ 0 & \beta_2 & 0 & \beta_3 & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & \ddots & 0 & \beta_{n-1} \\ \beta_0 & 0 & \dots & 0 & \beta_{n-1} & 0 \end{pmatrix} \tag{6}$$

Clearly, if β is all 1s, the rank of the matrix is $n-2$ as n is even.[4] Let us therefore now assume that β is not all 1s. When we look at a block of 1s in β e.g., $\beta_{i-1} = 0$, $\beta_i, \beta_{i+1}, \dots, \beta_{i+l-1} = 1$, and $\beta_l = 0$. Then clearly the l rows $i, i+1, \dots, i+l-1$ are linearly independent when l is even. When l is odd though, the sum of rows i, $i+2$, $i+4$, up to row $i+l-3$ will equal row $i+l-1$. In that case there are thus only $l-1$ linearly independent rows. As the blocks of 1s in β generate independent blocks of rows, we can summarise that the rank of the matrix is exactly $\theta(\beta)$. □

Analogously to the differential probabilities, the linear probabilities in the general case can be derived from this. It is likewise straightforward to derive how to determine whether $\alpha \in U_\beta^\perp$. As an explicit formulation of this is rather tedious, we instead refer to the implementation in Python given in the Appendix B where both is achieved in the case where $\gcd(a-b, n) = 1$ and n is even.

For a more intuitive approach and some elaboration on the linear probabilities, we refer to the ePrint version of this paper.

5 Finding Optimal Differential and Linear Characteristics

While there are various methods for finding good characteristics, determining optimal differential or linear characteristics remains a hard problem in general. The formulas derived for both differential and linear probabilities enable us to apply an algebraic

[4] The rank is $n-1$ when n is odd.

approach to finding the best characteristics. A similar technique has been applied to the ARX cipher Salsa20 [14] and the CAESAR candidate NORX [4]. For finding the optimal characteristics for SIMON we implemented an open source tool [13] based on the SAT/SMT solvers CryptoMiniSat [15] and STP [11].

In the next section we will show how SIMON can be modeled to find both the best differential and linear characteristics in this framework and how this can be used to solve cryptanalytic problems.

5.1 Model for Differential Cryptanalysis of SIMON

First we define the variables used in the model of SIMON. We use two n-bit variables x_i, y_i to represent the XOR-difference in the left and right halves of the state for each round and an additional variable z_i to store the XOR-difference of the output of the AND operation.

For representing the $\log_2$ probability of the characteristic we introduce an additional variable w_i for each round. The sum over all probabilities w_i then gives the probability of the corresponding differential characteristic. The values w_i are computed according to theorem 4 as

$$w_i = \mathrm{wt}(\texttt{varibits} + \texttt{doublebits}) \tag{1}$$

where $\mathrm{wt}(x)$ is the Hamming weight of x and

$$\begin{aligned} \texttt{varibits} &= S^a(x_i) \vee S^b(x_i) \\ \texttt{doublebits} &= S^b(x_i) \odot \overline{S^a(x_i)} \wedge S^{2a-b}(x_i) \end{aligned}$$

Therefore, for one round of SIMON we get the following set of constraints:

$$\begin{aligned} y_{i+1} &= x_i \\ 0 &= (z_i \odot \texttt{varibits}) \\ 0 &= (z_i + S^{a-b}(z_i)) \odot \texttt{doublebits} \\ x_{i+1} &= y_i + z_i + S^c(x_i) \\ w_i &= \mathrm{wt}(\texttt{varibits} + \texttt{doublebits}) \end{aligned} \tag{2}$$

A model for linear characteristics, though slightly more complex, can be implemented in a similar way. A description of this model can be found in the implementation of our framework. Despite the increase in complexity, we could not observe any significant impact on the solving time for the linear model.

5.2 Finding Optimal Characteristics

We can now use the previous model for SIMON to search for optimal differential characteristics. This is done by formulating the problem of finding a valid characteristic, with respect to our constraints, for a given probability w. This is important to limit

Table 1: Overview of the optimal differential (on top) and linear characteristics for different variants of SIMON. The probabilities are given as $\log_2(p)$, for linear characteristic the squared correlation is used.

Rounds:	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Differential															
SIMON32	-2	-4	-6	-8	-12	-14	-18	-20	-25	-30	-34	-36	-38	-40	-42
SIMON48	-2	-4	-6	-8	-12	-14	-18	-20	-26	-30	-35	-38	-44	-46	-50
SIMON64	-2	-4	-6	-8	-12	-14	-18	-20	-26	-30	-36	-38	-44	-48	-54
Linear															
SIMON32	-2	-4	-6	-8	-12	-14	-18	-20	-26	-30	-34	-36	-38	-40	-42
SIMON48	-2	-4	-6	-8	-12	-14	-18	-20	-26	-30	-36	-38	-44	-46	-50
SIMON64	-2	-4	-6	-8	-12	-14	-18	-20	-26	-30	-36	-38	-44	-48	-54

the search space and makes sense as we are usually more interested in differential characteristics with a high probability as they are more promising to lead to attacks with a lower complexity. Therefore, we start with a high probability and check if such a characteristic exists. If not we lower the probability.

The procedure can be described in the following way:

- For each round of the cipher add the corresponding constraints as defined in (2). This system of constraints then exactly describes the form of a valid characteristic for the given parameters.
- Add a condition which accumulates the probabilities of each round as defined in (1) and check if it is equal to our target probability w.
- Query if there exists an assignment of variables which is satisfiable under the constraints.
- Decrement the probability w and repeat the procedure.

One of the main advantages compared to other approaches is that we can prove an upper bound on the probability of characteristics for a given cipher and number of rounds. If the solvers determines the set of conditions unsatisfiable, we know that no characteristic with the specified probability exists. We used this approach to determine the characteristics with optimal probability for different variants of SIMON. The results are given in Table 1.

Upper Bound for the Characteristics.

During our experiments we observed that it seems to be an easy problem for the SMT/SAT solver to prove the absence of differential characteristics above $w_{\max}$. This can be used to get a lower bound on the probability of characteristics contributing to the differential. The procedure is similar to finding the optimal characteristics.

- Start with a very low initial probability w_i.
- Add the same system of constraints which were used for finding the characteristic.
- Add a constraint fixing the variables (x_0, y_0) to Δ_{in} and (x_r, y_r) to Δ_{out}.
- Query if there is a solution for this weight.
- Increase the probability w_i and repeat the procedure until a solution is found.

5.3 Computing the Probability of a Differential

Given a differential characteristic it is of interest to determine the probability of the associated differential $\Pr(\Delta_{\text{in}} \xrightarrow{f^r} \Delta_{\text{out}})$ as it might potentially have a much higher probability then the single characteristic. It is often assumed that the probability of the best characteristic can be used to approximate the probability of the best differential. However, this assumption only gives an inaccurate estimate in the case of Simon.

Similarly to the previous approach for finding the characteristic, we can formalise the problem of finding the probability of a given differential in the following way:

- Add the same system of constraints which were used for finding the characteristic.
- Add a constraint fixing the variables (x_0, y_0) to Δ_{in} and (x_r, y_r) to Δ_{out}.
- Use a SAT solver to find **all** solutions s_i for the probability w.
- Decrement the probability w and repeat the procedure.

The probability of the differential is then given by

$$\Pr(\Delta_{\text{in}} \xrightarrow{f^r} \Delta_{\text{out}}) = \sum_{i=w_{\min}}^{w_{\max}} s_i \cdot 2^{-i} \tag{3}$$

where s_i is the number of characteristics with a probability of 2^{-i}.

We used this approach to compute better estimates for the probability of various differentials (see Table 2). In the case of Simon32 we were able to find *all* characteristics contributing to the differentials for 13 and 14 rounds. The distribution of the characteristics and accumulated probability of the differential is given in Figure 1. It is interesting to see that the distribution of w in the range $[55, 89]$ is close to uniform and therefore the probability of the corresponding differential improves only negligible and converges quickly towards the measured probability[5].

[5] We encrypted all 2^{32} possible texts under 100 random keys to obtain the estimate of the probability for 13-round Simon32.

Table 2: Overview of the differentials and the range $[w_{\min}, w_{\max}]$ of the $\log_2$ probabilities of the characteristics contributing to the differential. For computing the lower bound $\log_2(p)$ of the probability of the differentials, we used all characteristics with probabilities in the range from $w_{\min}$ up to the values in brackets in the $w_{\max}$ column.

Cipher	Rounds	Δ_{in}	Δ_{out}	$w_{\min}$	$w_{\max}$	$\log_2(p)$
Simon32	13	$(0, 40)$	$(4000, 0)$	36	91 (91)	-28.79
Simon32	14	$(0, 8)$	$(800, 0)$	38	120 (120)	-30.81
Simon48	15	$(20, 800088)$	$(800208, 2)$	46	219 (79)	-41.02
Simon48	16	$(800000, 220082)$	$(800000, 220000)$	50	256 (68)	-44.33
Simon48	17	$(80, 222)$	$(222, 80)$	52	269 (85)	-46.32
Simon64	21	$(4000000, 11000000)$	$(11000000, 4000000)$	68	453 (89)	-57.57
Simon64	22	$(440, 1880)$	$(440, 100)$	72	502 (106)	-61.32

The performance of the whole process is very competitive compared to dedicated approaches. Enumerating all characteristics up to probability 2^{-46} for the 13-round Simon32 differential takes around 90 seconds on a single CPU core and already gives a better estimate compared to the results in [6]. A complete enumeration of all characteristics for 13-round Simon32 took close to one core month using CryptoMiniSat4 [15]. The computational effort for other variants of Simon is comparable given the same number of rounds. However, for these variants we can use differentials with a lower probability covering more rounds due to the increased block size. In this case the running time increases due to the larger interval $[w_{\min}, w_{\max}]$ and higher number of rounds.

For Simon48 and Simon64 we are able to improve the estimate given in [17]. Additionally we found differentials which can cover 17 rounds for Simon48 and 22 rounds for Simon64 which might have potential to improve previous attacks. Our results are also closer to the experimentally obtained estimates given in [10] but give a slightly lower probability. This can be due to the limited number of characteristics we use for the larger Simon variants or the different assumptions on the independence of rounds.

Our results are limited by the available computing power and in general it seems to be difficult to count all characteristics for weights in $[w_{\min}, w_{\max}]$, especially for the larger variants of Simon. However the whole process is embarrassingly parallel, as one can split up the computation for each probability w_i. Furthermore, the improvement that one gets decreases quickly. For all differentials we observed that the distribution of differential characteristics becomes flat after a certain point.

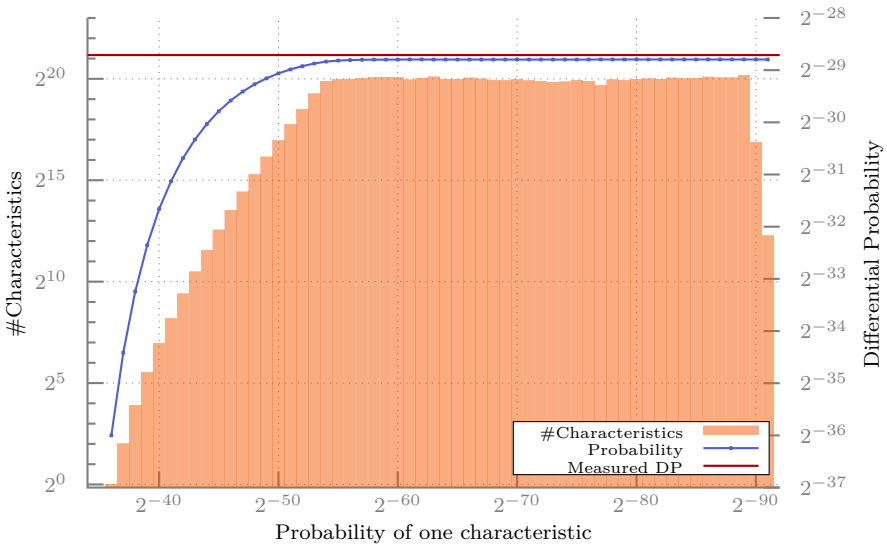

Figure 1: Distribution of the number of characteristics for the differential $(0, 40) \rightarrow (4000, 0)$ for 13-round SIMON32 and the accumulated probability. A total of $\approx 2^{25.21}$ characteristics contribute to the probability.

6 Analysis of the Parameter Choices

The designers of SIMON so far gave no justification for their choice of the rotation constants. Here we evaluate the space of rotation parameters with regard to different metrics for the quality of the parameters. Our results are certainly not a definite answer but are rather intended as a starting point to evaluating the design space and reverse engineering the design choices. We consider all possible sets of rotation constants (a, b, c)[6] and checked them for diffusion properties and the optimal differential and linear characteristics.

6.1 Diffusion

As a very simple measure to estimate the quality of the rotation constants, we measure the number of rounds that are needed to reach full diffusion. Full diffusion is reached when every state bit principally depends on all input bits. Compared to computing linear and differential properties it is an easy task to determine the dependency.

In Table 3 we give a comparison to how well the standard SIMON rotation parameters fare within the distribution of all possible parameter sets. The exact distributions for all SIMON variants can be found in the appendix in Table 8.

[6] Without lack of generality, we assume though that $a \geq b$.

Table 3: The number of rounds after which full diffusion is reached for the standard SIMON parameters in comparison to the whole possible set of parameters.

Block size	32	48	64	96	128
Standard parameters	7	8	9	11	13
Median	8	10	11	13	14
First quartile	7	9	9	11	12
Best possible	6	7	8	9	10
Rank	2nd	2nd	2nd	3rd	4th

6.2 Differential and Linear

As a second criteria for our parameters, we computed for all $a > b$ and $\gcd(a-b, n) = 1$ the optimal differential and linear characteristics for 10 rounds of SIMON32, SIMON48 and SIMON64. A list of the parameters which are optimal for all three variants of SIMON can be found in Appendix D.

It is important here to note that there are also many parameter sets, including the standard choice, for which the best 10-round characteristics of SIMON32 have a probability of 2^{-25} compared to the optimum of 2^{-26}. However, this difference by a factor of 2 does not seem to occur for more than 10 rounds and also not any larger variants of SIMON.

6.3 Interesting Alternative Parameter Sets

As one result of our investigation we chose three exemplary sets of parameters that surpass the standard parameters with regards to some metrics. Those variants are SIMON$[12, 5, 3]$, SIMON$[7, 0, 2]$ and SIMON$[1, 0, 2]$.

SIMON$[12, 5, 3]$ has the best diffusion amongst the parameters which have optimal differential and linear characteristics for 10 rounds. The two other choices are both restricted by setting $b = 0$ as this would allow a more efficient implementation in software. Among those SIMON$[7, 0, 2]$ has the best diffusion and the characteristics behave similar to the standard parameters. Ignoring the diffusion SIMON$[1, 0, 2]$ seems also an interesting choice as it is optimal for the differential and linear characteristics.

If we look though at the differential corresponding to the best differential characteristic of SIMON$[7, 0, 2]$ and SIMON$[1, 0, 2]$, then we can see the number of characteristics contributing to it is significantly higher than for the standard parameters (see Appendix Table 6). However, for SIMON$[12, 5, 3]$ the differential shows a surprisingly different behaviour and the probability of the differential is much closer to the probability of the characteristic. On the other side, the characteristics seem to be worse for the larger variants as can be seen in Table 7. Furthermore it might be desirable to have at least one rotation parameter that corresponds to a byte length, something that the standard parameter set features.

7 Conclusion and Future Work

In this work we analysed the general class of functions underlying the Simon block cipher. First we rigorously derived efficiently computable and easily implementable expressions for the exact differential and linear behaviour of Simon-like round functions.

Building upon this, we used those expressions for a computer aided approach based on SAT/SMT solvers to find both optimal differential and linear characteristics for Simon. Furthermore, we were able to find all characteristics contributing to the probability of a differential for Simon32 and gave better estimates for the probability for other variants.

Finally, we investigated the space of Simon variants using different rotation constants with respect to diffusion, and the optimal differential and linear characteristics. Interestingly, the default parameters seem to be not always optimal.

This work opens up for further investigations. In particular, the choice and justifications of the NSA parameters for Simon remains unclear. Besides our first progress concerning the round function, the design of the key schedule remains largely unclear and further investigation is needed here.

Acknowledgments

First of all, we wish to thank Tomer Ashur. Both the method to check whether a linear input mask gives a correlated or uncorrelated linear 1-round characteristic for a given output mask as well as the first version of the SMT/SAT model for linear characteristics in Simon were an outcome of our discussions. We furthermore wish to thank the reviewers for comments that helped to improve the paper.

References

[1] Farzaneh Abed, Eik List, Stefan Lucks, and Jakob Wenzel. "Differential Cryptanalysis of Round-Reduced SIMON and SPECK". In: *Fast Software Encryption, FSE 2014*. Ed. by Carlos Cid and Christian Rechberger. Vol. 8540. Lecture Notes in Computer Science. Springer, 2015, pp. 525–545. ISBN: 978-3-662-46705-3.

[2] Martin R. Albrecht, Benedikt Driessen, Elif Bilge Kavun, Gregor Leander, Christof Paar, and Tolga Yalçin. "Block Ciphers - Focus on the Linear Layer (feat. PRIDE)". In: *Advances in Cryptology - CRYPTO 2014*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8616. Lecture Notes in Computer Science. Springer, 2014, pp. 57–76. ISBN: 978-3-662-44370-5.

[3] Javad Alizadeh, Hoda AlKhzaimi, Mohammad Reza Aref, Nasour Bagheri, Praveen Gauravaram, Abhishek Kumar, Martin M. Lauridsen, and Somitra Kumar Sanadhya. "Cryptanalysis of SIMON Variants with Connections". In: *Radio Frequency Identification: Security and Privacy Issues, RFIDSec 2014*. Ed. by Nitesh Saxena and Ahmad-Reza Sadeghi. Vol. 8651. Lecture Notes in Computer Science. Springer, 2014, pp. 90–107. ISBN: 978-3-319-13065-1.

[4] Jean-Philippe Aumasson, Philipp Jovanovic, and Samuel Neves. "Analysis of NORX: Investigating Differential and Rotational Properties". In: *Progress in Cryptology - LATINCRYPT 2014*. Ed. by Diego F. Aranha and Alfred Menezes. Vol. 8895. Lecture Notes in Computer Science. Springer, 2015, pp. 306–324. ISBN: 978-3-319-16294-2.

[5] 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. `http://eprint.iacr.org/`. 2013.

[6] Alex Biryukov, Arnab Roy, and Vesselin Velichkov. "Differential Analysis of Block Ciphers SIMON and SPECK". In: *Fast Software Encryption, FSE 2014*. Ed. by Carlos Cid and Christian Rechberger. Vol. 8540. Lecture Notes in Computer Science. Springer, 2015, pp. 546–570. ISBN: 978-3-662-46705-3.

[7] 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: *Advances in Cryptology - ASIACRYPT 2012*. Ed. by Xiaoyun Wang and Kazue Sako. Vol. 7658. Lecture Notes in Computer Science. Springer, 2012, pp. 208–225. ISBN: 978-3-642-34960-7.

[8] Claude Carlet. "Vectorial Boolean Functions for Cryptography". In: *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*. Vol. 134. Encyclopedia of Mathematics And Its Applications. Cambridge Univ. Press, 2010, pp. 398–469.

[9] Joan Daemen, Michaël Peeters, Gilles Van Assche, and Vincent Rijmen. *The NOEKEON Block Cipher*. Submission to the NESSIE project. 2000.

[10] Itai Dinur, Orr Dunkelman, Masha Gutman, and Adi Shamir. *Improved Top-Down Techniques in Differential Cryptanalysis*. Cryptology ePrint Archive, Report 2015/268. `http://eprint.iacr.org/`. 2015.

[11] Vijay Ganesh, Trevor Hansen, Mate Soos, Dan Liew, and Ryan Govostes. *STP constraint solver*. `https://github.com/stp/stp`. 2014.

[12] Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, and Kerem Varici. "LS-Designs: Bitslice Encryption for Efficient Masked Software Implementations". In: *Fast Software Encryption, FSE 2014*. Ed. by Carlos Cid and Christian Rechberger. Vol. 8540. Lecture Notes in Computer Science. Springer, 2015, pp. 18–37. ISBN: 978-3-662-46705-3.

[13] Stefan Kölbl. *CryptoSMT: An easy to use tool for cryptanalysis of symmetric primitives*. `https://github.com/kste/cryptosmt`. 2015.

[14] Nicky Mouha and Bart Preneel. *Towards Finding Optimal Differential Characteristics for ARX: Application to Salsa20*. Cryptology ePrint Archive, Report 2013/328. `http://eprint.iacr.org/`. 2013.

[15] Mate Soos. *CryptoMiniSat SAT solver*. `https://github.com/msoos/cryptominisat/`. 2014.

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

[17] Siwei Sun, Lei Hu, Meiqin Wang, Peng Wang, Kexin Qiao, Xiaoshuang Ma, Danping Shi, Ling Song, and Kai Fu. *Towards Finding the Best Characteristics of Some Bit-oriented Block Ciphers and Automatic Enumeration of (Related-key) Differential and Linear Characteristics with Predefined Properties*. Cryptology ePrint Archive, Report 2014/747. `http://eprint.iacr.org/`. 2014.

[18] Siwei Sun, Lei Hu, Peng Wang, Kexin Qiao, Xiaoshuang Ma, and Ling Song. "Automatic Security Evaluation and (Related-key) Differential Characteristic Search: Application to SIMON, PRESENT, LBlock, DES(L) and Other Bit-Oriented Block Ciphers". In: *Advances in Cryptology - ASIACRYPT 2014*. Ed. by Palash Sarkar and Tetsu Iwata. Vol. 8873. Lecture Notes in Computer Science. Springer, 2014, pp. 158–178. ISBN: 978-3-662-45610-1.

[19] Qingju Wang, Zhiqiang Liu, Kerem Varici, Yu Sasaki, Vincent Rijmen, and Yosuke Todo. "Cryptanalysis of Reduced-Round SIMON32 and SIMON48". In: *Progress in Cryptology - INDOCRYPT 2014*. Ed. by Willi Meier and Debdeep Mukhopadhyay. Vol. 8885. Lecture Notes in Computer Science. Springer, 2014, pp. 143–160. ISBN: 978-3-319-13038-5.

A Short tutorial for calculating differential probabilities and square correlations in SIMON-like round functions

The idea of this section is to complement the rigorous proofs with a more intuitive approach to calculating the differential probabilities and square correlation of one

round of SIMON. This should also allow us to better understand the Python code given later for calculating these values. We restrict ourselves to a simplified version of the SIMON round function:

$$f : \mathbb{F}_2^n \to \mathbb{F}_2^n \tag{1}$$

$$f(m) = S^1(m) \odot m \ . \tag{2}$$

Writing this equation bitwise where m_i denotes the ith bit of m we obtain:

$$f_i(m) = m_{i-1} \odot m_i \ . \tag{3}$$

When using a bit subscript, we will always implicitly assume that the subscript is calculated modulo n, the number of bits. Thus m_{-1} and m_{n-1} will for example refer to the same bit.

A.1 Differential probabilities

Suppose now we are given a message $m = (m_{n-1}, \ldots, m_1, m_0)$ and an input difference $d = (d_{n-1}, \ldots, d_1, d_0)$. Then the resulting difference D for the function f is calculated as $D(m, d) = f(m) \oplus f(m \oplus d)$. This can be written bitwise as:

$$D_i(m, d) = (m_{i-1} \odot m_i) \oplus ((m_{i-1} \oplus d_{i-1}) \odot (m_i \oplus d_i)) \ . \tag{4}$$

By differentiating between the four possible different cases for d_i and d_{i-1}, we obtain the following:

$$D_i(m, d) = \begin{cases} 0, & \text{if } d_i = 0 \text{ and } d_{i-1} = 0 \\ m_i, & \text{if } d_i = 0 \text{ and } d_{i-1} = 1 \\ m_{i-1}, & \text{if } d_i = 1 \text{ and } d_{i-1} = 0 \\ \overline{m_i \oplus m_{i-1}}, & \text{if } d_i = 1 \text{ and } d_{i-1} = 1 \ . \end{cases} \tag{5}$$

In the last case, D_i is 1 exactly when $m_i = m_{i-1}$ and is 0 when $m_i \neq m_{i-1}$.

Let us now look at a first example. Let $n = 6$, and $d = 001010$. We then calculate $D(m, d)$ using the above bitwise definition of D:

i	5	4	3	2	1	0
d	0	0	1	0	1	0
$S^1(d)$	0	1	0	1	0	0
$D(m, d)$	0	m_4	m_2	m_2	m_0	0

. (6)

We can see that the resulting difference depends only on m_4, m_2 and m_0. Thus by adapting these bits appropriately we can generate the following resulting differences:

$$000000, 000010, 001100, 001110, 010000, 010010, 011100, 011110.$$

All these differences then have the same probability of $8/64 = 1/8$. Note that the reuse of a message bit, m_2 in this case, is due to a subsequence 101 in the difference.

Let us take a look at another example. Let again $n = 6$ and now $d = 011010$. Then we can again calculate $D(m, d)$ as

i	5	4	3	2	1	0
d	0	1	1	1	1	0
$S^1(d)$	1	1	1	1	0	0
$D(m, d)$	m_5	$\overline{m_4 \oplus m_3}$	$\overline{m_3 \oplus m_2}$	$\overline{m_2 \oplus m_1}$	m_0	0

. (7)

We can see here that consecutive 1s in the input difference will cause the respective output difference bit to depend on two message bits. Nevertheless are all five non-zero output difference bits independent of each other. Thus 2^5 different output differences are possible, each one with probability 1/32.

With the observations made above, we can now devise a rule that allows us to determine the probability of a given pair (α, β) of an input difference α and an output difference β. First we calculate the set of `varibits` which is the bits in which the output difference can be non-zero. So output bit β_i is in `varibits` if and only if α_i or α_{i-1} is non-zero:

$$\texttt{varibits} = \alpha \mid S^1(\alpha) \tag{8}$$

where $\mid$ denotes the bitwise or.

Next we have to calculate which of these output difference bits have to be the same because of patterns 101 in the input difference. We do this by calculating the set `doublebits` which is the output difference bits that always have to be the same as the difference bit one position to the right. Thus β_i is in `doublebits` if and only if α_i is 1, α_{i-1} is 0, and α_{i-2} is 1.

$$\texttt{doublebits} = \alpha \odot \overline{S^1(\alpha)} \odot S^2(\alpha) \tag{9}$$

To check whether input difference α can map to output difference β with non-zero probability, we only need to check whether all non-zero bits of β lie in `varibits` and that all bits of β that are in `doublebits` are the same as the bits to their right. The probability of the transition is then determine by the number of output difference bits that can be chosen freely, i.e. the number of bits in `varibits` minus the number of bits in `doublebits`.

Before we write this procedure down, we have to take a look at one special case, namely when all input difference bits are set, e.g. $n = 6$ and $d = 111111$. Then we can again calculate $D(m, d)$ as

i	5	4	3	2	1	0
d	1	1	1	1	1	1
$S^1(d)$	1	1	1	1	1	1
$D(m, d)$	$\overline{m_5 \oplus m_4}$	$\overline{m_4 \oplus m_3}$	$\overline{m_3 \oplus m_2}$	$\overline{m_2 \oplus m_1}$	$\overline{m_1 \oplus m_0}$	$\overline{m_0 \oplus m_4}$

. (10)

Although all bits of the output are influenced and all bits of the input take equal influence, there are not 64 possible output differences since by switching all bits of m the output difference does not change.

So which output differences are possible then? By fixing an output difference, we get a sequence of equations of the kind $m_i = m_{i+1}$ or $m_i \neq m_{i+1}$. This creates a closed chain of equations that have to be coherent to be satisfiable. As a 0 in the output difference creates an inequality and a 1 creates an equality, in the end it boils down to the condition that the number of 0s in the output difference has to be even when the input difference only consists of 1s.

Let us now summarise all of this in a method to calculate the probability that a given input difference α is mapped to a given output difference β:

1. Check if α is the difference with all bits set to 1. If that is the case, calculate the number of 0s in β. If this number is even, return probability 2^{-n+1}, otherwise return probability 0. If α is not all 1s, go to next step.

2. Calculate $\texttt{varibits}$ as $\texttt{varibits} = \alpha \mid S^1(\alpha)$. Check if β has any bits set to 1 which are not in $\texttt{varibits}$, i.e. check if $\overline{\texttt{varibits}} \odot \beta \neq \mathbf{0}$. Should this be the case, return probability 0. Otherwise continue with next step.

3. Calculate $\texttt{doublebits}$ as $\texttt{doublebits} = \alpha \odot \overline{S^1(\alpha)} \odot S^2(\alpha)$. Check whether there are any bits of β in $\texttt{doublebits}$ that are not equal to their right neighbour, i.e. check $\big(\beta + S^1(\beta)\big) \odot \texttt{doublebits} \neq \mathbf{0}$. Should this be the case, return probability 0. Otherwise continue with next step.

4. Return probability $2^{-\operatorname{wt}(\texttt{varibits}+\texttt{doublebits})}$.

This method allows us to determine differential probabilities of the function $f(x) = S^1(x) \odot x$. We only have to apply some affine transformation to convert this to a method for calculating the probability of the SIMON round function. A Python implementation of the more general method can be found in Section B.

A.2 Square correlations

Let us now look at how to calculate square correlations for $f(x) = S^1(x) \odot x$.

First we look at the case where the input mask α is all 0s. Let $n = 6$ and let the output mask β be 010110:

$$\begin{array}{cccccc} \alpha & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline m & m_5 & m_4 & m_3 & m_2 & m_1 & m_0 \\ S^1(m) & m_4 & m_3 & m_2 & m_1 & m_0 & m_5 \\ \hline \beta & 0 & 1 & 0 & 1 & 1 & 0 \end{array} . \tag{11}$$

The resulting expression is then

$$m_4 m_3 + m_2 m_1 + m_1 m_0. \tag{12}$$

Let us look at the first term. It is zero in 3 out of 4 cases. It thus has a correlation of $\frac{1}{2}$ and hence a square correlation of $\frac{1}{4}$.

Let us look at the next two terms m_2m_1 and m_1m_0. First we note that they are not independent as they share the variable m_1. So we cannot calculate the square correlation of the sum of these terms as the product of the square correlations of the single terms. But we can rewrite the sum of these terms as

$$m_2m_1 + m_1m_0 = m_1(m_2 + m_0). \tag{13}$$

Now (m_2+m_0) behaves like a single one bit variable. Therefore the square correlation of $m_1(m_2+m_0)$ is $\frac{1}{4}$ as well. As m_4m_3 and $m_1(m_2+m_0)$ do not share any variables, the square correlation of the whole expression $m_4m_3+m_2m_1+m_1m_0$ is then $\frac{1}{4}\cdot\frac{1}{4} = \frac{1}{16}$. It is easy to see that different "blocks" of 1s in β that are separated by at least one 0, will generate independent terms. We thus only need to look at the square correlations of the terms generated from these blocks and multiply these to get the final result.

Let us thus look at a longer block of 1s with $\beta = 011111$:

α	0	0	0	0	0	0
m	m_5	m_4	m_3	m_2	m_1	m_0
$S^1(m)$	m_4	m_3	m_2	m_1	m_0	m_5
β	1	1	1	1	1	0

. (14)

The resulting expression is

$$m_5m_4 + m_4m_3 + m_3m_2 + m_2m_1 + m_1m_0. \tag{15}$$

By combining the first and the second as well as the third and the fourth term, we get

$$m_4(m_5 + m_3) + m_2(m_3 + m_1) + m_0m_1. \tag{16}$$

As $(m_5 + m_3)$, $(m_3 + m_1)$, and m_1 are independent of each other, the three terms $m_4(m_5 + m_3)$, $m_2(m_3 + m_1)$, and m_0m_1 are independent and the square correlation of the whole expression is thus $\left(\frac{1}{4}\right)^3 = 2^{-6}$.

At this point we already dare to formulate a rule. The square correlation of the term generated by m consecutive blocks of 1s is $2^{-2\lceil\frac{m}{2}\rceil}$. As every pair of consecutive single terms can be combined to create one independent term of square correlation 2^{-2}, the total square correlation just depends on the number of terms left after such pairing. And this number is $\lceil\frac{m}{2}\rceil$.

Let us now consider a non-zero input mask α. Let $\alpha = 010010$ and let $\beta = 010100$:

α	0	1	0	1	1	1
m	m_5	m_4	m_3	m_2	m_1	m_0
$S^1(m)$	m_4	m_3	m_2	m_1	m_0	m_5
β	0	1	0	1	0	0

. (17)

The resulting expression is then

$$m_4m_3 + m_4 + m_2m_1 + m_2 + m_1 + m_0. \tag{18}$$

We see that we can combine the first two terms to get a term of square correlation 2^{-2} again: $m_4m_3 + m_4 = m_4(m_3 + 1)$. Note that if the second term had been m_3 instead, it would have worked too. For the next three terms we can do the same: $m_2m_1 + m_2 + m_1 = (m_2 + 1)(m_1 + 1) + 1$. Note that the bias of this term is now flipped; the square correlation is nonetheless also 2^{-2}. As the first two terms are independent of the next three terms, the square correlation of the combined first five terms is 2^{-4}. But when looking at the last term m_0, we see that it is independent of all other terms and unbiased. Thus the square correlation of the complete expression is 0.

As a matter of fact, it is easy to see that when for any i the respective bit α_i of the input mask is 1 but both β_i and β_{i+1} are 0, the resulting expression will always be unbiased. Thus we can say that every non-zero bit in the input mask belonging to some block of 1s in the output mask is a necessary condition for the whole expression to be unbiased. Note that every bit in the input mask can at most be associated with one block of 1s in the output mask.

Thus we can evaluate the square correlation of f for an input mask α and an output mask β like this: First we check whether every non-zero bit in the input mask is associated to a block of 1s in the output mask. Is this not the case, we already know that the square correlation is zero. Otherwise we continue to partition the output mask and the input mask into blocks of 1s and their associated input mask bits. For each of these blocks we determing the square correlation of the resulting expression and finally multiply these together to get the total square correlation.

But how do we evaluate a block of 1s with the associated input mask bits in general? In the last example, we saw that for a block of a single 1 in the output mask, the two associated bits of the input mask can take any value; the square correlation remains 2^{-2}.

How about in the case of a block of two 1s? Let us look at the case of $\alpha = 111001$ and let $\beta = 110110$:

α	1	1	1	0	0	1
m	m_5	m_4	m_3	m_2	m_1	m_0
$S^1(m)$	m_4	m_3	m_2	m_1	m_0	m_5
β	1	1	0	1	1	0

. (19)

The resulting expression is

$$m_5m_4 + m_4m_3 + m_5 + m_4 + m_3 + m_2m_1 + m_1m_0 + m_0. \tag{20}$$

Let us first look at the first block of 1s in the output mask β, i.e. at the expression $m_5m_4 + m_4m_3 + m_5 + m_4 + m_3$. Combining the first two terms, we get

$$m_4(m_5 + m_3) + m_5 + m_4 + m_3. \tag{21}$$

We can now combine the first term with m_4 to get

$$m_4(m_5 + m_3 + 1) + m_5 + m_3. \tag{22}$$

Finally we can also incorporate m_5 and m_3 to get

$$(m_4 + 1)(m_5 + m_3 + 1) + 1. \tag{23}$$

The expression thus has a square correlation of 2^{-2}.

Let us look at the expression generated by the second block of 1s in the output mask:

$$m_2 m_1 + m_1 m_0 + m_0. \tag{24}$$

Combining the first two terms, we get

$$m_1(m_2 + m_0) + m_0. \tag{25}$$

But now we see that the term m_0 is independent of the first term. Thus we are left with a square correlation of 0. Note that the square correlation would also be 0 if the last term were m_2 but not if the last term were m_1 in which case the square correlation would be 2^{-2}.

As a matter of fact, the rule to determine the square correlation of an expression generated by a block of two 1s in the output mask and the associated bits in the input mask is straightforward. There are three associated input mask bits. If and only if both or none of the two outer bits (m_2 and m_0 in the last example) are set to 1, is the expression biased and the square correlation is 2^{-2}.

In fact, for a block of an even number of 1s in the ouput mask, any combination of associated input bits, will lead to a biased expression with the same square correlation. For a block of an odd number of 1s in the output mask, we need to check the input mask though. There is an odd number of associated bits to this block in the input mask. Let us refer to the first bit and then every second bit as the odd bits, and to the second bit and then every second bit as the even bits (from which direction we count does not matter). The even bits do not have an influence on the square correlation. But the parity of the odd bits determines whether the expression for this block will be unbiased or not. If and only if the parity is even, is the expresssion biased.

We can summarise a method to calculate the square correlation for a given input mask α and a given output mask β that is not all 1s as follows:

1. Partition the 1s in the output mask into consecutive blocks of 1s. The total square correlation is now the product of the square correlations for each block.

2. For each block calculate the square correlation:

   a) If the block length is odd, this block is always biased and the square correlation is solely determined by its length.

b) If the block lenght is even, we need to check the input mask. There is an odd number of bits in the input mask that are associated with this output block. Calculate the XOR of every second bit of these associated bits starting with the first one (such that both outer bits are considered). If this XOR sum is 1, the block is unbiased and thus the whole expression is unbiased. If the XOR sum is 0, the square correlation for this block is determined by its length.

For an implementation of the method to calculate the square correlation in Python, see Section 6.

B Python code to calculate lifferential probabilities and square correlations in SIMON-like round functions

In the following, code for calculating the differential probabilities and square correlations of SIMON-like round functions ($f_{a,b,c}(x) = S^a(x) \odot S^b(x) + S^c(x)$) are given in Python. Restrictions are that the constants need to fulfil $\gcd(a-b, n) = 1$. We assume that the functions $S^d(x)$ and $\mathrm{wt}(x)$ have been implemented as well as a function `parity` that calculates the parity $\mathrm{wt}(x) \mod 2$ of a bit vector x. a, b, and c have to be defined in the program as well.

The differential probability of $\alpha \xrightarrow{f} \beta$ can then be calculated with the following function:

```
def pdiff (alpha, beta):
    # Use gamma instead of beta to get rid of the
    # linear part
    gamma = beta ^ S(alpha, c)
    # Take care of the case where alpha is all 1s
    if alpha == 2**n-1:
        if hw(~gamma)%2 == 0:
            return 2**(n-1)
        else:
            return 0
    # Determine bits that can take a nonzero difference
    varibits = S(alpha, a) | S(alpha, b)
    # Check whether gamma conforms with varibits
    if gamma & ~varibits != 0:
        return 0
    # Determine the bits that are duplicates
    doublebits = S(alpha, 2 * a - b) & ~S(alpha, a) &
                 S(alpha, b)
    # Check whether the duplicate bits are the same as
    # there counterpart
```

```
    if (gamma ^ S(gamma, a - b)) & doublebits != 0:
        return 0
    return 2**(-hw(varibits ^ doublebits))
```

The squared correlation of $\alpha \xrightarrow{f} \beta$ can be calculated with the following function. Here we assume n to be even, which is relevant for the case where β is all 1s.

```
def plin (alpha, beta):
    # Get rid of linear part of round function
    alpha ^= S(beta, -c)
    # If the input masks uses bits that have
    # corresponding bits in the output mask, the
    # correlation is 0.
    if ((S(beta, -a) | S(beta, -b)) ^ alpha) &
                                        alpha != 0:
        return 0
    #Take care of the case where beta is all 1s
    if beta == 2**n-1:
            t, v = alpha, 0
            while t != 0:
                v ^= t & 3
                t >>= 2
            if v != 0:
                return 0
            else:
                return 2**(-n + 2)
    # Set in the abits mask the first and then every
    # second bit of each block of 1s in the output
    # mask beta. Each corresponds to one independent
    # multiplication term, and thus adds a factor of
    # 2^(-2) to the square correlation.
    # Example:
    #   beta = 0111101110110 -> abits = 0101001010100
    tmp = beta
    abits = beta
    while tmp != 0:
        tmp = beta & S(tmp, -(a - b))
        abits ^= tmp
    # The sbits correspond to bits one to the right of
    # each block of an even number of 1s in the output
    # mask.
    # Example:
    #   beta = 0111101110110 -> sbits = 0000010000001
```

```
    sbits = S(beta, -(a - b)) &
                            ~beta & ~S(abits, -(a - b))
    # Adopt sbits to correspond to the respective bits
    # in the input mask
    sbits = S(sbits, -b)
    # The pbits are used to check whether the input
    # mask removes the bias from one of the output
    # mask blocks. It checks the parity of the sum of
    # every second inputmask bit for each block that
    # corresponds to a block of an even number of 1s
    # in the output mask.
    pbits = 0
    while sbits != 0:
        pbits ^= sbits & alpha
        sbits = S(sbits, (a - b)) & S(beta, -b)
        sbits = S(sbits, (a - b))
        pbits = S(pbits, 2 * (a - b))
    # If the parity is uneven for any one of the
    # blocks, there is no bias.
    if pbits != 0:
        return 0
    return 2**(-2 * hw(abits))
```

C Additional Differential Bounds

In Table 4 resp. 5 we give the distributions for the characteristics contributing to a differential up to the bound we computed them.

D Optimal parameters for differential characteristics

The following sets of rotation constants (a, b, c) are optimal for 10 rounds regarding differential characteristics for Simon32, Simon48, and Simon64

$(1,0,2),(1,0,3),(2,1,3),(4,3,5),(5,0,10),(5,0,15),(5,4,3),(7,0,14),(7,6,5)$
$(8,1,3),(8,3,14),(8,7,5),(10,5,15),(11,6,1),(12,1,7),(12,5,3),(12,7,1)$
$(13,0,10),(13,0,7),(13,8,2)$

Similar to the experiments for the default parameters, we used our framework to evaluate the quality of various rotation constants. In Table 7 we give an overview of the best differential characteristics for variants of Simon using a different set of

rotation constants. Table 6 shows that a carefully chosen set of constants can have a very strong effect on the differentials.

Table 4: For the 17-round differential $(80, 222) \xrightarrow{f^{17}} (222, 80)$ in SIMON48, the number of differential characteristics are listed here.

$\log_2(p)$	#Characteristics	$\log_2(p)$	#Characteristics
-52	1	-69	20890
-53	6	-70	38837
-54	15	-71	72822
-55	46	-72	133410
-56	100	-73	240790
-57	208	-74	353176
-58	379	-75	279833
-59	685	-76	235071
-60	1067	-77	259029
-61	1607	-78	225836
-62	2255	-79	256135
-63	2839	-80	252193
-64	3476	-81	252654
-65	4088	-82	198784
-66	5032	-83	229843
-67	7063	-84	208757
-68	11481	-85	253112

Table 5: For the 21-round differential $(4000000, 11000000) \xrightarrow{f^{21}} (11000000, 4000000)$ in SIMON64, the number of differential characteristics are listed here.

$\log_2(p)$	#Characteristics	$\log_2(p)$	#Characteristics
-68	2	-83	185709
-69	14	-84	173860
-70	70	-85	171902
-71	276	-86	171302
-72	951	-87	168190
-73	2880	-88	164694
-74	8101	-89	163141
-75	21062	-90	161089
-76	52255	-91	159354
-77	123206	-92	155804
-78	238297	-93	150954
-79	239305	-94	145061
-80	171895	-95	141914
-81	170187	-96	138480
-82	165671	-97	132931

Table 6: Distribution of the characteristics for a 13-round differential for SIMON32 using different set of constants.

$\log_2(p)$	$[8,1,2]$	$[12,5,3]$	$[7,0,2]$	$[1,0,2]$
-36	1	1	4	1
-37	4	2	16	6
-38	15	3	56	27
-39	46	2	144	88
-40	124	1	336	283
-41	288	0	744	822
-42	673	0	1644	2297
-43	1426	0	3420	6006
-44	2973	0	6933	14954
-45	5962	0	13270	34524
-46	11661	1	24436	73972
-47	21916	3	43784	150272
-48	40226	14	76261	292118
-49	72246	32	130068	/
-50	126574	54	218832	/
-51	218516	83	362284	/

Table 7: Overview of the optimal differential characteristics for SIMON variants.

Rounds:	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Differential $(12,5,3)$															
SIMON32	-2	-4	-6	-8	-12	-14	-18	-20	-26	-28	-34	-36	-42	-44	-47
SIMON48	-2	-4	-6	-8	-12	-14	-18	-20	-26	-30	-36	-36	-38	-40	-42
SIMON64	-2	-4	-6	-8	-12	-14	-18	-20	-26	-30	-35	-37	-43	-47	/
Differential $(1,0,2)$															
SIMON32	-2	-4	-6	-8	-12	-14	-18	-20	-26	-30	-36	-36	-38	-40	-42
SIMON48	-2	-4	-6	-8	-12	-14	-18	-20	-26	-30	-36	-38	-44	-48	-54
SIMON64	-2	-4	-6	-8	-12	-14	-18	-20	-26	-30	-36	-38	-44	-48	-54
Differential $(7,0,2)$															
SIMON32	-2	-4	-6	-8	-12	-14	-18	-20	-25	-30	-35	-36	-38	-40	-42
SIMON48	-2	-4	-6	-8	-12	-14	-18	-20	-26	-30	-35	-38	-44	-48	-53
SIMON64	-2	-4	-6	-8	-12	-14	-18	-20	-26	-30	-36	-38	-44	-48	/

Table 8: For each SIMON variant and each possible number of rounds, the number of possible combinations of rotation constants (a, b, c) with $a \geq b$ is given that reaches full diffusion.

SIMON32

Rounds	6	7	8	9	10	11	17	∞
$\#(a, b, c)$	48	600	528	88	144	128	64	576

SIMON48

Rounds	7	8	9	10	11	13	14	15	25	∞
$\#(a, b, c)$	48	1392	1680	792	528	344	144	128	64	2080

SIMON64

Rounds	8	9	10	11	12	13	15	17	18	19	33	∞
$\#(a, b, c)$	384	4800	2112	2256	1152	608	512	48	288	256	128	4352

SIMON96

Rounds	9	10	11	12	13	14	15	16	17
$\#(a, b, c)$	336	4272	13920	7104	5568	3456	912	1152	800
	19	21	25	26	27	49	∞		
	1568	640	48	288	256	128	16000		

SIMON128

Rounds	10	11	12	13	14	15	16	17	18	19	20
$\#(a, b, c)$	768	10944	26112	25536	9024	6912	7488	2496	192	1824	2304
	21	23	24	25	33	34	35	65	∞		
	1792	1024	960	512	96	576	512	256	33792		

Polytopic Cryptanalysis

Publication Information

Tyge Tiessen. “Polytopic Cryptanalysis”. In: *Advances in Cryptology - EUROCRYPT 2016*. Ed. by Jean-Sébastien Coron and Marc Fischlin. Vol. ??? Lecture Notes in Computer Science. Springer, 2016, ??? ISBN: ???????

Contribution

- Single author.

Remarks

This publication has been slightly edited to fit the format.

Polytopic Cryptanalysis

Tyge Tiessen

DTU Compute, Technical University of Denmark, Kgs. Lyngby, Denmark
tyti@dtu.dk

Abstract. Standard differential cryptanalysis uses statistical dependencies between the difference of two plaintexts and the difference of the respective two ciphertexts to attack a cipher. Here we introduce polytopic cryptanalysis which considers interdependencies between larger sets of texts as they traverse through the cipher. We prove that the methodology of standard differential cryptanalysis can unambiguously be extended and transferred to the polytopic case including impossible differentials. We show that impossible polytopic transitions have generic advantages over impossible differentials. To demonstrate the practical relevance of the generalization, we present new low-data attacks on round-reduced DES and AES using impossible polytopic transitions that are able to compete with existing attacks, partially outperforming these.

1 Introduction

Without doubt is differential cryptanalysis one of the most important tools that the cryptanalyst has at hand when trying to evaluate the security of a block cipher. Since its conception by Biham and Shamir [2] in their effort to break the Data Encryption Standard [26], it has been successfully applied to many block ciphers such that any modern block cipher is expected to have strong security arguments against this attack.

The methodology of differential cryptanalysis has been extended several times with a number of attack vectors, most importantly truncated differentials [20], impossible differentials [1], [19], and higher-order differentials [20], [22]. Further attacks include the boomerang attack [29], which bears some resemblance of second-order differential attacks, and differential-linear attacks [24].

Nonetheless many open problems remain in the field of differential cryptanalysis. Although the concept of higher-order differentials is almost 20 years old, it has not seen many good use cases. One reason has been the difficulty of determining the probability of higher-order differentials accurately without evaluating Boolean functions with prohibitively many terms. Thus the common use case remains

probability 1 higher-order differentials where we know that a derivative of a certain order has to evaluate to zero because of a limit in the degree of the function.

Another open problem is the exact determination of the success probability of boomerang attacks and their extensions. It has correctly been observed that the correlation between differentials must be taken into account to accurately determine the success probability [25]. The true probability can otherwise deviate arbitrarily from the estimated one.

Starting with Chabaud and Vaudenay [12], considerable effort has gone into shedding light on the relation and interdependencies of various cryptographic attacks (see for example [5], [6], [30]). With this paper, we offer a generalized view on the various types of differential attacks that might help to understand both the interrelation between the attacks as well as the probabilities of the attacks better.

Our contribution

In this paper we introduce polytopic cryptanalysis. It can be viewed as a generalization of standard differential cryptanalysis which it embeds as a special case. We prove that the definitions and methodology of differential cryptanalysis can unambiguously be extended to polytopic cryptanalysis, including the concept of impossible differentials. Polytopic cryptanalysis is general enough to even encompass attacks such as higher-order differentials and might thus be valuable as a reference framework.

For impossible polytopic transitions, we show that they exhibit properties that allow them to be very effective in scenarios where ordinary impossible differentials fail. This is mostly due to a generic limit in the diffusion of any block cipher that guarantees that only a negligible number of all polytopic transitions is possible for a sufficiently high choice of dimension. This also makes impossible polytopic transitions ideal for low-data attacks where standard impossible differentials usually have a high data complexity.

Finally we prove that polytopic cryptanalysis is not only theoretically intriguing but indeed relevant for practical cryptanalysis by demonstrating competitive impossible polytopic attacks on round-reduced DES and AES that partly outperform existing low-data attacks and offer different trade-offs between time and data complexity.

In the appendix, we further prove that higher-order differentials can be expressed as truncated polytopic transitions and are hence a special case of these. Thus higher-order differentials can be expressed in terms of a collection of polytopic trails just as differentials can be expressed as a collection of differential trails. A consequence of this is that it is principally possible to determine lower bounds for the probability of a higher-order differential by summing over the probabilities of a subset of the polytopic trails which it contains.

Related work

To our knowledge, the concept of polytopic transitions is new and has not been used in cryptanalysis before. Nonetheless there is other work that shares some similarities

with polytopic cryptanalysis.

Higher-order differentials [22] can in some sense be seen as a higher-dimensional version of a differential. However, most concepts of ordinary differentials do not seem to extend to higher-order differentials, such as characteristics or iterated differentials.

The idea of using several differentials simultaneously in an attack is not new (see for example [4]). However as opposed to assuming independence of the differentials, which does not hold in general (see [25]), we explicitly take their correlation into account and use it in our framework.

Another type of cryptanalysis that uses a larger set of texts instead of a single pair is integral cryptanalysis (see for example [3], [14]), in which structural properties of the cipher are used to elegantly determine a higher-order derivative to be zero without relying on bounds in the degree. These attacks can be considered a particular form of higher-order differentials.

Finally decorrelation theory [28] also considers relations between multiple pairs of plaintexts and ciphertexts but takes a different direction by considering security proofs based on a lack of correlation between the texts.

Organization of the paper

In Section 2, notation and concepts necessary for polytopic cryptanalysis are introduced. It is demonstrated how the concepts of differential cryptanalysis naturally extend to polytopic cryptanalysis. We also take a closer look at the probability of polytopic transitions and applicability of simple polytopic cryptanalysis.

In Section 3, we introduce impossible polytopic transitions. We show that impossible polytopic transitions offer some inherent advantages over impossible differentials and are particularly interesting for low-data attacks. We show that, given an efficient method to determine the possibility of a polytopic transition, generic impossible polytopic attack always exist.

In Section 4, we demonstrate the practicability of impossible polytopic transition attacks. We present some attacks on DES and AES that are able to compete with existing attacks with low-data complexity, partially outperforming these.

Furthermore, in Section B truncated polytopic transitions are introduced. We then give a proof that higher-order differentials are a special case of these. The cryptanalytic ramifications of the fact that higher-order differentials consist of polytopic trails are then discussed.

Notation

We use $\mathbb{F}_2^n$ to denote the n-dimensional binary vector space. To identify numbers in hexadecimal notation we use a typewriter font as in `3af179`. Random variables are denoted with bold capital letters ($\mathbf{X}$). We will denote d-difference (introduced later) by bold Greek letters ($\boldsymbol{\alpha}$) and standard differences by Roman (i.e., non-bold) Greek letters (α).

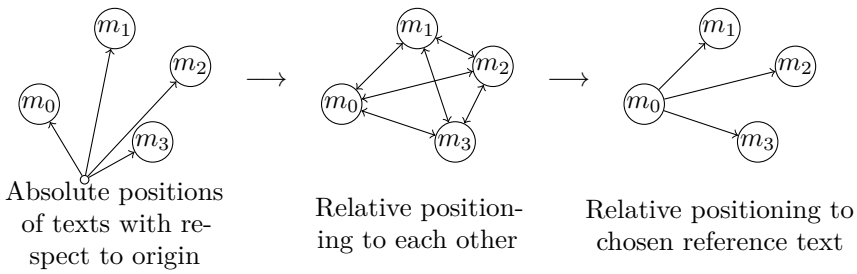

Figure 1: Depiction of three views of a polytope with four vertices.

2 Polytopes and polytopic transitions

Classical differential cryptanalysis utilizes the statistical interdependency of two texts as they traverse through the cipher. When we are not interested in the absolute position of the two texts in the state space, the difference between the two texts completely determines their relative positioning.

But there is no inherent reason that forces us to be restricted to only using a pair of texts. Let us instead consider an ordered set of texts as they traverse through the cipher.

Definition 1 (s-polytope)**.** An s-polytope in $\mathbb{F}_2^n$ is an s-tuple of values in $\mathbb{F}_2^n$.

Similar to differential cryptanalysis, we are not so much interested in the absolute position of these texts but the relations between the texts. If we choose one of the texts as the point of reference, the relations between all texts are already uniquely determined by only considering their differences with respect to the reference text. If we thus have $d+1$ texts, we can describe their relative positioning by a tuple of d differences (see also Fig. 1).

Definition 2 (d-difference)**.** A d-difference over $\mathbb{F}_2^n$ is a d-tuple of values in $\mathbb{F}_2^n$ describing the relative position of the texts of a $(d+1)$-polytope from one point of reference.

When we reduce a $(d+1)$-polytope to a corresponding d-difference, we loose the information of the absolute position of the polytope. A d-difference thus corresponds to an equivalence class of $(d+1)$-polytopes where polytopes are equivalent if and only if they can be transformed into each other by simple shifting in state space. We will mostly be dealing with these equivalence classes.

In principal there are many d-differences that correspond to one $(d+1)$-polytope depending on the choice of reference text and the order of the differences. As a convention we will construct a d-difference from a $(d+1)$-polytope as follows:

Convention. *For a $(d+1)$-polytope $(m_0, \ldots, m_d)$, the corresponding d-difference is created as $(m_0 \oplus m_1, m_0 \oplus m_2, \ldots, m_0 \oplus m_d)$.*

This means, we use the first text of the polytope as the reference text and write the differences in the same order as the remaining texts of the polytope. We will call the reference text the *anchor* of the d-difference. Hence if we are given a d-difference and the value of the anchor, we can reconstruct the corresponding $(d+1)$-polytope uniquely.

Example. Let (m_0, m_1, m_2, m_3) be a 4-polytope in $\mathbb{F}_2^n$. Then $(m_0 \oplus m_1, m_0 \oplus m_2, m_0 \oplus m_3)$ is the corresponding 3-difference with m_0 as the anchor.

In the following, we will now show that we can build a theory of polytopic cryptanalysis in which the same methodology as in standard differential cryptanalysis applies. Standard differential cryptanalysis is contained in this framework as a special case.

A short note regarding possible definitions of difference: in this paper we restrict ourselves to XOR-differences as the most common choice. Most, if not all, statements in this paper naturally extend to other definitions of difference, e.g., in modular arithmetic.

The equivalent of a differential in polytopic cryptanalysis is the polytopic transition. We use d-differences for the definition.

Definition 3 (Polytopic transition with fixed anchor)**.** Let $f : \mathbb{F}_2^n \to \mathbb{F}_2^q$. Let $\boldsymbol{\alpha}$ be a d-difference $(\alpha_1, \alpha_2, \ldots, \alpha_d)$ over $\mathbb{F}_2^n$ and let $\boldsymbol{\beta}$ be the d-difference $(\beta_1, \beta_2, \ldots, \beta_d)$ over $\mathbb{F}_2^q$. By the $(d+1)$-polytopic transition $\boldsymbol{\alpha} \xrightarrow[x]{f} \boldsymbol{\beta}$ we denote that f maps the polytope corresponding to $\boldsymbol{\alpha}$ with anchor x to a polytope corresponding to $\boldsymbol{\beta}$. More precisely, we have $\boldsymbol{\alpha} \xrightarrow[x]{f} \boldsymbol{\beta}$ if and only if

$$\begin{aligned} & f(x \oplus \alpha_1) \oplus f(x) = \beta_1 \\ \text{and } & f(x \oplus \alpha_2) \oplus f(x) = \beta_2 \\ & \ldots \\ \text{and } & f(x \oplus \alpha_d) \oplus f(x) = \beta_d. \end{aligned}$$

Building up on this definition, we can now define the probability of a polytopic transition under a random anchor.

Definition 4 (Polytopic transition)**.** Let f, $\boldsymbol{\alpha}$, and $\boldsymbol{\beta}$ again be as in Definition 3. The probability of the $(d+1)$-polytopic transition $\boldsymbol{\alpha} \xrightarrow{f} \boldsymbol{\beta}$ is then defined as:

$$\Pr\left(\boldsymbol{\alpha} \xrightarrow{f} \boldsymbol{\beta}\right) := \Pr_{\mathbf{X}}\left(\boldsymbol{\alpha} \xrightarrow[\mathbf{X}]{f} \boldsymbol{\beta}\right) \tag{1}$$

where $\mathbf{X}$ is a random variable, uniformly distributed on $\mathbb{F}_2^n$. We will at times also write $\boldsymbol{\alpha} \longrightarrow \boldsymbol{\beta}$ if the function is clear from the context or not important.

Note that this definition coincides with the definition of the differential probability when differences between only two texts (2-polytopes) are considered.

Let $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$ now be a function that is the repeated composition of round functions $f_i : \mathbb{F}_2^n \to \mathbb{F}_2^n$:

$$f := f_r \circ \cdots \circ f_2 \circ f_1. \tag{2}$$

Similarly to differential cryptanalysis, we can now define trails of polytopes:

Definition 5 (Polytopic trail)**.** Let f be as in Eq. (2). A polytopic trail on f is an $(r+1)$-tuple of d-differences $(\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_r)$ written as

$$\boldsymbol{\alpha}_0 \xrightarrow{f_1} \boldsymbol{\alpha}_1 \xrightarrow{f_2} \cdots \xrightarrow{f_r} \boldsymbol{\alpha}_r. \tag{3}$$

The probability of such a polytopic trail is defined as

$$\Pr_{\mathbf{X}}\left(\boldsymbol{\alpha}_0 \xrightarrow[\mathbf{X}]{f_1} \boldsymbol{\alpha}_1 \text{ and } \boldsymbol{\alpha}_1 \xrightarrow[f_1(\mathbf{X})]{f_2} \boldsymbol{\alpha}_2 \text{ and } \cdots \text{ and } \boldsymbol{\alpha}_{r-1} \xrightarrow[f_{r-1}\circ\cdots\circ f_1(\mathbf{X})]{f_r} \boldsymbol{\alpha}_r\right) \tag{4}$$

where $\mathbf{X}$ is a random variable, distributed uniformly on $\mathbb{F}_2^n$.

Similarly to differentials, it is possible to partition a polytopic transition over a composed function into all polytopic trails that feature the same input and output differences as the polytopic transition.

Proposition 6. *The probability of a polytopic transition $\boldsymbol{\alpha}_0 \xrightarrow{f} \boldsymbol{\alpha}_r$ over a function $f : \mathbb{F}_2^n \to \mathbb{F}_2^n, f = f_r \circ \cdots \circ f_2 \circ f_1$ is the sum of the probabilities of all polytopic trails $(\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_r)$ which it contains:*

$$\Pr\left(\boldsymbol{\alpha}_0 \xrightarrow{f} \boldsymbol{\alpha}_r\right) = \sum_{\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_{r-1}} \Pr\left(\boldsymbol{\alpha}_0 \xrightarrow{f_1} \boldsymbol{\alpha}_1 \xrightarrow{f_2} \cdots \xrightarrow{f_{r-1}} \boldsymbol{\alpha}_{r-1} \xrightarrow{f_r} \boldsymbol{\alpha}_r\right) \tag{5}$$

where $\boldsymbol{\alpha}_0, \ldots, \boldsymbol{\alpha}_r$ are d-differences and as such lie in $\mathbb{F}_2^{dn}$.

Proof. If we fix the initial value of the anchor, we also fix the trail that the polytope has to take. The set of polytopic trails gives us thus a partition of the possible anchor values and in particular a partition of the anchors for which the output polytope is

of type $\boldsymbol{\alpha}_r$. Using the above definitions we thus get:

$$\begin{aligned}
\Pr\left(\boldsymbol{\alpha}_0 \xrightarrow{f} \boldsymbol{\alpha}_r\right) &= \Pr_{\mathbf{X}}\left(\boldsymbol{\alpha}_0 \xrightarrow[\mathbf{X}]{f} \boldsymbol{\alpha}_r\right) \\
&= 2^{-n} \cdot \left|\left\{x \in \mathbb{F}_2^n \;\middle|\; \boldsymbol{\alpha}_0 \xrightarrow[x]{f} \boldsymbol{\alpha}_r\right\}\right| \\
&= 2^{-n} \cdot \sum_{\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_{r-1}} \left|\left\{x \in \mathbb{F}_2^n \;\middle|\; \boldsymbol{\alpha}_0 \xrightarrow[x]{f_1} \boldsymbol{\alpha}_1,\ \boldsymbol{\alpha}_1 \xrightarrow[f_1(x)]{f_2} \boldsymbol{\alpha}_2, \ldots \right.\right. \\
&\qquad\qquad \left.\left. \ldots,\ \boldsymbol{\alpha}_{r-1} \xrightarrow[f_{r-1}\circ\cdots\circ f_1(x)]{f_r} \boldsymbol{\alpha}_r\right\}\right| \\
&= \sum_{\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_{r-1}} \Pr_{\mathbf{X}}\left(\boldsymbol{\alpha}_0 \xrightarrow[\mathbf{X}]{f_1} \boldsymbol{\alpha}_1 \text{ and } \boldsymbol{\alpha}_1 \xrightarrow[f_1(\mathbf{X})]{f_2} \boldsymbol{\alpha}_2 \text{ and } \ldots \right. \\
&\qquad\qquad \left. \ldots \text{ and } \boldsymbol{\alpha}_{r-1} \xrightarrow[f_{r-1}\circ\cdots\circ f_1(\mathbf{X})]{f_r} \boldsymbol{\alpha}_r\right) \\
&= \sum_{\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_{r-1}} \Pr\left(\boldsymbol{\alpha}_0 \xrightarrow{f_1} \boldsymbol{\alpha}_1 \xrightarrow{f_2} \cdots \xrightarrow{f_{r-1}} \boldsymbol{\alpha}_{r-1} \xrightarrow{f_r} \boldsymbol{\alpha}_r\right)
\end{aligned}$$

which proves the proposition. □

To be able to calculate the probability of a differential trail, it is common in differential cryptanalysis to make an assumption on the independence of the round transitions. This is usually justified by showing that the cipher is a Markov cipher and by assuming the stochastic equivalence hypothesis (see [23]). As we will mostly be working with impossible trails where these assumptions are not needed, we will assume for now that this independence holds and refer the interested reader to Section A where the Markov model is adapted to polytopic cryptanalysis.

Under the assumption that the single round transitions are independent, we can work with polytopic transitions just as with differentials:

1. The probability of a polytopic transition is the sum of the probabilities of all polytopic trails with the same input and output d-difference.

2. The probability of a polytopic trail is the product of the probabilities of the 1-round polytopic transitions that constitute the trail.

We are thus principally able to calculate the probability of a polytopic transition over many rounds by knowing how to calculate the polytopic transition over single rounds.

Now to calculate the probability of a 1-round polytopic transition, we can use the following observations:

3. A linear function maps a d-difference with probability 1 to the d-difference that is the result of applying the linear function to each single difference in the d-difference.

4. Addition of a constant to the anchor leaves the d-difference unchanged.
5. The probability of a polytopic transition over an S-box layer is the product of the polytopic transitions for each S-box.

We are thus able to determine probabilities of polytopic transitions and polytopic trails just as we are used to from standard differential cryptanalysis.

A note on correlation, diffusion and the difference distribution table

When estimating the probability of a polytopic transition a first guess might be that it is just the product of the individual 1-dimensional differentials. For a 3-polytopic transition we might for example expect:

$$\Pr\Big((\alpha_0, \alpha_1) \longrightarrow (\beta_0, \beta_1)\Big) \stackrel{?}{=} \Pr\Big(\alpha_0 \longrightarrow \beta_0\Big) \cdot \Pr\Big(\alpha_1 \longrightarrow \beta_1\Big).$$

That this is generally *not* the case is a consequence of the following lemma.

Lemma 7. *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$. For a given input d-difference $\boldsymbol{\alpha}$ the number of output d-differences to which $\boldsymbol{\alpha}$ is mapped with non-zero probability is upper bounded by 2^n.*

Proof. This is just a result of the fact that the number of anchors for the transition is limited to 2^n:

$$\left|\left\{\boldsymbol{\beta} \in \mathbb{F}_2^{dn} \;\middle|\; \Pr\Big(\boldsymbol{\alpha} \xrightarrow{f} \boldsymbol{\beta}\Big) > 0\right\}\right| = \left|\left\{\boldsymbol{\beta} \in \mathbb{F}_2^{dn} \;\middle|\; \exists x \in \mathbb{F}_2^n : \boldsymbol{\alpha} \xrightarrow[x]{f} \boldsymbol{\beta}\right\}\right| \leq 2^n$$

□

One implication of this limitation of possible output d-differences is a correlation between differentials: the closer the distribution of differences of a function is to a uniform distribution, the stronger is the correlation of differentials over that function.

Example. Let us take the AES 8-bit S-box (denoted by S here) which is differentially 4-uniform. Consider the three differentials, $\mathtt{7} \xrightarrow{S} \mathtt{166}$, $\mathtt{25} \xrightarrow{S} \mathtt{183}$, and $\mathtt{25} \xrightarrow{S} \mathtt{1}$ which have probabilities 2^{-6}, 2^{-6}, and 2^{-7} respectively. The probabilities of the polytopic transitions of the combined differentials deviate strongly from the product of the single probabilities:

$$\Pr\Big((\mathtt{7}, \mathtt{25}) \xrightarrow{S} (\mathtt{166}, \mathtt{183})\Big) = 2^{-6} > \Pr\Big(\mathtt{7} \xrightarrow{S} \mathtt{166}\Big) \cdot \Pr\Big(\mathtt{25} \xrightarrow{S} \mathtt{183}\Big) = 2^{-12}$$
$$\Pr\Big((\mathtt{7}, \mathtt{25}) \xrightarrow{S} (\mathtt{166}, \mathtt{1})\Big) = 0 < \Pr\Big(\mathtt{7} \xrightarrow{S} \mathtt{166}\Big) \cdot \Pr\Big(\mathtt{25} \xrightarrow{S} \mathtt{1}\Big) = 2^{-13}.$$

Another consequence of Lemma 7 is that it sets an inherent limit to the maximal diffusion possible over one round. A one d-difference can at most be mapped to 2^n possible d-differences over one round, the number of possible d-differences reachable

can only increase by a factor of 2^n over each round. Thus when starting from one d-difference, after one round at most 2^n d-differences are possible, after two rounds at most 2^{2n} differences are possible, after three rounds at most 2^{3n} are possible and generally after round r at most 2^{rn} d-differences are possible.

In standard differential cryptanalysis, the number of possible output differences for a given input difference is limited by the state size of the function. This is no longer true for d-differences: if the state space is $\mathbb{F}_2^n$, the space of d-differences is $\mathbb{F}_2^{dn}$. The number of possible d-differences thus increases exponentially with the dimension d. This has a consequence for the size of the difference distribution table (DDT). For an 8-bit S-box, a classical DDT has a size of 2^{16} entries, i.e., 64 kilobytes. But already the DDT for 3-differences has a size of 2^{48}, i.e., 256 terabytes. Fortunately though, a third consequence of Lemma 7 is that the DDT table is sparse for $d > 1$. As a matter of fact, we can calculate any row of the DDT with a time complexity of 2^n by trying out all possible values for the anchor.

Relation to decorrelation theory.

Decorrelation theory [28] is a framework that can be used to design ciphers which are provably secure against a range of attacks including differential and linear cryptanalysis. A cipher is called perfectly decorrelated of order d when the image of any d-tuple of distinct plaintexts is uniformly distributed on all d-tuples of ciphertexts with distinct values under a uniformly distributed random key. It can for example be proved that a cipher which is perfectly decorrelated of order 2 is secure against standard differential and linear cryptanalysis.

When we consider $(d+1)$-polytopes in polytopic cryptanalysis, we can naturally circumvent security proofs for order-d perfectly decorrelated ciphers. The boomerang attack [29] for example – invented to break an order-2 perfectly decorrelated cipher – can be described as a 4-polytopic attack.

Limitations of simple polytopic cryptanalysis

Can simple polytopic cryptanalysis, i.e., using a single polytopic transition, outperform standard differential cryptanalysis? Unfortunately this is generally not the case as is shown in the following.

Definition 8. Let $\boldsymbol{\alpha} \longrightarrow \boldsymbol{\beta}$ be a $(d+1)$-polytopic transition with d-differences $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. Let $\boldsymbol{\alpha}' \longrightarrow \boldsymbol{\beta}'$ be a d'-difference with $d' \leq d$. We then write $(\boldsymbol{\alpha}', \boldsymbol{\beta}') \sqsubseteq (\boldsymbol{\alpha}, \boldsymbol{\beta})$ if and only if for each $i \in [1, d']$ there exists $j \in [1, d]$ such that the ith differences in $\boldsymbol{\alpha}'$ and $\boldsymbol{\beta}'$ correspond to the jth differences in $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$.

Using this notation, we have the following lemma:

Lemma 9. *Let $\boldsymbol{\alpha} \longrightarrow \boldsymbol{\beta}$ be a $(d+1)$-polytopic transition and let $\boldsymbol{\alpha}' \longrightarrow \boldsymbol{\beta}'$ be a $(d'+1)$-polytopic transition with $d' \leq d$ and $(\boldsymbol{\alpha}', \boldsymbol{\beta}') \sqsubseteq (\boldsymbol{\alpha}, \boldsymbol{\beta})$. Then*

$$\Pr(\boldsymbol{\alpha} \longrightarrow \boldsymbol{\beta}) \leq \Pr(\boldsymbol{\alpha}' \longrightarrow \boldsymbol{\beta}'). \tag{6}$$

Proof. This follows directly from the fact that $\boldsymbol{\alpha} \xrightarrow[x]{f} \boldsymbol{\beta}$ implies $\boldsymbol{\alpha}' \xrightarrow[x]{f} \boldsymbol{\beta}'$. □

In words, the probability of a polytopic transition is always at most as high as the probability of any lower dimensional polytopic transition that it contains. In particular, it can never have a higher probability than any standard differential that it contains.

It can in some instances still be profitable to use a single polytopic transition instead of a standard differential that it contains. This is the case when the probability of the polytopic transition is the same as (or close to) the probability of the best standard differential it contains. Due to the fact that the space of d-differences is much larger than that of standard differentials (2^{dn} vs. 2^n), one set of texts that follows the polytopic transition is usually enough to distinguish the biased distribution from a uniform distribution as opposed to standard differentials where at least two are needed. Nonetheless the cryptanalytic advantages of polytopic cryptanalysis lie elsewhere as we will see in the next sections.

3 Impossible polytopic cryptanalysis

Impossible differential cryptanalysis makes use of differentials with probability zero to distinguish a cipher from an ideal cipher. In this section, we extend the definition to encompass polytopic transitions.

Impossible polytopic cryptanalysis offers distinct advantages over standard impossible differential cryptanalysis that are a result of the exponential increase in the size of the space of d-differences with increasing dimension d. This not only allows impossible $(d+1)$-polytopic attacks using just a single set of $d+1$ chosen plaintexts, it also allows generic distinguishing attacks on $(d-1)$-round block ciphers whenever it is computationally easy to determine whether a $(d+1)$-polytopic transition is possible or not. We elaborate this in more detail later in this section.

Definition 10. An impossible $(d+1)$-polytopic transition is a $(d+1)$-polytopic transition that occurs with probability zero.

In impossible differential attacks, we use knowledge of an impossible differential over $r-1$ rounds to filter out wrong round key guesses for the last round: any round key that decrypts a text pair such that their difference adheres to the impossible differential has to be wrong. The large disadvantage of this attack is that it always requires a large number of text pairs to sufficiently reduce the number of possible keys. This is due to the fact that the filtering probability corresponds to the fraction of the impossible differentials among all differentials. Unfortunately for the attacker, most ciphers are designed to provide good diffusion, such that this ratio is usually low after a few rounds.

This is exactly where the advantage of impossible polytopic transitions lies. Due to the exponential increase in the size of the space of d-differences (from $\mathbb{F}_2^n$ to $\mathbb{F}_2^{dn}$) and the limitation of the diffusion to maximally a factor of 2^n (see Lemma 7), the ratio

of possible $(d+1)$-polytopic transitions to impossible $(d+1)$-polytopic transitions will be low for many more rounds than possible for standard differentials. In fact, by increasing the dimension d of the polytopic transition, it can be assured that the ratio of possible to impossible polytopic transitions is close to zero for an almost arbitrary number of rounds.

An impossible $(d+1)$-polytopic attack could then proceed as follows. Let n be the block size of the cipher and let l be the number of bits in the last round key.

1. Choose a d and a d-difference such that the ratio of possible to impossible $(d+1)$-polytopic transitions is lower than 2^{-l-1}.

2. Get the r-round encryption of $d+1$ plaintexts chosen such that they adhere to the input d-difference.

3. For each guess of the round key k_r decrypt the last round. If the obtained d-difference after the $(r-1)$th round is possible, keep the key as a candidate. Otherwise discard it.

Clearly this should leave us on average with one round key candidate which is bound to be the correct one. In practice, an attack would likely be more complex, e.g., with only partially guessed round keys and tradeoffs in the filtering probability and the data/time complexities.

While the data complexity is limited to $d+1$ chosen plaintexts (and thus very low), the time complexity is harder to determine and depends on the difficulty of determining whether an obtained $(r-1)$-round $(d+1)$-polytopic transition is possible or not. The straightforward approach is to precompute a list of possible d-differences after round $r-1$. Both the exponentially increasing memory requirements and the time of the precomputation limit this approach though. In spite of this, attacks using this approach are competitive with existing low-data attacks as we show in Section 4.

One possibility to reduce the memory complexity is to use a meet-in-the-middle approach where one searches for a collision in the possible d-differences reachable from the input d-difference and the calculated d-difference after round $(r-1)$ at a round somewhere in the middle of the cipher. This however requires to repeat the computation for every newly calculated d-difference and thus limits its use in the scenario where we calculated a new d-difference after round $(r-1)$ for each key guess (not in a distinguishing attack though).

Clearly any method that could efficiently determine the impossibility of most impossible polytopic transitions would prove extremely useful in an attack. Intuitively it might seem that it is generally a hard problem to determine the possibility of a polytopic transition. As a matter of fact though, there already exists a cryptographic technique that provides a very efficient distinguisher for certain types of polytopic transitions, namely higher-order differentials which are shown in Section B to correspond to truncated polytopic transitions. This raises the hope that better distinguishing techniques could still be discovered.

There is one important further effect of the increase in the size of the difference space: it allows us to restrict ourselves to impossible d-differences on only a part of

the state. It is even possible to restrict the d-difference to a d-difference in one bit and still use it for efficient filtering.[1] In Section 4 we will use these techniques in impossible polytopic attacks to demonstrate the validity of the attacks and provide a usage scenario.

Wrong keys and random permutations

Note that while impossible polytopic attacks – just like impossible differential attacks – do not require the stochastic equivalence hypothesis, practical attacks require another hypothesis: the wrong-key randomization hypothesis. This hypothesis states that when decrypting one or several rounds with a wrong key guess creates a function that behaves like a random function. For our setting, we formulate it is as following:

Wrong-key randomization hypothesis. *When decrypting one or multiple rounds of a block cipher with a wrong key guess, the resulting polytopic transition probability will be close to the transition probability over a random permutation for almost all key guesses.*

Let us therefore take a look at the polytopic transition probabilities over random functions and random permutation. To simplify the treatment, we make the following definition:

Definition 11 (Degenerate d-difference)**.** Let $\boldsymbol{\alpha}$ be a d-difference over $\mathbb{F}_2^n$: $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d)$. We call $\boldsymbol{\alpha}$ *degenerate* if there exists an i with $1 \leq i \leq d$ with $\alpha_i = 0$ or if there exists a pair i, j with $1 \leq i < j \leq d$ and $\alpha_i = \alpha_j$. Otherwise we call $\boldsymbol{\alpha}$ *non-degenerate.*

Clearly if and only if a d-difference $\boldsymbol{\alpha}$ is degenerate, there exist two texts in the underlying $(d+1)$-polytope that are identical. To understand the transition probability of a degenerate d-difference it is thus sufficient to evaluate the transition probability of a non-degenerate d'-difference ($d' < d$) that contains the same set of texts. For the following two propositions, we will thus restrict ourselves to non-degenerate d-differences.

Proposition 12 (Distribution over random function)**.** *Let $\boldsymbol{\alpha}$ be a non-degenerate d-difference over $\mathbb{F}_2^n$. Let $\mathbf{F}$ be a uniformly distributed random function from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$. The image of $\boldsymbol{\alpha}$ is then uniformly distributed over all d-difference over $\mathbb{F}_2^m$. In particular* $\Pr\left(\boldsymbol{\alpha} \xrightarrow{F} \boldsymbol{\beta}\right) = 2^{-md}$ *for any d-difference $\boldsymbol{\beta} \in (\mathbb{F}_2^m)^d$.*

Proof. Let $(m_0, m_1, \ldots, m_d)$ be a $(d+1)$-polytope that adheres to $\boldsymbol{\alpha}$. Then the polytope $(\mathbf{F}(m_0), \mathbf{F}(m_1), \ldots, \mathbf{F}(m_d))$ is clearly uniformly randomly distributed on $(\mathbb{F}_2^m)^{d+1}$ and accordingly $\boldsymbol{\beta}$ with $\boldsymbol{\alpha} \xrightarrow{\mathbf{F}} \boldsymbol{\beta}$ is distributed uniformly randomly on $(\mathbb{F}_2^m)^d$. □

[1] In standard differential cryptanalysis, this would require a probability 1 truncated differential.

For the image of a d-difference over a random permutation, we have a similar result:

Proposition 13 (Distribution over random permutation)**.** *Let $\boldsymbol{\alpha}$ be a non-degenerate d-difference over $\mathbb{F}_2^n$. Let $\mathbf{F}$ be a uniformly distributed random permutation on $\mathbb{F}_2^n$. The image of $\boldsymbol{\alpha}$ is then uniformly distributed over all non-degenerate d-difference over $\mathbb{F}_2^n$.*

Proof. Let $(m_0, m_1, \ldots, m_d)$ be a $(d+1)$-polytope that adheres to $\boldsymbol{\alpha}$. As $\boldsymbol{\alpha}$ is non-degenerate, all m_i are distinct. Thus the polytope $(\mathbf{F}(m_0), \mathbf{F}(m_1), \ldots, \mathbf{F}(m_d))$ is clearly uniformly randomly distributed on all $(d+1)$-polytopes in $(\mathbb{F}_2^m)^{d+1}$ with distinct values. Accordingly $\boldsymbol{\beta}$ with $\boldsymbol{\alpha} \xrightarrow{F} \boldsymbol{\beta}$ is distributed uniformly randomly on all non-degenerate d-differences over $\mathbb{F}_2^n$. □

As long as $d \ll 2^n$, we can thus well approximate the probability $\Pr\left(\boldsymbol{\alpha} \xrightarrow{\mathbf{F}} \boldsymbol{\beta}\right)$ by 2^{-dn} when $\boldsymbol{\beta}$ is non-degenerate.

In the following, these proposition will be useful when we try to estimate the probability that a partial decryption with a wrong key guess will still give us a possible intermediate d-difference. We will then always assume that the wrong-key randomization hypothesis holds and that the probability of getting a particular d-difference on m bits is the same as if we had used a random permutation, i.e., it is 2^{-dm} (as our d is always small).

4 Impossible polytopic attacks on DES and AES

Without much doubt are the Data Encryption Standard (DES) [26] and the Advanced Encryption Standard (AES) [15] the most studied and best cryptanalyzed block ciphers. Any cryptanalytic improvement on these ciphers should thus be a good indicator of the novelty and quality of a new cryptanalytic attack. We believe that these ciphers thus pose ideal candidates to demonstrate that the generalization of differential cryptanalysis to polytopic cryptanalysis is not a mere intellectual exercise but useful for practical cryptanalysis.

In the following, we demonstrate several impossible polytopic attacks on reduced-round versions of DES and AES that make only use of a very small set of chosen plaintexts. The natural reference frame for these attacks are hence low-data attacks. In Table 1 and in Table 2 we compare our attacks to other low-data attacks on round-reduced versions of DES and AES respectively. We should mention here that [11] only states attacks on 7 and 8 rounds of DES. It is not clear whether the techniques therein could also be used to improve complexities of meet-in-the-middle attacks for 5- and 6-round versions of that cipher.

We stress here that in contrast to at least some of the other low-data attacks, our attacks make no assumption on the key schedule and work equally well with independent round keys. In fact, all of our attacks are straight-forward applications

Table 1: Comparison table of low-data attacks on round-reduced DES. Data complexity is measured in number of required known plaintexts (KP) or chosen plaintexts (CP). Time complexity is measured in round-reduced DES encryption equivalents. Memory complexity is measured in plaintexts (8 bytes). For the other attacks no memory requirements were explicitly specified in the publications. They should be low though. The attacks of this paper are in bold.

Rounds	Attack Type	Time	Data	Memory	Source
5	Differential	$> 2^{11.7}$	64 CP	-	As in [18]
	Linear	$> 2^{13.8}$	72 CP	-	As in [18]
	MitM	$2^{45.5}$	1 KP	-	From [13]
	MitM	$2^{37.9}$	28 KP	-	From [18]
	MitM	2^{30}	8 CP	-	From [18]
	Imp. polytopic	$\mathbf{2^{13.2}}$	**4 CP**	$\mathbf{2^{9}}$	**This paper**
6	Differential	$2^{13.7}$	256 CP	-	As in [18]
	Linear	$2^{13.9}$	>104 KP	-	As in [18]
	MitM	$2^{51.8}$	1 KP	-	From [18]
	Truncated diff.	2^{48}	7 CP	-	From [20]
	Truncated diff.	$2^{11.8}$	46 CP	-	From [20]
	Imp. polytopic	$\mathbf{2^{32.2}}$	**4 CP**	$\mathbf{2^{10}}$	**This paper**
	Imp. polytopic	$\mathbf{2^{18.4}}$	**48 CP**	$\mathbf{2^{9}}$	**This paper**
7	MitM Sieve	2^{53}	1 KP	-	From [11]
	Imp. polytopic	$\mathbf{2^{43}}$	**16 CP**	$\mathbf{2^{43}}$	**This paper**
	Imp. polytopic	$\mathbf{2^{37.8}}$	**48 CP**	$\mathbf{2^{10}}$	**This paper**
8	MitM Sieve	2^{53}	16 KP	-	From [11]

of the ideas developed in this paper. There is likely still room for improvement of these attacks using details of the ciphers and more finely controlled trade-offs.

In all of the following attacks, we determine the possibility or impossibility of a polytopic transition by deterministically generating a list of all d-differences that are reachable from the starting d-difference, i.e., we generate and keep a list of all possible d-differences. The determination of these lists is straightforward using the rules described in Section 2. The sizes of these lists are the limiting factors of the attacks both for the time and the memory complexities.

4.1 Attacks on the DES

For a good reference for the DES, we refer to [21]. A summary of the results for DES can be found in Table 1.

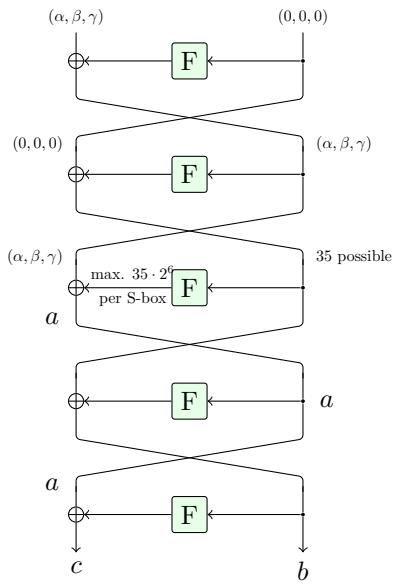

Figure 1: Outline of the 5-round attack on DES.

A 5-round attack.

Let us start with an impossible 4-polytopic attack on 5-round DES. We split our input 3-difference into two parts, one for the left 32 state bits and one for the right 32 state bits. Let us denote the left 3-difference as (α, β, γ). For the right half we choose the 3-difference $(0, 0, 0)$. This allows us to pass the first round for free (as can be seen in Fig. 1).

The number of possible 3-differences after the second round depends now on our choice of α, β, and γ. To keep this number low, clearly it is good to choose the differences to activate as few S-boxes as possible. We experimentally tried out different natural choices and chose the values

$$(\alpha, \beta, \gamma) = (\mathtt{02000000}, \mathtt{04000000}, \mathtt{06000000}).$$

All of these three differences only activate S-box 2 in round 2. With this choice we get 35 possible 3-differences after round 2. Note that the left 3-difference is still (α, β, γ) after round 2 while the 35 variations only appear in the right half.

As discussed earlier, the maximal number of output d-differences for a fixed input d-difference is inherently limited by the size of the domain of the function. A consequence of this is that for any of the 35 3-differences after round 2 the possible number of output 3-differences of any S-box in round 3 is limited to 2^6 as shown in Fig. 1. But by guessing the 6 bits of round key 5 that go into the corresponding S-box in round 5, we can determine the 3-difference in the same four output bits of

round 3 now coming from the ciphertexts. For the right guess of the 6 key bits, the determined 3-difference will be possible. For a wrong key guess though, we expect the 3-difference to take a random value in the set of all 3-differences on 4 bits.

But the size of the space of 3-differences in these four output bits is now $2^{4\cdot 3} = 2^{12}$. Thus when fixing one of the 35 possible 3-differences after round 2, we expect on average to get one suggestion for the 6 key bits in that S-box. Repeating this for every S-box, we get on average one suggestion for the last round key for each of the 35 possible 3-differences after round 2, leaving us with an average of 35 key candidates for the last round key.

What are the complexities of the attack? Clearly we only need 4 chosen plaintexts. For the time complexity we get the following: For each of the 35 possible 3-differences after round 2, we have to determine the 2^6 possible output 3-differences and for each of these, we have to see in the list of possible 3-differences obtained from the key guesses whether there is a guess of the 6 key bits that gives us exactly that 3-difference. Thus we have a total of $35 \cdot 8 \cdot 2^6 = 2^{14.2}$ steps each of which should be easier than one round of DES encryption. This leaves us with a time complexity of $\approx 2^{12}$ 5-round DES encryptions equivalents. But to completely determine the DES key we need 8 additional bits that are not present in the last round key. As we expect on average maximally 35 round keys, we are left with trying out the $35 \cdot 2^8 = 2^{13.2}$ full key candidates, setting the time complexity of this attack to that value.

The only memory requirement in this attack is storing the list of possible 3-differences for each key guess in each S-box. This should roughly be no more than 2^{12} bytes.

A 6-round attack.

The 6-round attack proceeds exactly as the 5-round attack, with the only difference being that instead of determining the possible 3-difference output of each S-box in round 3, we do the same in round 4 and thus have to repeat the attack for every possible 3-difference after round 3.

Experimental testing revealed that it is beneficial for this attack to choose a different choice of α, β, and γ, namely

$$(\alpha, \beta, \gamma) = (\mathsf{20000000}, \mathsf{40000000}, \mathsf{60000000}),$$

which now activates S-box 1 instead of S-box 2 as it gives us the lowest number of 3-differences after round 3. For this choice, we get a number of 48 possible 3-differences after round 2 and $2^{24.12}$ possible 3-differences after round 3. Now substituting 35 with this number in the previous attack, gives us the time complexity for this 6-round attack.

A note regarding the memory requirement of this attack: As we loop over the $2^{24.12}$ possible 3-differences after round 3, we are not required to store all of them at any time. By doing the attack while creating these possible 3-differences we can keep the memory complexity nearly as low as before, namely to roughly 2^{13} bytes.

A 7-round attack.

Unfortunately extending from 6 to 7 rounds as done when going from 5 to 6 rounds is not possible, due to the prohibitively large number of possible 3-differences after round 4. Instead we use a different angle.

It is well known that when attacking r-round DES, guessing the appropriate 36 round key bits of the last round key and the appropriate 6 bits of the round key in round $r-1$ allows us to determine the output state bits of an S-box of our choice after round $r-3$. We will thus restrict ourselves to looking for impossible d-differences in only one S-box. We choose S-box 1 here.

In order to have a sufficiently high success rate, we need to increase the dimension of our polytopic transitions to increase the size of the d-difference space of the four output bits of the S-box of our choice. For this attack we choose $d = 15$ giving us a 15-difference space size of 2^{60} in four bits.

For our choice of input 15-difference, we again leave all differences in the right side to 0, while choosing for the 15-difference on the left side:

$$\begin{aligned}(&\mathtt{00000002}, \mathtt{00000004}, \mathtt{00000006}, \mathtt{02000000}, \mathtt{02000002}, \mathtt{02000004},\\ &\mathtt{02000006}, \mathtt{04000000}, \mathtt{04000002}, \mathtt{04000004}, \mathtt{04000006}, \mathtt{06000000},\\ &\mathtt{06000002}, \mathtt{06000004}, \mathtt{06000006})\end{aligned}$$

which only activates S-boxes 2 and 8. For this choice of input 15-difference we get 1470 possible 15-differences after round 2 and $2^{36.43}$ possible 15-differences after round 3.

For each of these $2^{36.43}$ possible 15-differences after round 3, we calculate the 2^6 possible output 15-differences of S-box 1 after round 4. Now having precomputed a list of possible 15-differences in the output bits of S-box 1 after round 4 for each of the 2^{42} guessed key bits of round 7 and 6, we can easily test whether we get a collision. What is the probability of this? The 15-difference space size in the four bits is 2^{60} and, we get maximally 2^{42} possible 15-differences from the key guesses. This leaves us with a chance of 2^{-18} that we find a 15-difference in that list. Thus for each of the $2^{36.43}$ possible 15-differences after round 3, we expect on average at most 2^{-12} suggestions for the guessed 42 key bits, a total of $2^{24.43}$ suggestions.

What are the complexities for this attack? Clearly again, the data complexity is 16 chosen plaintexts. For the time complexity, for each of the $2^{42.42}$ possible 4-bit 15-differences obtained after round 4, we have to see whether it is contained in the list of 2^{42} 3-differences which we obtained from the key guesses. To do this efficiently, we first have to sort the list which should take $2^{42} \cdot 42 = 2^{47.4}$ elementary steps. Assuming that a 7-round DES encryption takes at least 42 elementary steps, we can upperbound this complexity with 2^{42} DES encryption equivalents. As finding an entry in a list of 2^{42} entries also takes approximately 42 elementary steps, this leaves us with a total time complexity of at most 2^{43} 7-round DES encryption equivalents. As each suggestion gives us 42 DES key bits and as the list of suggestions has a size of $2^{24.23}$, we can find the correct full key with $2^{38.23}$ 7-round DES trial encryptions

which is lower than then the previously mentioned time complexity and can thus be disregarded.

The data complexity is determined by the size of the list of 4-bit 15-differences generated from the key guesses. This gives us a memory requirement of $2^{42}(15 \cdot 4 + 42)$ bits $\approx 2^{46}$ bytes.

Extension of the attacks using more plaintexts.

The attacks for 5 and 6 rounds can be extended by one round at the cost of a higher data complexity. The extension can be made at the beginning of the cipher in the following way.

Let us suppose we start with a 3-difference $(\delta_1, \delta_2, \delta_3)$ in the left half and the 3-difference (α, β, γ) in the right half. If we knew the output 3-difference of the round function in the first round, we could choose $(\delta_1, \delta_2, \delta_3)$ accordingly to make sure that we end up at the starting position of the original attack. Thus by guessing this value and repeating the attack for each guessed value of this 3-difference we can make sure we still retrieve the key.

Fortunately the values of (α, β, γ) are already chosen to give a minimal number of possible 3-difference in the round function. Thus the time complexity only increases by this value, i.e., 35 and 48. The data complexity increases even less. As it turns out, 12 different values for the left half of the input text are enough to generate all of the 35 resp. 48 3-differences. Thus the data complexity only increases to 48 chosen plaintexts.

We should mention that the same technique can be used to extend the 7-round attack to an 8-round attack. But this leaves us with the same time complexity as the 8-round attack in [11], albeit at a much higher data cost.

Experimental results.

To verify the correctness of the above attacks and their complexities, we implemented the 5-round and 6-round attacks that use 4 chosen plaintexts. We ran the attacks on a single core of an Intel Core i5-4300U processor. We ran the 5-round attack 100000 times which took about 140 seconds. The average number of suggested round keys was 47 which is slightly higher than the expected number of 35. The suggested number of round keys was below 35 though in 84 percent of the cases and below 100 in 95 percent of the cases. In fact, the raised average is created by a few outliers in the distribution: taking the average on all but the 0.02 percent worst cases, we get 33.1 round key suggestions per case. While this shows that the estimated probability is generally good, it also demonstrates that the wrong-key randomization hypothesis has to be handled with care.

Running the six-round attack 10 times, an attack ran an average time of 10 min and produced an average of $2^{22.3}$ candidates for the last round key. As expected, the correct round key was always in the list of candidate round keys for both the 5-round and 6-round attacks.

Table 2: Comparison table of low-data attacks on round-reduced AES. Data complexity is measured in number of required chosen plaintexts (CP). Time complexity is measured in round-reduced AES encryption equivalents. Memory complexity is measured in plaintexts (16 bytes). The column 'keyschedule' denotes whether the attacks use the AES key schedule. All attacks that rely on the keyschedule are attacks on AES-128. The attacks of this paper are in bold.

Rounds	Attack Type	Time	Data	Memory	Keyschedule	Source
4	Guess & Det.	2^{120}	1 KP	2^{120}	Yes	As in [10]
	Diff. MitM	2^{104}	3 CP	1	Yes	As in [8], [9]
	Guess & Det.	2^{80}	2 CP	2^{80}	Yes	As in [10]
	Guess & Det.	2^{32}	4 CP	2^{24}	Yes	As in [10]
	Imp. polyt.	$\mathbf{2^{38}}$	**8 CP**	$\mathbf{2^{15}}$	**No**	**This paper**
5	MitM	2^{64}	8 CP	2^{56}	Yes	As in [17], Sec. 7.5.1
	Imp. polyt.	$\mathbf{2^{70}}$	**15 CP**	$\mathbf{2^{41}}$	**No**	**This paper**

4.2 Attacks on the AES

For a good reference for the AES, we refer to [15]. A summary of the results for AES can be found in Table 2.

A 4-round attack.

We first present here an impossible 8-polytopic attack on 4-round AES. For the input 7-difference, we choose a 7-difference that activates only the first byte, i.e., that is all-zero in all other bytes. Such a 7-difference can be mapped after round 1 to at most 2^8 different 7-differences. If we restrict ourselves to the 7-differences in the first column after round 2, we can then at most have 2^{16} different 7-differences in this column. In particular, we can have at most have 2^{16} different 7-differences in the first byte. For a depiction of this, see Fig. 2.

If we now request the encryptions of 8 plaintexts that adhere to our chosen start 7-difference, we can now determine the corresponding 7-difference after round 2 in the first byte by guessing 40 round key bits of round keys 3 and 4. If this 7-difference does not belong to the set of 2^{16} possible ones, we can discard the key guess as wrong.

How many guesses of the 40 key bits, do we expect to survive the filtering? There are 2^{56} possible 7-difference on a byte and only 2^{16} possible ones coming from our chosen input 7-difference. This leaves a chance of 2^{-40} for a wrong key guess to produce a correct 7-difference. We thus expect on average 2 suggestions for the 40 key bits, among them the right one. To determine the remaining round key bits, we can use the same texts, only restricting ourselves to different columns.

The data complexity of the attack is limited to 8 chosen plaintexts. The time complexity is dominated by determining the 7-difference in the first byte after round

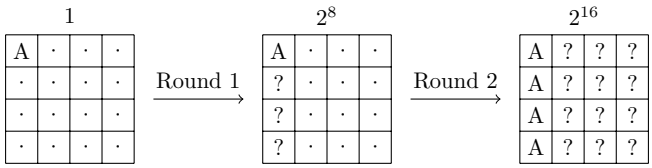

Figure 2: Diffusion of the starting 7-difference for the 4-round attack on AES. The letter A shows a byte position in which a possible 7-difference is non-zero and known. A dot indicates a byte position where the 7-difference is known to be zero. A question mark indicates a byte position where arbitrary values for the 7-differences are allowed. In total there are 2^{16} different 7-differences possible in the first column after the second round.

2 for each guess of the 40 key bits and checking whether it is among the 2^{16} possible ones. This can be done in less than 16 table lookups on average for each key guess. Thus the time complexity corresponds to $2^{40} \cdot 2^{-2} = 2^{38}$ 4-round AES encryption equivalents, assuming one 4-round encryption corresponds to $4 \cdot 16$ table lookups. The memory complexity is limited to a table of the 2^{16} allowed 7-difference in one byte, corresponding to 2^{19} bytes or 2^{15} plaintext equivalents.

A 5-round attack.

In this attack, we are working with 15-polytopes and trace the possible 14-differences one round further than in the 4-round attack. Again we choose our starting 14-difference such that it only activates the first byte. After two rounds we then have maximally 2^{40} different 14-differences on the whole state. If we restrict ourselves to only the first column of the state after round 3, we then get an additional 2^{32} possible 14-differences in this column for each of the 2^{40} possible 14-differences after round 2, resulting in a total of 2^{72} possible 14-differences in the first column after round 3. This is depicted in Fig. 3. In particular again, we can have at most have 2^{72} different 14-differences in the first byte.

Let us suppose now we have the encrypted values of a 15-polytope that adheres to our starting 14-difference. We can then again find the respective 14-difference in the first byte after the third round by guessing 40 key bits in round keys 4 and 5. There are in total 2^{112} different 14-differences in one byte. The chance of a wrong key guess to produce one of the possible 2^{72} 14-differences is thus 2^{-40}. We thus expect on average 2 suggestions for the 40 key bits, among them the right one. To determine the remaining round key bits, we can again use the same texts but restricting ourselves to a different column.

To lower the memory complexity of this attack it is advantageous to not store the 2^{72} possible 14-differences but store for each of the 2^{40} key guesses the obtained 14-difference. This gives a memory complexity of $2^{40} \cdot (14 + 5)$ bytes corresponding to 2^{41} plaintext equivalents. The time complexity is then dominated by constructing

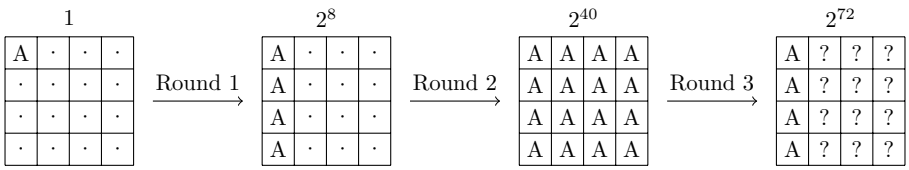

Figure 3: Diffusion of the starting 14-difference for the 5-round attack on AES. The letter A shows a byte position in which a possible 14-difference is non-zero and known. A dot indicates a byte position where the 14-difference is known to be zero. A question mark indicates a byte position where arbitrary values for the 14-differences are allowed. In total there are 2^{72} different 14-differences possible in the first column after the third round.

the 2^{72} possible 14-differences and testing whether they correspond to one of the key guesses. This should not take more than the equivalent of $2^{72} \cdot 16$ table lookups resulting in a time complexity of 2^{70} 5-round AES encryption equivalents. The data complexity is restricted to the 15 chosen plaintexts needed to construct one 15-polytope corresponding to the starting 14-difference.

5 Conclusion

In this paper, we developed and studied polytopic cryptanalysis. We were able to show that the methodology and notation of standard cryptanalysis can be unambiguously extended to polytopic cryptanalysis, including the concept of impossible differentials. Standard differential cryptanalysis remains as a special case of polytopic cryptanalysis.

For impossible polytopic transitions, we demonstrated that both the increase in the size of the space of d-differences and the inherent limit in the diffusion of d-differences in a cipher allow them to be very effective in settings where ordinary impossible differentials fail. This is the case when the number of rounds is so high that impossible differentials do no longer exist or when the allowed data complexity is too low.

Finally we showed the practical relevance of this framework by demonstrating novel low-data attacks on DES and AES that are able to compete with existing attacks.

Acknowledgements

The author thanks Christian Rechberger, Stefan Kölbl, and Martin M. Lauridsen for fruitful discussions. The author also thanks Dmitry Khovratovich and the anonymous reviewers for comments that helped to considerably improve the quality of the paper.

References

[1] Eli Biham, Alex Biryukov, and Adi Shamir. "Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials". In: *J. Cryptology* 18.4 (2005), pp. 291–311.

[2] Eli Biham and Adi Shamir. "Differential Cryptanalysis of DES-like Cryptosystems". In: *J. Cryptology* 4.1 (1991), pp. 3–72.

[3] Alex Biryukov and Adi Shamir. "Structural Cryptanalysis of SASAS". In: *J. Cryptology* 23.4 (2010), pp. 505–518.

[4] Céline Blondeau and Benoît Gérard. "Multiple Differential Cryptanalysis: Theory and Practice". In: *Fast Software Encryption, FSE 2011*. Ed. by Antoine Joux. Vol. 6733. Lecture Notes in Computer Science. Springer, 2011, pp. 35–54. ISBN: 978-3-642-21701-2.

[5] Céline Blondeau, Gregor Leander, and Kaisa Nyberg. "Differential-Linear Cryptanalysis Revisited". In: *Fast Software Encryption, FSE 2014*. Ed. by Carlos Cid and Christian Rechberger. Vol. 8540. Lecture Notes in Computer Science. Springer, 2015, pp. 411–430. ISBN: 978-3-662-46705-3.

[6] Céline Blondeau and Kaisa Nyberg. "Links between Truncated Differential and Multidimensional Linear Properties of Block Ciphers and Underlying Attack Complexities". In: *Advances in Cryptology - EUROCRYPT 2014*. Ed. by Phong Q. Nguyen and Elisabeth Oswald. Vol. 8441. Lecture Notes in Computer Science. Springer, 2014, pp. 165–182. ISBN: 978-3-642-55219-9.

[7] 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: *Cryptographic Hardware and Embedded Systems - CHES 2007*. Ed. by Pascal Paillier and Ingrid Verbauwhede. Vol. 4727. Lecture Notes in Computer Science. Springer, 2007, pp. 450–466. ISBN: 978-3-540-74734-5.

[8] Charles Bouillaguet, Patrick Derbez, Orr Dunkelman, Pierre-Alain Fouque, Nathan Keller, and Vincent Rijmen. "Low-Data Complexity Attacks on AES". In: *IEEE Transactions on Information Theory* 58.11 (2012), pp. 7002–7017.

[9] Charles Bouillaguet, Patrick Derbez, Orr Dunkelman, Nathan Keller, Vincent Rijmen, and Pierre-Alain Fouque. *Low Data Complexity Attacks on AES*. Cryptology ePrint Archive, Report 2010/633. `http://eprint.iacr.org/`. 2010.

[10] Charles Bouillaguet, Patrick Derbez, and Pierre-Alain Fouque. "Automatic Search of Attacks on Round-Reduced AES and Applications". In: *Advances in Cryptology - CRYPTO 2011*. Ed. by Phillip Rogaway. Vol. 6841. Lecture Notes in Computer Science. Springer, 2011, pp. 169–187. ISBN: 978-3-642-22791-2.

[11] Anne Canteaut, María Naya-Plasencia, and Bastien Vayssière. "Sieve-in-the-Middle: Improved MITM Attacks". In: *Advances in Cryptology - CRYPTO 2013*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8042. Lecture Notes in Computer Science. Springer, 2013, pp. 222–240. ISBN: 978-3-642-40040-7.

[12] Florent Chabaud and Serge Vaudenay. "Links Between Differential and Linear Cryptanalysis". In: *Advances in Cryptology - EUROCRYPT '94*. Ed. by Alfredo De Santis. Vol. 950. Lecture Notes in Computer Science. Springer, 1995, pp. 356–365. ISBN: 3-540-60176-7.

[13] David Chaum and Jan-Hendrik Evertse. "Crytanalysis of DES with a Reduced Number of Rounds: Sequences of Linear Factors in Block Ciphers". In: *Advances in Cryptology - CRYPTO '85*. Ed. by Hugh C. Williams. Vol. 218. Lecture Notes in Computer Science. Springer, 1986, pp. 192–211. ISBN: 3-540-16463-4.

[14] Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. "The Block Cipher Square". In: *Fast Software Encryption, FSE '97*. Ed. by Eli Biham. Vol. 1267. Lecture Notes in Computer Science. Springer, 1997, pp. 149–165. ISBN: 3-540-63247-6.

[15] 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.

[16] Christophe De Cannière, Orr Dunkelman, and Miroslav Knezevic. "KATAN and KTANTAN - A Family of Small and Efficient Hardware-Oriented Block Ciphers". In: *Cryptographic Hardware and Embedded Systems - CHES 2009,* ed. by Christophe Clavier and Kris Gaj. Vol. 5747. Lecture Notes in Computer Science. Springer, 2009, pp. 272–288. ISBN: 978-3-642-04137-2.

[17] Patrick Derbez. "Meet-in-the-Middle Attacks on AES". PhD thesis. Ecole Normale Supérieure de Paris - ENS Paris, Dec. 2013. URL: `https://tel.archives-ouvertes.fr/tel-00918146`.

[18] Orr Dunkelman, Gautham Sekar, and Bart Preneel. "Improved Meet-in-the-Middle Attacks on Reduced-Round DES". In: *Progress in Cryptology - INDOCRYPT 2007*. Ed. by K. Srinathan, C. Pandu Rangan, and Moti Yung. Vol. 4859. Lecture Notes in Computer Science. Springer, 2007, pp. 86–100. ISBN: 978-3-540-77025-1.

[19] Lars R. Knudsen. *DEAL - A 128-bit Block Cipher*. Technical Report 151. Submitted as an AES candidate by Richard Outerbridge. Department of Informatics, University of Bergen, Norway, Feb. 1998.

[20] Lars R. Knudsen. "Truncated and Higher Order Differentials". In: *Fast Software Encryption, FSE '94*. Ed. by Bart Preneel. Vol. 1008. Lecture Notes in Computer Science. Springer, 1995, pp. 196–211.

[21] Lars R. Knudsen and Matthew Robshaw. *The Block Cipher Companion*. Information Security and Cryptography. Springer, 2011. ISBN: 978-3-642-17341-7.

[22] Xuejia Lai. "Higher Order Derivatives and Differential Cryptanalysis". In: *Communications and Cryptography, Two Sides of One Tapestry*. Ed. by Richard E. Blahut, Jr. Daniel J. Costello, Ueli Maurer, and Thomas Mittelholzer. Kluwer Academic Publishers, 1994, pp. 227–233. ISBN: 978-1-4613-6159-6.

[23] Xuejia Lai and James L. Massey. "Markov Ciphers and Differential Cryptanalysis". In: *Advances in Cryptology - EUROCRYPT '91*. Ed. by Donald W. Davies. Vol. 547. Lecture Notes in Computer Science. Springer, 1991, pp. 17–38. ISBN: 3-540-54620-0.

[24] Susan K. Langford and Martin E. Hellman. "Differential-Linear Cryptanalysis". In: *Advances in Cryptology - CRYPTO '94*. Ed. by Yvo Desmedt. Vol. 839. Lecture Notes in Computer Science. Springer, 1994, pp. 17–25. ISBN: 3-540-58333-5.

[25] Sean Murphy. "The Return of the Cryptographic Boomerang". In: *IEEE Transactions on Information Theory* 57.4 (2011), pp. 2517–2521.

[26] National Institute of Standards and Technology. *Data Encryption Standard*. Federal Information Processing Standard (FIPS), Publication 46. U.S. Department of Commerce, Washington D.C., Jan. 1977.

[27] Taizo Shirai, Kyoji Shibutani, Toru Akishita, Shiho Moriai, and Tetsu Iwata. "The 128-Bit Blockcipher CLEFIA (Extended Abstract)". In: *Fast Software Encryption, FSE 2007*. Ed. by Alex Biryukov. Vol. 4593. Lecture Notes in Computer Science. Springer, 2007, pp. 181–195. ISBN: 978-3-540-74617-1.

[28] Serge Vaudenay. "Decorrelation: A Theory for Block Cipher Security". In: *J. Cryptology* 16.4 (2003), pp. 249–286.

[29] David Wagner. "The Boomerang Attack". In: *Fast Software Encryption, FSE '99*. Ed. by Lars R. Knudsen. Vol. 1636. Lecture Notes in Computer Science. Springer, 1999, pp. 156–170. ISBN: 3-540-66226-X.

[30] David Wagner. "Towards a Unifying View of Block Cipher Cryptanalysis". In: *Fast Software Encryption, FSE 2004*. Ed. by Bimal K. Roy and Willi Meier. Vol. 3017. Lecture Notes in Computer Science. Springer, 2004, pp. 16–33. ISBN: 3-540-22171-9.

A Markov model in polytopic cryptanalysis

To develop the Markov model, we first need to introduce keys in the function over which the transitions take place. We will thus restrict our discussion to product ciphers i.e., block ciphers that are constructed through repeated composition of round functions. In contrast to Eq. (2), each round function f^i is now keyed with its own round key k_i which itself is derived from the key k of the cipher via a key schedule.[2]

[2] For a clearer notation, we moved the index from subscript to superscript.

We can then write the block cipher f_k as:

$$f_k := f^r_{k_r} \circ \cdots \circ f^2_{k_2} \circ f^1_{k_1}. \tag{1}$$

The first assumption that we now need to make, is that the round keys are independent. The second assumption is that the product cipher is a Markov cipher. Here we adopt the notion of a Markov cipher from [23] to polytopic cryptanalysis:

Definition 14. A product cipher is a $(d+1)$-polytopic Markov cipher if and only if for all round functions f^i, for any $(d+1)$-polytopic transition $\boldsymbol{\alpha} \longrightarrow \boldsymbol{\beta}$ for that round function and any fixed inputs $x, y \in \mathbb{F}_2^n$, we have

$$\Pr_{\mathbf{K}}\left(\boldsymbol{\alpha} \xrightarrow[x]{f^i_{\mathbf{K}}} \boldsymbol{\beta}\right) = \Pr_{\mathbf{K}}\left(\boldsymbol{\alpha} \xrightarrow[y]{f^i_{\mathbf{K}}} \boldsymbol{\beta}\right) \tag{2}$$

where $\mathbf{K}$ is a random variable distributed uniformly over the spaces of round keys.

In words, a cipher is a $(d+1)$-polytopic Markov cipher if and only if the probabilities of 1-round $(d+1)$-polytopic transitions do not depend on the specific anchor as long as the round key is distributed uniformly at random. For $d = 1$, the definition coincides with the classical definition.

Just as with the standard definition of Markov ciphers, most block ciphers are $(d+1)$-polytopic Markov ciphers for any d as the round keys are usually added to any part of the state that enters the non-linear part of the round function (for a counterexample, see [16]). Examples of $(d+1)$-polytopic Markov ciphers are SPN ciphers such as AES [15] or PRESENT [7], and Feistel ciphers such as DES [26] or CLEFIA [27]. We are not aware of any cipher that is Markov in the classical definition but not $(d+1)$-polytopic Markov.

In the following, we extend the central theorem from [23] (Theorem 2) to the case of $(d+1)$-polytopes.

Theorem 15. *Let $f_k = f^r_{k_r} \circ \cdots \circ f^1_{k_1}$ be a $(d+1)$-polytopic Markov cipher with independent round keys, chosen uniformly at random and let $\boldsymbol{\delta}_0, \boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_r$ be a series of d-differences such that $\boldsymbol{\delta}_0$ is the input d-difference of round 1 and $\boldsymbol{\delta}_i$ is the output d-difference of round i of some fixed input $(d+1)$-polytope. The series $\boldsymbol{\delta}_0, \boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_r$ then forms a Markov chain.*

The following proof follows the lines of the original proof from [23].

Proof. We limit ourselves here to showing that

$$\Pr_{\mathbf{K}_1,\mathbf{K}_2}\left(\boldsymbol{\delta}_1 \xrightarrow[f^1_{\mathbf{K}_1}(x)]{f^2_{\mathbf{K}_2}} \boldsymbol{\delta}_2 \;\middle|\; \boldsymbol{\delta}_0 \xrightarrow[x]{f^1_{\mathbf{K}_1}} \boldsymbol{\delta}_1\right) = \Pr_{\mathbf{K}_2}\left(\boldsymbol{\delta}_1 \xrightarrow[z]{f^2_{\mathbf{K}_2}} \boldsymbol{\delta}_2\right) \tag{3}$$

where x and z are any elements from $\mathbb{F}_2^n$ and $\mathbf{K}_1$ and $\mathbf{K}_2$ are distributed uniformly at random over their respective round key spaces and the conditioned event has positive

probability. The theorem then follows easily by induction and application of the same arguments to the other rounds.

For any $x, z \in \mathbb{F}_2^n$, we now have

$$\begin{aligned}
&\Pr_{\mathbf{K}_1,\mathbf{K}_2}\left(\boldsymbol{\delta}_1 \xrightarrow[f^1_{\mathbf{K}_1}(x)]{f^2_{\mathbf{K}_2}} \boldsymbol{\delta}_2 \text{ and } \boldsymbol{\delta}_0 \xrightarrow[x]{f^1_{\mathbf{K}_1}} \boldsymbol{\delta}_1\right) \\
&= \sum_{y\in\mathbb{F}_2^n} \Pr_{\mathbf{K}_1,\mathbf{K}_2}\left(\boldsymbol{\delta}_1 \xrightarrow[y]{f^2_{\mathbf{K}_2}} \boldsymbol{\delta}_2 \text{ and } \boldsymbol{\delta}_0 \xrightarrow[x]{f^1_{\mathbf{K}_1}} \boldsymbol{\delta}_1 \text{ and } f^1_{\mathbf{K}_1}(x) = y\right) \\
&= \sum_{y\in\mathbb{F}_2^n} \Pr_{\mathbf{K}_2}\left(\boldsymbol{\delta}_1 \xrightarrow[y]{f^2_{\mathbf{K}_2}} \boldsymbol{\delta}_2\right) \cdot \Pr_{\mathbf{K}_1}\left(\boldsymbol{\delta}_0 \xrightarrow[x]{f^1_{\mathbf{K}_1}} \boldsymbol{\delta}_1 \text{ and } f^1_{\mathbf{K}_1}(x) = y\right) \\
&= \Pr_{\mathbf{K}_2}\left(\boldsymbol{\delta}_1 \xrightarrow[z]{f^2_{\mathbf{K}_2}} \boldsymbol{\delta}_2\right) \cdot \sum_{y\in\mathbb{F}_2^n} \Pr_{\mathbf{K}_1}\left(\boldsymbol{\delta}_0 \xrightarrow[x]{f^1_{\mathbf{K}_1}} \boldsymbol{\delta}_1 \text{ and } f^1_{\mathbf{K}_1}(x) = y\right) \\
&= \Pr_{\mathbf{K}_2}\left(\boldsymbol{\delta}_1 \xrightarrow[z]{f^2_{\mathbf{K}_2}} \boldsymbol{\delta}_2\right) \cdot \Pr_{\mathbf{K}_1}\left(\boldsymbol{\delta}_0 \xrightarrow[x]{f^1_{\mathbf{K}_1}} \boldsymbol{\delta}_1\right)
\end{aligned}$$

where the second equality comes from the independence of keys K_1 and K_2 and the third equality comes from the Markov property of the cipher. From this, Eq. (3) follows directly. □

The important consequence of the fact that the sequence of d-differences forms a Markov chain is that, just as in standard differential cryptanalysis, the average probability of a particular polytopic trail with respect to independent random round keys is the product of the single polytopic 1-round transitions of which it consists. We then have the following result:

Corollary 16. *Let f_k, $f^i_{k_i}, 1 \le i \le r$ be as before. Let $\boldsymbol{\alpha}_0 \xrightarrow{f_1} \boldsymbol{\alpha}_1 \xrightarrow{f_2} \cdots \xrightarrow{f_r} \boldsymbol{\alpha}_r$ be an r-round $(d+1)$-polytopic trail. Then*

$$\Pr\left(\boldsymbol{\alpha}_0 \xrightarrow{f^1_{\mathbf{K}_1}} \boldsymbol{\alpha}_1 \xrightarrow{f^2_{\mathbf{K}_2}} \cdots \xrightarrow{f^r_{\mathbf{K}_r}} \boldsymbol{\alpha}_r\right) = \prod_{i=1}^{r} \Pr\left(\boldsymbol{\alpha}_{i-1} \xrightarrow{f^i_{\mathbf{K}_i}} \boldsymbol{\alpha}_i\right) \tag{4}$$

where $x \in \mathbb{F}_2^n$ and the $\mathbf{K}_i$ are uniformly randomly distributed on their respective spaces.

Proof. This is a direct consequence of the fact that d-differences form a Markov chain. □

In most attacks though, we are attacking one fixed key and can not average the attack over all keys. Thus the following assumption is necessary:

Hypothesis of stochastic equivalence. *Let f be as above. The hypothesis of stochastic equivalence then refers to the assumption that the probability of any polytopic*

trail $\boldsymbol{\alpha}_0 \xrightarrow{f_1} \boldsymbol{\alpha}_1 \xrightarrow{f_2} \cdots \xrightarrow{f_r} \boldsymbol{\alpha}_r$ *is roughly the same for the large majority of keys:*

$$\Pr\left(\boldsymbol{\alpha}_0 \xrightarrow{f^1_{\mathbf{K}_1}} \boldsymbol{\alpha}_1 \xrightarrow{f^2_{\mathbf{K}_2}} \cdots \xrightarrow{f^r_{\mathbf{K}_r}} \boldsymbol{\alpha}_r\right) \approx \Pr\left(\boldsymbol{\alpha}_0 \xrightarrow{f^1_{k_1}} \boldsymbol{\alpha}_1 \xrightarrow{f^2_{k_2}} \cdots \xrightarrow{f^r_{k_r}} \boldsymbol{\alpha}_r\right) \tag{5}$$

for almost all tuples of round keys $(k_1, k_2, \ldots, k_r)$.

B Truncated polytopic transitions and higher-order differentials

In this section, we extend the definition of truncated differentials to polytopic transitions and prove that higher-order differentials are a special case of these. We then gauge the cryptographic ramifications of this.

In accordance with usual definitions for standard truncated differentials (see for example [6], we define:

Definition 17. A truncated d-difference is an affine subspace of the space of d-differences. A truncated $(d+1)$-polytopic transition is a pair (A, B) of truncated d-differences, mostly denoted as $A \xrightarrow{f} B$. The probability of a truncated $(d+1)$-polytopic transition (A, B) is defined as the probability that an input d-difference chosen uniformly randomly from A maps to a d-difference in B:

$$\Pr\left(A \xrightarrow{f} B\right) := |A|^{-1} \sum_{\substack{\boldsymbol{\alpha} \in A \\ \boldsymbol{\beta} \in B}} \Pr\left(\boldsymbol{\alpha} \xrightarrow{f} \boldsymbol{\beta}\right) \tag{1}$$

As the truncated input d-difference is usually just a single d-difference, the probability of a truncated differential is then just the probability that this input d-difference maps to any of the output d-differences in the output truncated d-difference. With a slight abuse of notation, we will denote the truncated polytopic transition then also as $\boldsymbol{\alpha} \xrightarrow{f} B$ where $\boldsymbol{\alpha}$ is the single d-difference of the input truncated d-difference.

A particular case of a truncated d-difference is the case where the individual differences of the d-differences always add up to the same value. This is in fact just the kind of d-differences one is interested in when working with higher-order derivatives. We refer here to Lai's original paper on higher-order derivatives [22] and Knudsen's paper on higher-order differentials [20] for reference and notation.

Theorem 18. *A differential of order t is a special case of a truncated 2^t-polytopic transition. In particular, its probability is the sum of the probabilities of all 2^t-polytopic trails that adhere to the truncated 2^t-polytopic transition.*

Proof. Let $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$. Let $(\alpha_1, \ldots, \alpha_t)$ be the set of linearly independent differences that are used as the base for our derivative. Let $L(\alpha_1, \ldots, \alpha_t)$ denote the linear space spanned by these differences. Let furthermore β be the output difference we are

interested in. The probability of the t-th order differential $\Delta_{\alpha_1,\ldots,\alpha_t} f(\mathbf{X}) = \beta$ is then defined as the probability that

$$\sum_{\gamma \in L(\alpha_1,\ldots,\alpha_t)} f(\mathbf{X} \oplus \gamma) = \beta \tag{2}$$

holds with $\mathbf{X}$ being a random variable, uniformly distributed on $\mathbb{F}_2^n$.

Let B now be the truncated $(2^t - 1)$-difference defined as

$$B := \left\{ (\delta_1, \ldots, \delta_{2^t-1}) \;\middle|\; \sum_{i=1}^{2^t-1} \delta_i = \beta \right\}.$$

Let $\gamma_1, \gamma_2, \ldots, \gamma_{2^t-1}$ be an arbitrary ordering of the non-zero elements of the linear space $L(\alpha_1, \ldots, \alpha_t)$ and let $\boldsymbol{\alpha} = (\gamma_1, \ldots, \gamma_{2^t-1})$ be the $(2^t - 1)$-difference consisting of these. We will then show that the probability of the t-th order differential $(\alpha_1, \ldots, \alpha_t, \beta)$ is equal to the the probability of the truncated 2^t-polytopic transition $\boldsymbol{\alpha} \xrightarrow{f} B$. With $\mathbf{X}$ being a random variable, uniformly distributed on $\mathbb{F}_2^n$, we have

$$\begin{aligned}
&\Pr\left(\boldsymbol{\alpha} \xrightarrow{f} B\right) \\
&= \Pr_{\mathbf{X}}\left(\sum_{i=1}^{2^t-1} \Big(f(\mathbf{X} \oplus \gamma_i) \oplus f(\mathbf{X})\Big) = \beta\right) \\
&= \Pr_{\mathbf{X}}\left(\sum_{i=1}^{2^t-1} \Big(f(\mathbf{X} \oplus \gamma_i)\Big) \oplus f(\mathbf{X}) = \beta\right) \\
&= \Pr_{\mathbf{X}}\left(\sum_{\gamma \in L(\alpha_1,\ldots,\alpha_t)} \Big(f(\mathbf{X} \oplus \gamma)\Big) = \beta\right)
\end{aligned}$$

which proves the theorem. □

Example. Let α_1 and α_2 be two differences with respect to which we want to take the second order derivative and let β be the output value we are interested in. The probability that $\Delta_{\alpha_1,\alpha_2} f(\mathbf{X}) = \beta$ for uniformly randomly chosen $\mathbf{X}$ is then nothing else than the probability that the 3-difference $(\alpha_1, \alpha_2, \alpha_1 \oplus \alpha_2)$ is mapped to a 3-difference $(\beta_1, \beta_2, \beta_3)$ with $\beta_1 \oplus \beta_2 \oplus \beta_3 = \beta$.

This theoretical connection between truncated and higher-order differentials has an interesting consequence: a higher-order differentials can be regarded as the union of polytopic trails. This principally allows us to determine lower bounds for the probability of higher-order differentials by summing over the probabilities of a subset of all polytopic trails that it contains – just as we are used to from standard differentials.

As shown in Lemma 9, the probability of a $(d+1)$-polytopic trail is always at most as high as the probability of the worst standard differential trail that it contains. A situation in which the probability of a higher-order differential at the same time is dominated by a single polytopic trail and has a higher probability than any ordinary differential can thus never occur. To find a higher-order differential with a higher probability than any ordinary differential for a given cipher, we are thus always forced to sum over many polytopic trails. Whether this number can remain manageable for a large number of rounds will require further research and is beyond the scope of this paper.