ANALYSIS OF LIGHTWEIGHT AND EFFICIENT SYMMETRIC-KEY PRIMITIVES

RALPH ANKELE

INFORMATION SECURITY GROUP

ROYAL HOLLOWAY UNIVERSITY OF LONDON

SUPERVISOR

Prof. Dr. Carlos Cid

EXAMINERS

Prof. Dr. Sean Murphy

Prof. Dr. Vincent Rijmen

THESIS SUBMITTED FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

APRIL 6, 2019

DECLARATION OF AUTHORSHIP

These doctoral studies were conducted under the supervision of Prof. Carlos Cid.

The work presented in this thesis is the result of original research carried out by myself, in collaboration with others, whilst enrolled in the Department of Information Security as a candidate for the degree of Doctor of Philosophy. This work has not been submitted for any other degree or award in any other university or educational establishment.

Ralph Ankele
April 6, 2019

ACKNOWLEDGEMENTS

First and foremost, I wish to thank my supervisor Carlos Cid. Throughout my PhD you provided me with guidance, support and advice. Moreover, you encouraged me to explore my own topics and interests that made this three years of my PhD studies a very enjoyable experience. I would also like to thank my advisor Kenny Paterson for many interesting discussions. Many thanks also to Miroslav Knežević who hosted me during my internship at NXP Semiconductors in Leuven, and to Gregor Leander who hosted me during my research visit at Ruhr University Bochum.

Many thanks are due to my co-authors that I have had the privilege to work with. In particular, I would like to thank Robin Ankele, Subhadeep Banik, Erik Boss, Avik Chakraborti, Christoph Dobraunig, Jian Guo, Miroslav Knežević, Stefan Kölbl, Eran Lambooij, Gregor Leander, Eik List, Marco Martinoli, Florian Mendel, Siang Meng Sim, Yosuke Todo, Tolga Yalçın and Gaoli Wang. Moreover, I would like to thank the cryptographic community, and all the nice people I have met at several conferences/workshops/schools for interesting discussions and the activities apart from the actual events.

Moreover, I want to express my gratitude to the ECRYPT-NET project. I am very grateful for all the organised events, the travel funding that allowed me to visit 37 events during my PhD and for connecting me to a huge research network of some of the top researchers in the cryptographic community. Special thanks to all the ECRYPT-NET fellows, Gustavo Banegas, Erik Boss, Dušan Božilov, Henry de Valence, Simon Friedberger, Marie-Sarah Lacharité, Chaoyun Li, Marco Martinoli, Michele Minelli, Matthias Minihold, Lorenz Panny, Răzvan Rosie, Danilo Šijačić, Eduardo Soria-Vázquez, and Junwei Wang.

None of this would have been possible without the support from my family and friends during my entire studies. In particular I would like to thank my parents, Bettina and Walter, for their encouragement and support. I would also like to thank my twin-brother Robin, for many discussions during my PhD and comments that improved this thesis.

Finally, I would like to thank my better half, Rebeca, for all the endless support during my studies, moving with me abroad to pursue my PhD, for making me laugh and for pretending to be interested in my work, when I once again mansplained the importance of my PhD studies in too much detail. You alone are my biggest support of all. Thank you for being who you are, and no one else.

Ralph Ankele
April 6, 2019

ABSTRACT

Symmetric-key primitives are used to ensure the confidentiality of two or more parties to maintain a private communication channel. While legacy cryptographic primitives just ensure the confidentiality of messages, modern symmetric-key algorithms further grant integrity and authenticity, simultaneously.

With the rise of the Internet, and the flourishing digitalisation of many services, privacy and security of those communication channels became more and more important. Many conventional standards and recommendations for cryptographic algorithms have been published. Those include for block ciphers the Data Encryption Standard (DES), and the Advanced Encryption Standard (AES), and for hash functions the Secure Hash Algorithms SHA-1, SHA-2, and Keccak as SHA-3. Yet, modern cryptographic algorithms are designed to run on desktop/server systems, however, in resource constrained environments those conventional cryptographic standards are often inefficient or difficult to implement. Lightweight cryptography emerged from the lack of primitives that are capable of running in highly constraint but interconnected environments (*i.e.,* sensor networks, automotive systems, RFID tags, and smart grids) the Internet of Things (IoT) in general.

This thesis presents novel advances in those areas. The research in this thesis is split in two parts, contributing to the foundations and the cryptanalysis of lightweight and efficient symmetric-key primitives.

In Part I of this thesis, we present a broad study of different design strategies of lightweight block ciphers against the security of differential cryptanalysis. Accordingly, we show that many lightweight ciphers have a significant gap between single differential trails and differentials. Furthermore, we study energy-efficient S-boxes, that are an important building block of ciphers based on substitution permutation networks. In our research, we analyse all 4-bit S-boxes and give recommendations for S-boxes with low-energy consumption that can be used in battery-powered embedded devices such as medical implants.

Part II focuses on cryptanalysis of lightweight block ciphers. First we present zero-correlation attacks on the STK construction of the Tweakey framework, by considering linear masks in the tweakey schedule. By transforming the attacks to integral attacks, we can reduce the data complexity and show attacks on the tweakable lightweight block ciphers Qarma, Skinny and Deoxys. Second, we study related-tweakey impossible differential attacks against the tweakable lightweight block cipher Skinny where we present an attacks on 23 (out of 36) rounds on Skinny-64/128. Third, we study differential attacks on reduced-round versions of the block cipher family Sparx. Sparx has recently been published and it is the first ARX-based block cipher with provable bounds against differential and linear cryptanalysis. In our work, we show truncated differential and rectangle attacks on several reduced-round versions of Sparx.

CONTENTS

1 INTRODUCTION

CONTENTS

1.1 MOTIVATION

Over the last few years, the number of electronic devices connected to the Internet have drastically increased. While the first electronic device, that has been connected to the Internet (*i.e.,* at that time the *ARPANET*) was a *Coca-Cola drink dispenser* [319], nowadays there are billions of devices connected to the Internet. This network of embedded devices, home appliances, vehicles, sensor networks, or many other embedded devices containing electronics, sensors, and actuators is called the *Internet of Things (IoT)*. The number of IoT devices is increasing by around 30% year-over-year, and initial estimates are showing that there will be around 30 billion connected devices by 2020 [268]. Moreover, the global market value of the IoT is predicted to reach 7.1 trillion US$ by 2020 [176]. There is an extensive market for IoT devices showing the huge benefits of connected devices. This include home automation, elder care, medical and health care, smart traffic control, transport systems, smart grids, intelligent maintenance systems, smart cities, and many further use cases. Home automation systems allow to control lightning, heating, air conditioning, media and security systems that in a long term can benefit the energy consumption in flats and houses. Elderly care systems and smart houses can provide assistance for people with disabilities, with systems offering voice control or sensors that monitor for medical emergencies. Moreover, in the medical and health care setting mobile hearth monitors and glucose monitoring systems for diabetes patients, can connect to a smart phone and be used to, for example monitor the medicine intake. Furthermore, smart traffic control, smart parking and electronic toll collection systems can speed up waiting times and allow a better traffic flow. In industry, RFID-chips can be used on supply chain networks. Moreover, in agriculture, temperature, humidity and wind speed sensors can be used to improve the quality and quantity of growing crops.

While there are many benefits and use cases with connected devices, there are also plenty of security and privacy issues coming along when devices are designed/manufactured without consulting security-aware engineers. Moreover, often standard cryptographic algorithms do not fit on embedded devices or decrease the efficiency, and are therefore often ignored. There are many examples for major security and privacy breaches that occurred in a short timespan. In October 2016 one of the largest *Distributed Denial of Service (DDoS)* attacks was launched on Dyn, a Domain Name Service provider, using an IoT botnet [66]. Consequently, parts of the Internet went down, including Twitter, Netflix, CNN, Reddit and the Guardian. The botnet was created with the malware *Mirai* and distributed to other IoT devices by using default usernames and passwords. Shortly later, in November 2016 Ronen *et al.* [289] showed a massive attack on the *Zigbee Light Link* protocol, where in particular they were targeting *Philips Hue smart lamps*. In their attack, they use a side-channel attack to extract the global AES-CCM key and then to spread a worm via the over-the-air (OTA) update mechanism to further infect other smart lamps. They could then use the malicious smart lamps in large scale DDoS attacks. Moreover, to show the seriousness of attacks against IoT devices, in 2015 Miller and Valasek presented an attack to take over the whole control system of a Jeep Cherokee [247]. In their attack, they exploit a predictable Wi-Fi password in the multimedia system to update the firmware. The multimedia system did not have proper authenticity checks making the attack possible. As the multimedia system is connected to the *Control Area Network (CAN)* that connects around 70 electronic control units including engine control, transmission, airbags and braking, the attackers were able to take full control of the car. Another safety critical vulnerability of medical implants was published in January 2017. Hackers were able to exploit a vulnerability in the transmitter of cardiac devices [218], such as pacemakers and defibrillators. The vulnerability occurred when a medical implant remotely shared its data with physicians. The hackers were able to show that they can deplete the battery of the implant or even administer incorrect pacing or shocks. Apart from security and safety issues, the lack of security in IoT devices, also leads to severe privacy issues. In February 2017, the German Federal Network Agency released a press statement [158] about some children's toys that can be used to spy on people. In particular, the *Cayla doll* had several critical vulnerabilities that can be used to access cameras and microphones, that are part of the doll, via the Internet.

The cryptographic research community, industry and standardisation agencies recognized the security problems within IoT devices, and all the involved bodies are eager to find solutions. As a result, in recent years many new ciphers have been published that are optimised for resource-constrained environments. The overall research area is called *Lightweight Cryptography*. Security and privacy are important in embedded devices, as we mentioned above by showcasing several recent

vulnerabilities in IoT devices. However, as the majority of modern cryptographic algorithms are designed for implementation on desktop/server systems, many of those algorithms become inefficient or are impossible to implement in constrained devices.
While there has been a significant effort by the academic research community and industry to efficiently implement standard algorithms, such as DES and AES, on resource constrained devices, there are still further issues with many legacy algorithms. There are many results on hardware [38, 44, 331] and software [239, 275, 305] implementations of AES that allow very small implementations with an area of around 2200 GE and a latency of 246 cycles per byte for encryption. Moreover, some microprocessors are often shipped with crypto co-processors that offer hardware acceleration for AES [165].
Nevertheless, hardware acceleration also reaches its limits and too many optimisations allow for *side-channel attacks*. Moreover, optimisations for speed are not always enough, as many embedded devices also require small memory and code footprints. AES is a reasonable fast cipher when implemented in embedded environments, however, its large block size and S-box do not allow small implementations, and further AES is vulnerable to side-channel attacks. Additionally, other cryptographic standards such as SHA-2 and SHA-3 require a large amount of memory, for storing the internal states (*i.e.,* 512 bits for SHA-2 and 1600 bits for SHA-3).
Biryukov and Perrin provided a large study about the *state-of-the-art in lightweight symmetric cryptography* [81]. In their work they listed many proprietary/legacy algorithms mainly proposed from industry, where many of them have been broken [41, 179]. Table 1 gives an overview of some proprietary/legacy lightweight algorithms identified by Biryukov and Perrin.
The cryptographic research community published an extensive number of lightweight cryptographic primitives in the last few years to overcome the issues with proprietary/legacy algorithms. Most of the modern lightweight cryptographic algorithms are from academia, while a few are from industry. Modern lightweight ciphers are often *lightweight* by design. This lightweightness can also be seen as a set of specific design choices. In the following we will list a few design choices that were recently used for designing dedicated lightweight symmetric primitives. Moreover, we often have to decide between different trade-off's, when designing a lightweight cipher. Figure 1 illustrates the trade-off's for lightweight cryptography.
Cryptographic primitives can be implemented in software to secure protocol communication and encrypt data that needs to be stored securely. Among the relevant metrics for software implementations are the amount of data that is written to memory (RAM consumption), the code size, and the throughput. Many microcontrollers operate on small words of 8, 16, 32 bits. For smaller microcontrollers, *e.g.,* 8-bit processors, the costs of rotations are quite expensive, however, rotations by the word-size of the microcontroller are usually cheap. Therefore, many recent lightweight ciphers are designed to use rotations by multiples of the word-size, which are cheaper to implement. Some further tricks are to pre-compute often used variables, and also for example subkeys of a cipher. This can save time, at the expense of more memory. More sophisticated techniques are bit-sliced implementations [237].

Table 1: Proprietary/legacy lightweight primitives as identified by Biryukov and Perrin [81]

Name	Intended platform	Key	Internal State	IV	Reference
A5/1	Cell Phones	64	64	22	[104]
A5/2		64	81	22	[42]
CMEA		64	16-48	-	[339]
ORYX		96	96	-	[340]
A5-GMR-1	Satellite Phones	64	82	19	[143]
A5-GMR-2		64	68	22	[143]
DSC	Cordless phones	64	80	35	[235]
SecureMemory	Atmel chips	64	109	128	[161]
CryptoMemory		64	117	128	[161]
Hitag2	Car key / immobilizers	48	48	64	[333]
Megamos		96	57	56	[334]
Keeloq		64	32	-	[179]
DST40		40	40	-	[100]
iClass	Smart cards	64	40	-	[160]
Crypto-1		48	48	96	[266]
CSS	DVD players	40	42	-	[47]
Cryptomeria		56	64	-	[102]
CSA-BC	Digital televisions	64	64	-	[346]
CSA-SC		64	103	64	[346]
PC1	Amazon Kindle	128	152	-	[80]
SecurID	Secure token	64	64	-	[78]
E0	Bluetooth devices	128	128	-	[358]

Many efficient implementations are also directly implemented in hardware. Among the relevant metrics for efficient hardware implementations are the memory consumption, the implementation's size (normally measured in Gate Equivalent (GE)), latency, throughput, and power consumption. All of those metrics are related to each other, and it is important to find trade-off's according to a specific use case. As memory is often one of the most expensive parts, lightweight ciphers are designed to operate on smaller block sizes and use smaller key sizes.

Another desirable design feature of lightweight ciphers is resilience against side-channel attacks. In side-channel attacks, leakage information from real-world phys-

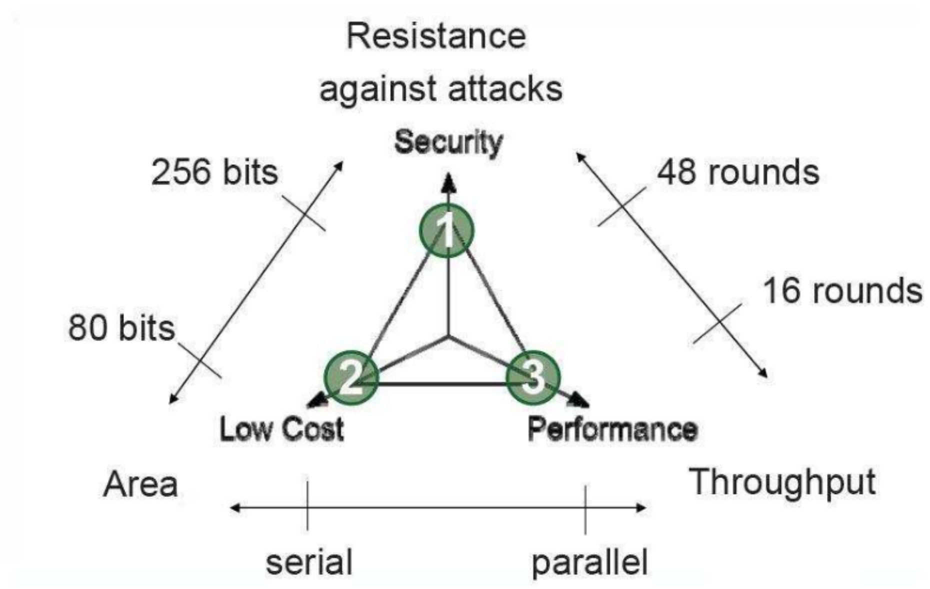


Figure 1: Trade-off's in Lightweight Cryptography between Security/Throughput/Area[1]

ical processes such as power consumption or varying execution times are exploited to extract secret information from a cipher. Such attacks require an attacker to have physical access to the device. To prevent against such attacks, several countermeasures have to be applied, which often add severe overhead to the implementation of a cipher. Masking [280] is an implementation technique to protect against side-channel attacks, which uses an external source of random data to randomize the input of operations that might leak information.

The components and operations in a lightweight block cipher are typically simpler than in standard block ciphers like AES. In contrast to simplifying the round functions, often the number of rounds has to be increased to achieve the same security. As memory is very expensive the implementation of a S-Box as look-up table can lead to a large hardware footprint. Therefore, lightweight block ciphers have usually small (*e.g.,* 4-bit) S-Boxes. To save further memory, lightweight block ciphers are using small block sizes (*i.e.,* 64 bits, rather then 128 bits). Another option is to reduce the key sizes to 80 or 96 bits for efficiency. As a result, simpler key schedules improve the memory, latency and power consumption of lightweight block ciphers. In 2007 Bogdanov *et al.* [95] proposed PRESENT, an ultra-lightweight block cipher based on a Substitution-Permutation Network that is optimised for hardware and can be implemented with just 1075 GE. PRESENT is bit-oriented and has a hardwired diffusion layer. In 2011, Guo *et al.* [168] designed LED, an SPN cipher that is heavily

[1] Lightweight cryptography trade-off's as identified by Gregor Leander. Available at: `https://www.cosic.esat.kuleuven.be/ecrypt/courses/albena11/slides/gregor_leander_lightweight.pdf`. Date accessed: 19. Oct. 2018

based on AES. Interesting in that design is the lack of the key schedule, as it applies the same 64-bit key every four rounds to the state for LED-64. The 128-bit version simply divides the key in two 64-bit sub-keys and then alternately adds them to the state. Reducing the latency is the main goal of the block cipher PRINCE [101]. There is no real key schedule in PRINCE, as it derives three 64-bit keys from a 128-bit master key. PRINCE is a reflection cipher, meaning that the first rounds are the inverse of the last rounds, so that the decryption of a key k is identical to an encryption with key $k \oplus \alpha$, where α is a constant based on π. The block cipher MIDORI [37] was designed for reducing the energy consumption when implemented in hardware. It has an AES-like structure and a very lightweight almost-MDS involutory matrix M as diffusion layer. In 2013, SIMON and SPECK [46] were designed by the NSA. Both ciphers perform exceptionally well in both hardware and software and were recently considered for standardisation. Compared to the standard approach, no security analysis or design rational was given by the designers. SIMON is hardware-oriented and based on a Feistel-Network with only the following operations: AND, rotation, XOR. SPECK is software oriented and based on an ARX construction with the typical operations: addition, rotation, XOR. In 2016, SKINNY [50] has been published to compete with SIMON. The main idea behind the design is to be efficient as possible but without sacrificing security. SKINNY is a tweakable block cipher based on the TWEAKEY framework [188] with the components chosen because of a good compromise between cryptographic properties and hardware costs.

Conventional hash functions such as SHA-1, SHA-2 and SHA-3 (*i.e.,* KECCAK) may not be suitable for constraint environments due to their large internal state sizes and high power consumption. Lightweight hash functions differ in various aspects as they are optimised for smaller message sizes and/or have smaller internal states and output sizes. PHOTON [167] is a P-Sponge based AES-like hash function, with an internal state size of 100 to 288 bits and an digest of 80 to 256 bits. The state update function is close to the LED cipher. In 2011, Bogdanov *et al.* [94] designed SPONGENT, a P-Sponge where the permutation is a modified version of the block cipher PRESENT. SIPHASH [31] has an ARX structure and is inspired by BLAKE and SKEIN and has a digest size of 64 bits.

Stream ciphers generate a key stream from a given key k and an initialization vector IV, which is then simply XORed with the plaintext to generate a ciphertext. It must be infeasible for an attacker to retrieve the key, even if a large part of the keystream is available to the attacker. In 2008, the *eSTREAM competition* aimed to identify a portfolio of stream ciphers that should be suitable for widespread adoption. Three of the finalists are suitable for hardware applications in a restricted environment. GRAIN was designed by Hell *et al.* [170] and is based on two finite state registers whose clocking influence each others update function to make it non-linear. GRAIN requires 3239 GE in hardware. MICKEY [35] is based on two linear feedback shift registers (LFSR) that are irregularly clocked. MICKEY requires 3600 GE in hardware. TRIVIUM is also a finalist from the eSTREAM competition that has three LFSR's with different length. TRIVIUM [129] requires 3488 GE in hardware.

The aim of authenticated encryption is to provide confidentiality and integrity (*i.e.,* data authenticity) simultaneously. In 2014, the *CAESAR (Competition for Authenticated Encryption: Security, Applicability and Robustness)* competition started with the aim to identify a portfolio of authenticated ciphers that offer advantages over AES-GCM and are suitable for widespread adoption.
ACORN [351] is based on six LFSR's and has a state size of 293 bits. ACORN provides full security, for both, encryption and authentication. The hardware costs should be close to that of Trivium according to the designers. SCREAM [164] is a tweakable block ciphers in the Tweakable Authenticated Encryption (TAE) mode. SCREAM is based on LS designs Robin and Fantomas. Bertoni *et al.* designed Ketje [62] that is a lightweight variant of SHA-3. Ketje relies on the Sponge construction in the MonkeyWrap mode. The internal state size is only 200 bits for Ketje-Jr and 400 bits for Ketje-Sr. Ascon [140] is an easy to implement, Sponge-based authenticated cipher with a custom tailored SPN cipher. It is fast in both, hardware and software even with added countermeasures against side-channel attacks. Another CAESAR candidate is the 64-bit tweakable block cipher Joltik [186], that is based on the Tweakey framework. Joltik is AES-like and uses the S-Box of Piccolo and the round constants of LED. The MDS matrix is involutory and non-circulant.

The contributions of this thesis are in the cryptanalysis of symmetric-key primitives and the analysis of several components of symmetric-key ciphers. Third-party cryptanalysis of symmetric-key primitives ensures that there is more trust in the security claims of designers, by further offering insights in the security of the analysed ciphers. Moreover, as the design of cryptographic primitives is often limited by time constraints and the huge choice of different components in ciphers, the initial attacks and analysis is naturally limited. Therefore, it demands a deeper analysis by the cryptographic community. Within this thesis, some novel ideas are used to extend the currently best attacks on several recently proposed lightweight ciphers. Moreover, we study the components of symmetric-key primitives and some cryptographic attacks in general.

1.2 PUBLICATIONS

This thesis is based on the research contributions which the author performed during his PhD studies. The majority of the work has been published at conferences. The papers are listed below:

Conference Papers:

1. Ankele, R., Banik, S., Chakraborti, A., List, E., Mendel, F., Sim, S.M., Wang, G.: Related-key impossible-differential attack on reduced-round Skinny. In: Gollmann, D., Miyaji, A., Kikuchi, H. (eds.) ACNS 17: 15th International Conference on Applied Cryptography and Network Security. Lecture Notes in

Computer Science, vol. 10355, pp. 208–228. Springer, Heidelberg, Germany, Kanazawa, Japan (Jul 10–12, 2017)

2. Ankele, R., Böhl, F., Friedberger, S.: MergeMAC: A MAC for authentication with strict time constraints and limited bandwidth. In: Preneel, B., Vercauteren, F. (eds.) ACNS 18: 16th International Conference on Applied Cryptography and Network Security. Lecture Notes in Computer Science, vol. 10892, pp. 381–399. Springer, Heidelberg, Germany, Leuven, Belgium (Jul 2–4, 2018)
3. Ankele, R., List, E.: Differential cryptanalysis of round-reduced Sparx-64/128. In: Preneel, B., Vercauteren, F. (eds.) ACNS 18: 16th International Conference on Applied Cryptography and Network Security. Lecture Notes in Computer Science, vol. 10892, pp. 459–475. Springer, Heidelberg, Germany, Leuven, Belgium (Jul 2–4, 2018)
4. Ankele, R., Kölbl, S.: Mind the gap - a closer look at the security of block ciphers against differential cryptanalysis. In: Cid, C., Jacobson, M.J. (eds.) Selected Areas in Cryptography – SAC 2018. Springer International Publishing, Cham (2018)

In Submission:

1. Ankele, R., Dobraunig, C., Guo, J., Lambooij, E., Leander, G., Todo, Y.: Zero-correlation attacks on the TWEAKEY framework. IACR Transactions on Symmetric Cryptology 2018(4) (Sep 2018), submitted

Pre-Prints:

1. Ankele, Ralph, Ankele, Robin: Software benchmarking of the 2nd round CAESAR candidates. Cryptology ePrint Archive, Report 2016/740 (2016), `http://eprint.iacr.org/2016/740`
2. Ankele, R., Banik, S., Chakraborti, A., List, E., Mendel, F., Sim, S.M., Wang, G.: Related-key impossible-differential attack on reduced-round SKINNY. Cryptology ePrint Archive, Report 2016/1127 (2016), `http://eprint.iacr.org/2016/1127`
3. Ankele, R., List, E.: Differential cryptanalysis of round-reduced Sparx-64/128. Cryptology ePrint Archive, Report 2018/332 (2018), `https://eprint.iacr.org/2018/332`
4. Ankele, R., Böhl, F., Friedberger, S.: MergeMAC: A MAC for authentication with strict time constraints and limited bandwidth. Cryptology ePrint Archive, Report 2018/342 (2018), `https://eprint.iacr.org/2018/342`
5.

Technical Reports:

1. Ankele, R., Banegas, G., Boss, E., Božilov, D., Friedberger, S., Lacharité, M.S., Li, C., Martinoli, M., Minelli, M., Minihold, M., Rosie, R., Šijačić, D., Soria-

Vázquez, E.: Technical report on first designs for IoT & cloud. Tech. rep., European Integrated Research Training Network on Advanced Cryptographic Technologies for the Internet of Things and the Cloud (2017)

2. Ankele, R., Banegas, G., Boss, E., Božilov, D., Friedberger, S., Lacharité, M.S., Li, C., Martinoli, M., Minelli, M., Minihold, M., Panny, L., Rosie, R., Šijačić, D., Soria-Vázquez, E., Wang, J.: Technical report on implementations for IoT & cloud. Tech. rep., European Integrated Research Training Network on Advanced Cryptographic Technologies for the Internet of Things and the Cloud (2018)

1.3 THESIS STRUCTURE

This thesis presents some selected research contributions of the author during his studies at the Information Security Group at Royal Holloway University of London. The contributions were accomplished under the supervision of Prof. Carlos Cid. In the following we give a detailed summary of the thesis outline.

1.3.1 Part I - Foundations

Part I of this thesis presents novel research into the foundations of symmetric cryptography. In Chapter 3 we study some aspects of differential cryptanalysis applied to many lightweight block ciphers. Differential cryptanalysis is one of the most powerful attack vectors in symmetric cryptography, first published in the early 90s and since then many extensions and further insights have been published. Chapter 4 presents research in some of the building blocks of block ciphers. We analyse all possible 4-bit S-boxes regarding their energy consumption. Due to the interconnectivity of many devices and the need for security in embedded devices such as sensor networks, RFID tags and medical implants, the Internet of Things (IoT) in general, many of those devices are battery-powered and the energy consumption plays a critical roll in the performance evaluation of ciphers used in those devices.

Chapter 3 - Differential Cryptanalysis of Lightweight Block Ciphers

In this chapter, we study the effects of differential cryptanalysis of several recently proposed lightweight block ciphers. Resistance against differential cryptanalysis is an important design criteria for any modern block cipher and most designs rely on finding some upper bound on probability of single differential trails. However, already at EUROCRYPT'91, Lai *et al.*comprehended that differential cryptanalysis rather uses *differentials* instead of single *trails*.
We consider exactly the gap between these two approaches and investigate this gap in the context of recent lightweight cryptographic primitives. This shows that for many recent designs like Midori, Skinny or Sparx one has to be careful as bounds from counting the number of active S-boxes only give an inaccurate evaluation of the best differential distinguishers. For several designs we found new differential distinguishers and show how this gap evolves. We found an 8-round differential

distinguisher for SKINNY-64 with a probability of $2^{-56.93}$, while the best single trail only suggests a probability of 2^{-72}. Our approach is integrated into publicly available tools and can easily be used when developing new cryptographic primitives.
Moreover, as differential cryptanalysis is critically dependent on the distribution over the keys for the probability of differentials, we provide experiments for some of these new differentials found, in order to confirm that our estimates for the probability are correct. While for SKINNY-64 the distribution over the keys follows a Poisson distribution, as one would expect, we noticed that SPECK-64 follows a bimodal distribution, and the distribution of MIDORI-64 suggests a large class of weak keys.

Chapter 4 - Analysis of Low-Energy 4-bit S-boxes

Many devices in resource constrained environments, the Internet of Things (IoT) in general, are powered by batteries. Thus, those devices often operate on a tight power/energy budget. Medical implants such as pacemakers, insulin pumps or brain implants are some examples for this devices. Security and privacy is crucial in the communication channel of those devices.
Lightweight cryptography is an active field of research and there have been many cipher proposals in the last few years. In conventional cryptographic standards the trade-off between security and performance is optimised for high performance environments. However, in resource constrained environments those cryptographic standards are difficult or impossible to implement. Hence, lightweight ciphers have been optimized for area, power consumption, memory complexity, latency and throughput. Yet, little work has been done on energy efficient ciphers.
We are trying to fill this gap and give a detailed study on energy-efficient design strategies for block ciphers. Moreover, we concentrate our research on Substitution-Boxes (S-boxes) that are an important building block of Substitution Permutation Networks (SPN). We analyse all *optimal* 4×4-bit S-boxes, and classify them in two groups, based on PRESENT-like and PRINCE-like designs, that we show to be optimal block cipher designs for low-energy consumption. In that context we further analyse all involutory 4-bit permutations, and study the differential and linear branch numbers of all optimal 4-bit affine equivalence classes.
As a result, we give recommendations for optimal low-energy S-boxes, that cipher designers can use in their ciphers, for instance in the upcoming lightweight cryptography standardisation process by the National Institute of Standards and Technology (NIST).

1.3.2 Part II - Cryptanalysis

Part II of this thesis presents novel research in the cryptanalysis of some recently proposed lightweight block cipher families. The analysis of block ciphers is an important step to strengthen the trust and the reliability in a cipher. In Chapter 5 we analyse the security of the TWEAKEY framework, with applications to the tweakable block ciphers QARMA, MANTIS and SKINNY. Moreover, in Chapter 6 we analyse the security of the tweakable block cipher SKINNY. Finally, in Chapter 7 we analyse the security of Sparx, an ARX-based block cipher designed according to a new design strategy called the *long-trail strategy* (LTS).

Chapter 5 - Zero-Correlation Attacks on Tweakable Block Ciphers with a linear Tweak Schedule

In this chapter, we present zero-correlation attacks on tweakable block ciphers with a linear tweak schedule, based on [23]. The design and analysis of dedicated tweakable block ciphers is a quite recent and very active research field that provides an ongoing stream of new insights. For instance, results of Kranz, Leander, and Wiemer from FSE'17 show that the addition of a tweak using a linear tweak schedule does not introduce new linear characteristics. We consider for the first time—to the best of our knowledge— the effect of the tweak on zero-correlation linear cryptanalysis. It turns out that the tweak can be used to get zero-correlation linear hulls covering more rounds, which also implies the existence of integral distinguishers on the same number of rounds. The so obtained integral distinguishers cover more rounds compared to existing ones that have been found using the division property, for the tweakable block ciphers Qarma, Mantis, and Skinny. In particular, this leads to the best attack (with respect to number of rounds) on a round-reduced variant of Qarma.

Chapter 6 - Cryptanalysis of the Tweakable Block Cipher Skinny-64/128

In this chapter, we present a related-tweakey impossible-differential attack on a round-reduced version of the tweakable block cipher Skinny based on [20]. Furthermore, we present some unpublished results on integral distinguisher of Skinny.
At CRYPTO'16, Beierle *et al.* presented Skinny, a family of lightweight tweakable block ciphers intended to offer an alternative to the NSA designs Simon and Speck. Skinny can be implemented efficiently in both software and hardware and supports block sizes of 64 and 128 bits as well as tweakey sizes of 64, 128, 192 and 128, 256, 384 bits respectively. In this chapter we present a related-tweakey impossible-differential attack on up to 23 (out of 36) rounds of Skinny-64/128 for different tweak sizes. All our attacks can be trivially extended to Skinny-128/128.

Chapter 7 - Differential Cryptanalysis of Round-Reduced Sparx-64/128

In this chapter, we analyse the lightweight block cipher Sparx presented at ASIACRYPT 2016. We present truncated-differential attacks and rectangle attacks on a round-reduced version of Sparx based on [27].
Sparx is a family of ARX-based block ciphers designed according to the *long-trail strategy* (LTS) that was introduced together with Sparx by Dinu *et al.* at ASIACRYPT 2016. Similar to the wide-trail strategy, the LTS allows provable upper bounds on the length of differential characteristics and linear paths. Thus, the cipher is a highly interesting target for third-party cryptanalysis. However, the only third-party cryptanalysis on Sparx-64/128 to date was given by Abdelkhalek *et al.* at AFRICACRYPT 2017 who proposed impossible-differential attacks on 15 and 16 (out of 24) rounds.

We present chosen-ciphertext differential attacks on 16 rounds of Sparx-64/128. First, we show a truncated-differential analysis that requires 2^{32} chosen ciphertexts and approximately 2^{93} encryptions. Second, we illustrate the effectiveness of boomerangs on Sparx by a rectangle attack that requires approximately $2^{59.6}$ chosen ciphertexts and about $2^{122.2}$ encryption equivalents.

2 BACKGROUND

CONTENTS

2.1 NOTATION

We denote by $\mathbb{F}_2$ the finite field of two elements $x \in \{0,1\}$. For positive integer n, we denote by $\mathbb{F}_2^n$ the n-dimensional vector space over $\mathbb{F}_2$. We represent functions by upper case letters and indices by lowercase letters. We denote with $\{0,1\}^n$ the set of all n-bit strings and $\{0,1\}^*$ the set of bit strings of arbitrary length. Let $x, y \in \{0,1\}^n$ for some positive integer n then, we denote by $x \| y$ the concatenation of x and y, by $x \oplus y$ their bitwise `XOR`, by $x \lll r$ a rotation by r bits to the left and by $x \ggg r$ rotation by r bits to the right; moreover, we denote by $x \boxplus y = (x+y) \bmod 2^n$ modular addition, and by $x \boxminus y = (x-y) \bmod 2^n$ modular subtraction. For all bit strings $x \in \{0,1\}^n$, we index the bits $x = (x_{n-1} \dots x_1 x_0)$ where x_{n-1} is the most significant and x_0 the least significant bit of x. Given a bit string $x = (x^1 \| \dots \| x^m) \in (\{0,1\}^{mn})$ consisting of m words of n bit each, we denote by

$$x \lll_n r \overset{\text{def}}{=} (x^1 \lll r) \| \dots \| (x^m \lll r)$$

the word-wise independent rotated value. We overload the notation for tuples of bit strings $x \in (\{0,1\}^n)^m$: $x = (x^1, \dots, x^m)$, to still mean word-wise independent rotation $x \lll_n r \overset{\text{def}}{=} (x^1 \lll r), \dots, (x^m \lll r)$. We use typewriter font to represent hexadecimal values, *e.g.*, `0110` $= 272$. We use the same font but with annotation to represent bit strings, *e.g.*, $(\texttt{0110})_2 = 6$. Moreover, we will use the symbol $*$ at the position of certain bits to indicate that they can take arbitrary values, *e.g.*, $(\texttt{0*10})_2 \in \{2, 6\}$. As a shorthand notation for probabilities p, we often write $h_w = -\log_2(p)$ when the meaning of p is clear from the context.

2.2 MODERN CRYPTOGRAPHY

Cryptography (from the Ancient Greek $\kappa\rho\upsilon\pi\tau o\zeta$, translated *hidden, secret*; and from $\gamma\rho\alpha\phi\epsilon\iota\upsilon$, translated *to write*) is the study of techniques to securely communicate in the presence of third parties. There are various aspects that modern encryption schemes try to achieve, such as *data confidentiality*, *data integrity*, *authentication*, and *non-repudiation*. The application areas of cryptography are reaching wide-spread areas, including *secure communication*, *electronic commerce*, *digital cryptocurrencies*, *chip-based payment cards*, *digital rights management*, and many further use cases.
Pre-modern cryptography, also called historical-cryptography, was mainly based on encrypting messages to hide information if the messages are intercepted by enemies (*i.e.*, adversaries). The history of cryptography reaches back many thousand years until the ancient Greek and Roman empire. The main types of ciphers were simple transposition ciphers, which rearranged the order of letters in a message, or substitution ciphers that replace parts of letters with other letters. Some well known examples are the *CAESAR* cipher; an early substitution cipher that replaces each letter with another letter shifted by a fixed position in the alphabet. The *Vigenère* cipher is a polyalphabetic substitution cipher, by applying a series of *CAESAR* ciphers in sequence, but with different shift values. In the early 20th century, many mechanical encryption devices were invented. Those include the German *Enigma*, the *Lorenz*, the British *Typex*, the US *M-209*, and the USSR *Fialka M-125* machines[1].
With the invention of computers and further research into cryptographic algorithms simple transposition or substitution ciphers became deprecated. Modern cryptography is based on mathematical and computational hard problems, making them hard to be broken by computationally-bounded adversaries. These schemes are often provable secure, based on the underlying hard mathematical problem such as for example the *integer factorization* or the *discrete logarithm problem*. Moreover, there also exist *information-theoretical secure* schemes, that cannot be broken even with unlimited computing power, such as the *one-time pad*. However, these schemes are inefficient for larger instances.

2.2.1 Cryptographic Goals and Principles

The fundamental problem that cryptography tries to solve is to ensure a private communication channel between two or more parties. In this context, we normally consider two parties, named Alice and Bob, that are communicating over an insecure channel, such as the Internet. In our scenario, we consider the communication channel insecure, as there may exist a third party or adversary, named Eve, who behaves malicious and tries to intercept and tamper with the communication between Alice and Bob. Note that neither Alice, Bob or Eve are necessarily physical persons,

[1] `http://cryptomuseum.com/crypto/index.htm`

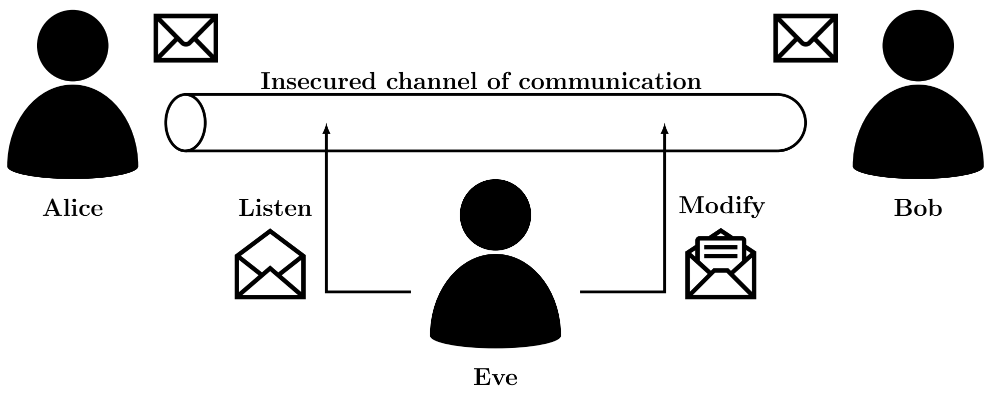


Figure 2: Basic setup of two parties communicating over an insecure channel.

instead they might as well be servers, institutions or governments. Figure 2 illustrates the basic communication setup between two parties over an insecure channel. While in general the main goal of cryptography is to achieve a private communication channel by keeping the communication secret, the aim of modern cryptography is threefold. These properties guarantee the security of the communication between Alice and Bob:

- *Confidentiality:* It must be impossible for a *passive adversary* Eve, meaning that she only listens to the communication between Alice and Bob without modifying any of its content that she intercepts, to obtain any meaningful information.
- *Integrity:* It must be impossible for an *active adversary* Eve, meaning she can tamper with the messages, that a message that has been modified by a third party, in this scenario from Eve, is accepted. In other words, this implies that the communicating parties, Alice and Bob, should be able to detect when a modification of any of their messages exchanged, occurs.
- *Authenticity:* It must be impossible for an *active adversary* Eve, meaning that she can tamper with the messages, withhold the messages or even create completely new messages, to impersonate one of the communicating parties. In particular, Alice and Bob should be able to rely that the messages they receive, is originating either from Alice or Bob.

Authenticity is a special case of integrity. While integrity requires that changes in data are detectable, with authenticity one requires that the data is the same as when it was under control of a specific entity. In that case, *authentication* makes sure that a given entity actually is who one believes it is. Authenticity can then be achieved by providing authentication of data through integrity.

2.2.2 Symmetric vs Public-Key Cryptography

Cryptography can be split in many different areas. Yet, when we want to encrypt a message, we typically work in two different areas, *symmetric-key*, also often called

secret-key cryptography, and *asymmetric-key*, also often called *public-key* cryptography.
Symmetric-key cryptography involves using one single-key to encrypt and decrypt data. The advantage of symmetric-key cryptography is that it is efficient and secure. However, it works well locally, when it is easy to store the key somewhere secure. In a global network, such as the Internet, in order to decrypt the messages a user has to have access to the key. This means, we have an additional overhead, that the symmetric-key has to be exchange initially. Therefore, one would require a complicated key-exchange protocol, or the users have to meet initially to securely exchange keys.
Public-key cryptography solves the issues with the keys by offering a pair of keys. The public key can be sent along with the message, and as its name mentions, it is public accessible. However, the second key of the pair, the private key must always be in the possession of the sender, and must not be disclosed, *i.e.,* it must be kept private. The private key is generated based on a derivation from the public key, and only a receiver that has access to the private key can decrypt messages. Public-key cryptography is therefore often based on mathematical hard problems, such as factoring and the discrete logarithm. Compared to symmetric-key cryptography, public-key cryptography is often very slow and resource intensive.

2.2.3 Kerckhoffs' Principle

In 1883, Auguste Kerckhoffs published six design principles for military ciphers in the journal of military sciences titled *La Cryptographie Militaires* [198](in French):

1. The system must be substantially, if not mathematically, undecipherable;
2. The system must not require secrecy and can be stolen by the enemy without causing trouble;
3. It must be easy to communicate and retain the key without the aid of written notes, it must also be easy to change or modify the key at the discretion of the correspondents;
4. The system ought to be compatible with telegram communication;
5. The system must be portable, and its use must not require more than one person;
6. Finally, given the circumstances in which such system is applied, it must be easy to use and must neither stress the mind or require the knowledge of a long series of rules.

While some of them are not longer relevant, due to the use of computers that run the complex encryption algorithms, Kerckhoffs' second axiom is commonly known as *Kerckhoffs' principle* and is still important in the design of modern cryptographic algorithms. Kerckhoffs' principle basically states that the security of the cipher has to rely exclusively on the secrecy of the key. This is in strong contrast to steganographic encoding, which has been used in ancient Greek and the Roman empire and even further until the nineteenth century, mainly in transposition ciphers of alphanumeric characters.

While Kerckhoffs' principle was followed by many ciphers that are published in academic publications and are used in standards, there have been a few ciphers that were mainly proposed by industry, that did not follow the principle and the security also depends on the secrecy of the algorithms. Some examples are the KEELOQ block cipher, that was used for many remote keyless entry systems in cars/garages, the MIFARE stream cipher which is still widely used in contactless smart cards for bus and train tickets, the DECT cipher for cordless home telephones, baby phones and traffic lights, and also the *Kindle Cipher* (PC1) that has been used by the Amazon Kindle eBook reader as a DRM system. All those ciphers are highly efficient, but they are based on proprietary/not public specifications. Moreover, they do not use standardised design principles, and are practically broken by the cryptographic community [80, 131, 179, 267].
However, even government agencies use an approach of security by obscurity where they further rely on the secrecy of their designs or implementations. This includes a list called *NSA Suite A Cryptography* that includes classified algorithms in the United States of America that are used for especially sensitive data. Moreover, the Russian intelligence agencies use proprietary algorithms such as GOST and KUZNYECHIK, which have been attacked [30, 180] or parts of it have been successfully reverse engineered [82].

2.3 SYMMETRIC PRIMITIVES

Symmetric cryptographic primitives are used in a wide range of applications, such as *secure communication*, *electronic commerce*, *digital cryptocurrencies*, *chip-based payment cards*, *digital rights management*. There are several different primitives that are tailored to achieve the main goals in cryptography *authenticity, confidentiality* and *integrity*. These primitives include *block ciphers, stream ciphers, hash functions, message authentication codes* and *authenticated ciphers*. In the following we will give an overview of those primitives.

2.3.1 Block Ciphers

A *block cipher* is a deterministic algorithm that operates on a number of bits with a fixed length, called a *block*. Block ciphers provide confidentiality of data, by transforming an input message M, that is split in chunks of the block size n, into a ciphertext C. Due to the simplicity of many block ciphers, they are very flexible cryptographic building blocks and are used to design many other cryptographic primitives such as stream ciphers, hash functions, message authentication codes and authenticated encryption schemes. We can formally define a block cipher as follows:

Definition 1. Let k, n be two positive integers, then a block cipher $\mathcal{E}$ is given by

$$\begin{aligned} \mathcal{E}(K, M) : \mathbb{F}_2^k \times \mathbb{F}_2^n &\to \mathbb{F}_2^n \\ (K, M) &\to C, \end{aligned}$$

where $K \in \mathbb{F}_2^k$ is a key of fixed length k. A block cipher represents a permutation over $\mathbb{F}_2^n$, where n is the block length of the cipher. The encryption function $\mathcal{E}$ takes a key K and transforms a message M into a ciphertext C. For a fixed key K we can further define the inverse of the encryption function $\mathcal{D} = \mathcal{E}^{-1}$ to be the decryption function.

In the remainder of this thesis, we will denote the encryption function of block cipher with $\mathcal{E}_k(\cdot) = \mathcal{E}(K, \cdot)$ and the decryption function as $\mathcal{D}_k(\cdot) = \mathcal{D}(K, \cdot)$ respectively. To ensure correctness it has to hold that:

$$\forall K \in \mathbb{F}_2^k, \forall M \in \mathbb{F}_2^n : \mathcal{D}_K(\mathcal{E}_K(M)) = M. \tag{1}$$

A block cipher should therefore, for a fixed key K be a permutation over $\mathbb{F}_2^n$. Typical block sizes are $n = 64, 128$ bits, where 128-bit are considered in a high security use-cases and 64-bit are normally used in lightweight block ciphers that are implemented in constrained environments. The key lengths typically vary between $k = 64, 80, 96, 128, 192, 256$ bits. Recently, the European *Excellence in the Area of Cryptology Network* ECRYPT-CSA published a report [1] recommending 80-bit security for legacy standards[2], 128-bit for near term protection[3] and 256-bit for long term protection[4].
In the design of block ciphers, there are two important concepts that were identified by Shannon in his seminal paper [308] in 1949. These are the concepts of *confusion* and *diffusion*:

- *Confusion:* defines that each bit of the ciphertext C should depend on parts of the secret key K, obscuring the relation between both as much as possible. This can also be interpreted as any statistical property in the ciphertext should not depend on the plaintext.
- *Diffusion:* defines that if a single bit in the plaintext is changed/flipped, then statistically half of the bits in the ciphertext should change. This can also be interpreted as that a block cipher should be highly non-linear. So any statistical property in the plaintext, should not be present in the ciphertext.

Iterated Designs

Most of the modern block ciphers are iterated designs, which means that they apply an invertible transformation, also called a *round function*, repeatedly. Each iteration is refereed to as a *round* in a block cipher. The concept of a round-based block cipher can be written as the composition of several rounds:

$$\mathcal{E}_K = F_{i-1}(K_{i-1}, \cdot) \circ \cdots \circ F_0(K_0, \cdot), \tag{2}$$

where $F_i(K_i, \cdot)$ is the i^{th} round function and K_i is the i^{th} round key. The round function F has to be a bijection in $\mathbb{F}_2^n$ to ensure correctness.
The theoretical concept of an iterated block cipher was first addressed by Even and Mansour [153] in 1991. Their work was motivated by the DESX construction

[2] Legacy standard: Should not be used in new systems
[3] Near term protection: Security for at least ten years (2018-2028)
[4] Long term protection: Security for thirty to fifty years (2018-2068)

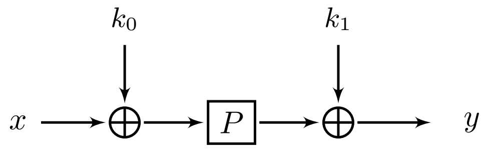


Figure 3: Even Mansour construction.

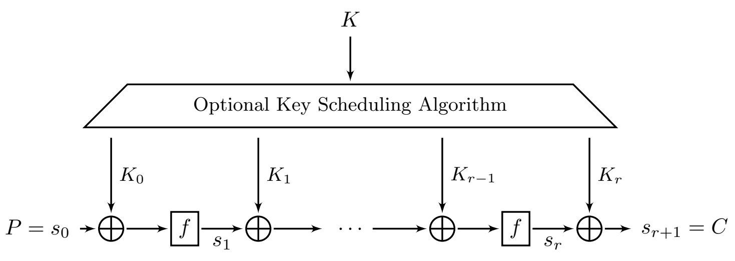


Figure 4: Key-Alternating Cipher.

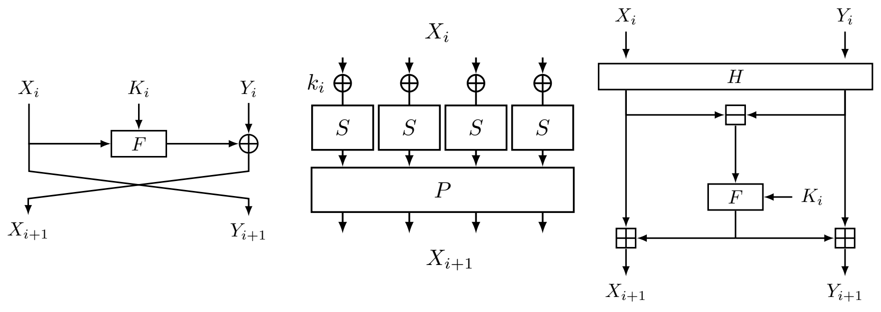
proposed by Rivest [284] in 1984, in which he improved DES by adding two independent pre- and post-whitening keys to the plaintext and ciphertext, respectively. The Even-Mansour scheme replaced the round function of DES in the middle with a publicly known permutation, and kept the whitening keys. The resulting scheme is provable secure and extremely simple. The encryption consists of adding a key K with XOR to the plaintext, applying a publicly known permutation π and then adding the key K again via XOR to produce the ciphertext. Figure 3 illustrates the Even-Mansour construction.

KEY-ALTERNATING CIPHERS. A further generalisation of the Even-Mansour construction are *key-alternating ciphers* that iterates the Even-Mansour construction over multiple rounds. Key-alternating ciphers apply a round function F_i for several rounds alternating with an application of round keys K_i that is added usually with XOR. The round keys K_i are generated by a *key schedule algorithm*, that expands the key K (also called master key) into several round keys K_i. Formally we can denote the key schedule algorithm by a function

$$\mathcal{KS} : \mathbb{F}_2^k \to (\mathbb{F}_2^k)^i, \tag{3}$$

$$K \to (K_0, K_1, \dots, K_{i-1}). \tag{4}$$

Figure 4 illustrates the construction for key-alternating ciphers. Again, in key-alternating ciphers extra whitening keys are added in the begin and the end. The intention of those whitening keys is to restrict control of the inputs/outputs of the cipher in the first and last rounds, to an adversary. This further helps to increase the security against *i.e.,* meet-in-the-middle attacks.
It is fairly easy to construct a secure block cipher that is cryptographically secure, by simply using a large number of rounds. However, a cipher with many rounds will be inefficient. Modern block ciphers have additional design criteria, including performance such as small area, low latency, high throughput, low energy and power consumption and security against side channel attacks. Furthermore, a ci-



(a) Feistel construction **(b)** Substitution Permutation Network **(c)** Lai-Massey Scheme

Figure 5: Common block cipher constructions.

pher should be easy to analyse, meaning that it should be simple to analyse and understand, while the security should depend on the cryptographic keys as outlined by Kerckhoffs' principle in Section 2.2.3. This further helps to determine a recommendation for the number of rounds needed for a block cipher to be secure against cryptographic attacks as outlined in Section 2.4.
In the following, we are describing the most commonly used design strategies for block ciphers, including *Feistel ciphers*, *Substitution Permutation Networks* and *Lai-Massey Ciphers*.

FEISTEL CIPHERS. A Feistel cipher is a symmetric structure for designing block ciphers. Feistel ciphers were first published by Feistel and Coppersmith in IBM's commercial LUCIFER cipher [154], and named after the first author. The most common Feistel cipher is the Data Encryption Standard (DES), which is based on LUCIFER with modifications by the National Security Agency (NSA). A Feistel cipher is an iterated cipher and an internal function called a round function F. The big advantage of a Feistel cipher is that encryption and decryption is very similar or identical in some cases. Thus compared to substitution permutation networks the round function does not have to be invertible. This allows to reuse the same hardware implementation for both encryption and decryption, and further the round keys just have to be used reversed. The Feistel construction is well studied. Luby and Rackoff [234] proved that if the round function in a Feistel cipher is a pseudorandom function, then three rounds are sufficient to consider a Feistel cipher to be a pseudorandom permutation.
We can describe the encryption function as follows. The state is typically halved into two halves of $n/2$ bits each that are called the *left* and *right* halves, that we can denote as (X_i, Y_i). One round consists typically of a round function F that is applied to one of the two halves of the state, together with a round key K_i. The output is then combined with XOR to the other half of the state, and finally the two halves are

swapped. One round of a general Feistel cipher is illustrated in Figure 5a. We can formally describe the state update of a Feistel cipher as follows:

$$X_{i+1} = Y_i \oplus F(X_i, K_i) \tag{5}$$
$$Y_{i+1} = X_i \tag{6}$$

Some examples of commonly used Feistel ciphers are FEAL [250], Twofish [303], Camellia [29], GOST [142], and Simon [46].

SUBSTITUTION PERMUTATION NETWORKS. Substitution Permutation Networks (SPN) are a common approach to design a block cipher. The design approach links cryptographic building blocks together that are specifically selected to fulfil certain criteria (*i.e., confusion, diffusion*) as outlined in the work of Shannon. The round function of an SPN cipher typically consists of the following three building blocks:

- *Substitution Layer:* In general, a Substitution Box (S-box) takes inputs of m bits and transforms then into outputs of n bits. S-boxes can simply be implemented as a lookup tables, where the input is then just substituted with a value in the lookup table. Typically a layer of several smaller S-boxes with 8 bits is used. Lightweight ciphers normally have S-boxes with 4 bits. The aim of an S-box is to achieve confusion, that is to obscure the relationship between the key and the ciphertext. This part is normally the only non-linear operation in a block cipher.
- *Permutation Layer:* The permutation layer in an SPN is normally applied to the whole state. The permutation layer aims to diffuse the state, *i.e.,* if one bit of the plaintext is changed, then statistically half of the bits in the ciphertext should change. A common approach is to combine two steps. First, the state bits are permuted by mapping the bits into new positions. Second, a linear mixing is applied to the state, in which the transformed bits can be noted as a linear combination of the input bits. The best diffusion properties are achieved by MDS matrices. Some lightweight designs further use binary matrices (*i.e.,* Skinny [50], Midori [37]) or just use a simple permutation that can be implemented just by wiring (*i.e.,* Present [95]).
- *Key Addition:* A round key K_i is mixed into the state, typically by using the `XOR` operation.

One round of a general Substitution Permutation Network is illustrated in Figure 5b.
The most common SPN cipher is the Advanced Encryption Standard (AES) [259]. After some flaws have been published against DES in the 1990s [73, 125, 238], the National Institute of Standards and Technology (NIST) announced a cryptographic competition for a new standard in 1997. In 2001, the *Rijndael* block cipher family, designed by Daemen and Rijmen was selected as the winner. The AES is a family of block ciphers consisting of a block size $n = 128$ and key size of $k = 128, 192, 256$. The round function of AES is depicted in Figure 6. At the time of writing, the AES is the most widespread used block cipher, and it is used in many applications. Several hardware vendors and chip designers also have dedicated AES implementations in their chipsets. For example Intel offers hardware optimised implementations in

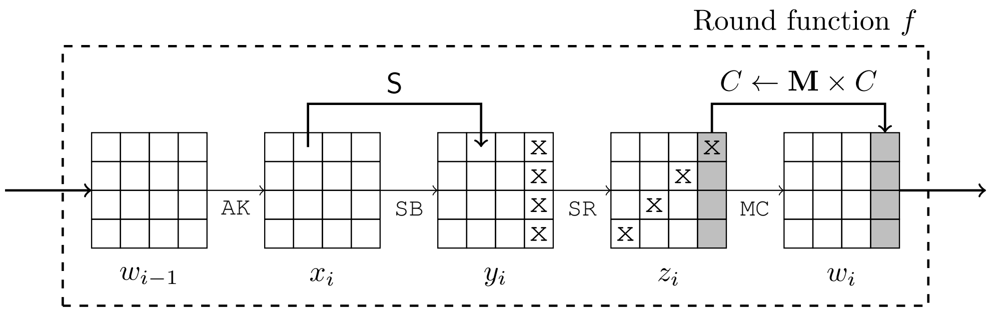


Figure 6: Advanced Encryption Standard (AES).

their AES New Instructions (AES-NI) [165]. The state in AES is arranged in a 4×4 matrix, consisting of one byte per entry. The round function of AES consists of four steps, in the following order:

- `SubBytes`: Each byte is replaced with another one, according to a lookup table. This is the only non-linear operation in AES.
- `ShiftRows`: Each row of the state is cyclically shifted by $i = 0, 1, 2, 3$ steps.
- `MixColumns`: Each column is multiplied by an MDS matrix, where the linear mixing operation combines the four bytes of each column. The MDS matrix is the following:

$$\begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix}.$$

- `AddRoundKey`: Each byte is combined with a round key using the `XOR` operation.

One round of AES is illustrated in Figure 6.

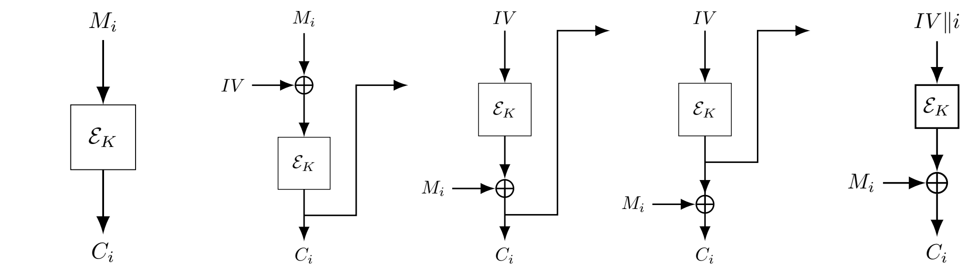
LAI-MASSEY CIPHERS. The Lai-Massey scheme has first been published by Lai and Massey in the design of IDEA [214]. The design closely follows the principles of Feistel ciphers, however with a few differences. Similar as in Feistel ciphers, the round function F does not have to be invertible and the state is split into two words of $n/2$ each. However, contrary to Feistel ciphers, the round function is applied to the difference of both state words. The result of the round function is then added to both state words. Let $F : \mathbb{F}_2^{n/2} \to \mathbb{F}_2^{n/2}$ denote the *round function*, let $H : \mathbb{F}_2^n \to \mathbb{F}_2^n$ denote the *half-round function* and let $K_0, K_1, \ldots, K_n$ be the sub-keys for rounds $0, 1, \ldots, n$. Let (X_i, Y_i) denote the state, which is updated as follows:

$$(X_i', Y_i') = H(X_i, Y_i) \tag{7}$$

$$T_i = F_{K_i}(X_i' \boxminus Y_i') \tag{8}$$

$$(X_{i+1}, Y_{i+1}) = (X_i' \boxplus T_i, Y_i' \boxplus T_i) \tag{9}$$

One round of a general Lai-Massey scheme is illustrated in Figure 5c.



(a) Electronic Codebook (ECB). (b) Cipher Block Chaining (CBC). (c) Cipher Feedback (CFB). (d) Output Feedback (OFB). (e) Counter (CTR).

Figure 7: Block cipher modes of operation.

Modes of Operation

Block ciphers are a fixed mapping of $\mathbb{F}_2^n \to \mathbb{F}_2^n$. However, in real-world applications messages have a length different from the typical block sizes $n = 64, 128$, *i.e.*, we want to encrypt arbitrary length messages $\mathbb{F}_2^* \to \mathbb{F}_2^*$. The solution of this problem are block cipher modes of operation. In a mode, the input message is split into several chunks of the block size n and if necessary padded to be a multiple of n. In the following, the most common block cipher modes are described.

ELECTRONIC CODEBOOK (ECB). In Electronic Codebook mode the message is split into blocks with a block length defined by the underlying block cipher. Each block is then encrypted separately making the mode fully parallel for both encryption and decryption, respectively. Formally, we can denote the encryption of a message block M_i by:

$$C_i = \mathcal{E}_K(M_i) \tag{10}$$

The resulting ciphertext blocks are then concatenated together and form the ciphertext. Electronic Codebook mode has an obvious disadvantage, as there is no interaction with any of the other plaintext/ciphertext blocks. Identical plaintext blocks are encrypted to identical ciphertext blocks leaving it vulnerable to any adversary that searches for patterns in the ciphertexts. It is not recommended to use Electronic Codebook mode in cryptographic protocols. The mode is illustrated in Figure 7a.

CIPHER BLOCK CHAINING (CBC). The Cipher Block Chaining mode was invented by Ehrsam *et al.* [150] in 1976. The mode splits the message into blocks with a block length defined by the underlying block cipher. Each block of plaintext is then combined by XOR with the previous ciphertext block before being feed to the underlying block cipher. This chaining of blocks makes each ciphertext dependent on all plaintext blocks that are processed beforehand. The initial chaining value is replaced by an initialisation vector. Formally, we can denote the encryption of a message block M_i by:

$$C_0 = IV \tag{11}$$
$$C_i = \mathcal{E}_K(M_i \oplus C_{i-1}) \tag{12}$$

As Cipher Block Chaining depends on all previous plaintexts it is not parallelizable in encryption direction, however it is fully parallelizable in decryption direction. Cipher Block Chaining is the most commonly used block cipher mode. The mode is illustrated in Figure 7b.

CIPHER FEEDBACK (CFB). The Cipher Feedback mode is similar to the Cipher Block Chaining mode and allows turning a block cipher into a self-synchronizing stream cipher. Similar as in Cipher Block Chaining mode the message is split into blocks with a block length defined by the underlying block cipher. The i^{th} ciphertext block C_i is then calculated by encrypting the previous ciphertext C_{i-1} and combined with the current message block M_i by XOR. Again, each ciphertext is dependent on the previous message blocks. The initial chaining value is replaced by an initialisation vector. Formally, we can denote the encryption of a message block M_i by:

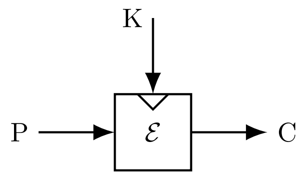
$$C_0 = IV \tag{13}$$
$$C_i = \mathcal{E}_K(C_{i-1}) \oplus M_i \tag{14}$$

Similar as in Cipher Block Chaining mode encryption can not be parallelised, but it is possible to fully parallelise decryption. The straightforward Cipher Feedback mode is not self-synchronizing, as it is just possible to detect transmission errors if a whole block is lost. Losing just a few bits will permanently distort the following blocks. To turn Cipher Feedback mode into a self-synchronizing stream cipher one can use a shift register that allows to re-synchronize after x-bits are lost, where x is the number of bits the shift register is shifting. This is also known as CFB-1 or CFB-8 [145], according to the size of shifting. The mode is illustrated in Figure 7c.

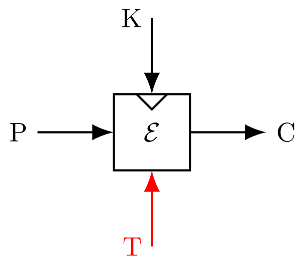
OUTPUT FEEDBACK (OFB). The Output Feedback mode can be used to turn a block cipher into a synchronous stream cipher. The message is split into blocks with a block length defined by the underlying block cipher. A keystream is then computed by initially taking an initialization vector that is feed to an underlying block cipher. The output of the block cipher is then always feed as input to the next block cipher. A ciphertext C_i is computed by combining the message M_i with XOR to the current output of the keystream. Formally, we can denote the encryption of a message block M_i by:

$$C_0 = IV \tag{15}$$
$$C_i = \mathcal{E}_K(C_{i-1} \oplus M_{i-1}) \oplus M_i \tag{16}$$

Because of the symmetry of XOR encryption and decryption is the same. Moreover, as the plaintext/ciphertext is just combined in the final step, the keystream can be calculated in advance and the XOR with the plaintext/ciphertext can be fully parallelised. The mode is illustrated in Figure 7d.



(a) Regular Block Cipher



(b) Tweakable Block Cipher

Figure 8: Regular vs Tweakable Block Ciphers.

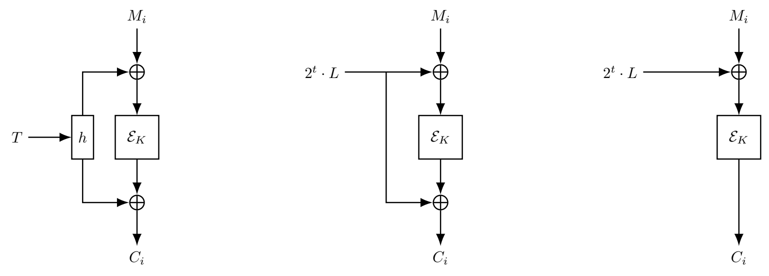
COUNTER (CTR). The Counter mode was published by Diffie and Hellman [133] in 1979. Like Output Feedback mode it can be used to turn block ciphers into a stream cipher. The message is split into blocks with a block length defined by the underlying block cipher. It further uses an initialization vector IV (also called *nonce*) that is concatenated with an counter cnt. The counter should not be repeated, and a simple increment-by-one counter is the most popular. The IV and counter are then fed to the block cipher and the result is combined with each message block M_i to generated ciphertext block C_i. For the next block, the counter is incremented by one. Formally, we can denote the encryption of a message block M_i by:

$$C_i = \mathcal{E}_K(IV \| cnt) \oplus M_i \tag{17}$$

Counter mode is deterministic and allows parallelisation for both encryption and decryption. The mode is illustrated in Figure 7e.

Tweakable Block Ciphers

Tweakable block ciphers have first been published with the HASTY PUDDING cipher [304]. Later, Liskov, Rivest and Wagner [227] generalised the concept of tweakable block ciphers and suggested to move the randomisation of symmetric primitives at protocol level, usually done with a nonce or an initialization vector, to block cipher level. A *block cipher* can be described as a mapping $\mathcal{E} : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$, where n is the block size, and k is the key size of the cipher. The block cipher $\mathcal{E}_K$ then transforms an n-bit plaintext P to an n-bit ciphertext $C = E(K,P)$ using a k-bit key. A *tweakable block cipher* adds another t-bit input T called *tweak*. Formally, we can denote this as $\mathcal{E} : \{0,1\}^k \times \{0,1\}^t \times \{0,1\}^n \to \{0,1\}^n$. The ciphertext is now generated as $C = E(K,T,P)$. Similar as for a regular block cipher $E(K,\cdot)$, a tweakable block cipher $E(K,T,\cdot)$ is a permutation for all $K,T \in \{0,1\}^k \times \{0,1\}^t$. The tweak as additional input is public and provides a family of unrelated block ciphers. A comparison of tweakable block ciphers and regular block ciphers is illustrated in Figure 8.
The application areas of tweakable block ciphers are authenticated encryption modes (*e.g.,* OCB [288], DEOXYS [189]), message authentication codes (*e.g.,* PMAC [86], OMAC [181]), format-preserving encryption algorithms and disk encryption (*e.g.,* XTS).



(a) LRW construction. (b) XEX construction. (c) XE construction.

Figure 9: Modes for Tweakable Block Ciphers.

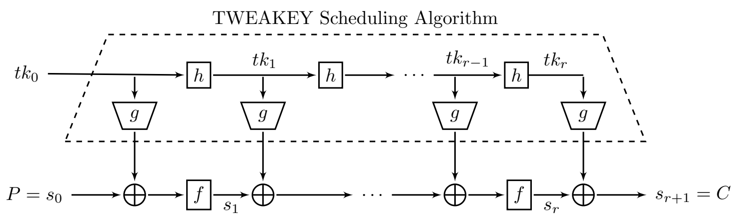


Figure 10: The TWEAKEY construction.

We can construct tweakable block ciphers either as dedicated designs, from block ciphers or from permutations. Dedicated designs, such as SKINNY, JOLTIK-BC, DEOXYS-BC, can be constructed *i.e.,* using the TWEAKEY framework. The XE, XEX [287] and LRW [227] modes allow a designer to turn a block cipher into a tweakable block cipher. Further, Minematsu's TBC [248] mode allows to use a permutation instead of a block cipher. In the following, we describe the first two mentioned constructions in more detail.

THE LRW, XEX AND XE CONSTRUCTIONS. Liskov, Rivest and Wagner [227] generalised the concept of tweakable block ciphers and proposed a mode that turns block ciphers into tweakable block ciphers. The LRW construction uses an XOR-universal hash function h, where the tweak is processed by $h_{k'}(T) = k' \otimes T$, where $\otimes$ represents a field multiplication. The construction is secure up to the birthday bound. The LRW construction is illustrated in Figure 9a.
The XE and XEX construction is designed by Rogaway [287] in 2004. Similar to the LRW construction they are used to turn a block cipher into a tweakable block cipher. The tweak is obtained through a mask, that is calculated from a few constants α, β, γ and the encryption of a nonce N. XE and XEX both achieve security up to the birthday bound. The constructions are illustrated in Figure 9b and Figure 9c.

THE TWEAKEY CONSTRUCTION. From an efficiency point of view, updating the tweak in a tweakable block cipher should be very efficient. This implies that the tweak schedule should be lighter than the key schedule. However, from a security point of view, the tweak is publicly known and fully controllable by an adversary. Thus, this implies that the tweak schedule should be at least as strong as the key

schedule. The TWEAKEY framework [188] tries to tackle this paradox by considering *tweak = key*, or in the notion of the designers *tweakey*. In the TWEAKEY framework, the regular key schedule is replaced by an *tweakey* schedule, that instead of round keys generates round tweakeys. An tweakable block cipher using an n-bit key and an n-bit tweak, has a $2n$-bit tweakey. The TWEAKEY framework has two functions, h is the tweakey update function that updates the tweakey material round by round in a linear fashion. The g function is the tweakey extraction function, that extracts the round tweakeys and compresses the tweakey from $2n$-bits to n-bits. The TWEAKEY construction is illustrated in Figure 10.
Yet, simply using the same tweak each round for the whole block size decreases the security [187]. Therefore, the designers of TWEAKEY proposed the *Superposition-*TWEAKEY (STK) construction. The idea is to separate the tweakey material in several words and then use the tweakey schedule for one word, and superpose it in a secure way. To turn the TWEAKEY framework into the STK construction the tweakey update function h should consist of the same permutation each round and is followed by a multiplication of each nibble with a value $\alpha_k \in GF(2^c)$ that is different for each tweakey word. Moreover, the round tweakey extraction function g consist of combining each tweakey word by XOR to compress them to n-bit. We give some detailed insights in the security of the TWEAKEY framework in Chapter 5.

2.3.2 Stream Ciphers

A stream cipher is a symmetric-key algorithm that combines plaintext bits with a pseudorandom stream of key bits, also called keystream. In comparison to block ciphers, that encrypt blocks of fixed length at a time, stream ciphers encrypt single bits at a time. This is done by considering an arbitrary message of length $|M|$ and combining it, normally using the XOR operation, with a keystream of the same length as the message. We can define a stream cipher as follows:

Definition 2. Let k, iv be two positive integers, then a stream cipher $\mathcal{S}$ is given by

$$\begin{aligned}\mathcal{S}(K, IV, M) : \mathbb{F}_2^k \times \mathbb{F}_2^{iv} \times \mathbb{F}_2^* &\to \mathbb{F}_2^* \\ (K, IV, M) &\to C,\end{aligned}$$

where $K \in \mathbb{F}_2^k$ is a key of fixed length k and $IV \in \mathbb{F}_2^{iv}$ is an initialisation vector or seed for the stream cipher of fixed length iv. The stream cipher generates a pseudorandom keystream that can then be combined with the message M to generate the ciphertext C.

In general, a stream cipher can be seen as the one-time pad (OTP), or also known as Vernam cipher. The one-time pad computes the ciphertext $C = M \oplus K$, where $|C| = |M| = |K|$. It was proven to be secure by Shannon in 1949 [308], but its security requires that the keystream is never repeated, and the keystream is chosen completely at random. While this is ideal in a theoretical sense, it is usually infeasible to achieve in a practical application. To circumvent this problem, stream ciphers generate a pseudorandom keystream from a random seed value, which depends on the secret key with a convenient key size, such as *i.e.,* 128-bits.
There exist two types of stream ciphers, synchronous and self-synchronising stream ciphers that can be distinguished by how the keystream is generated. In the former

Table 2: The final portfolio of the eSTREAM competition.

Profile 1 (software)	Profile 2 (hardware)
HC-128 [350]	Grain [170]
Rabbit [91]	MICKEY [35]
Salsa 20/12 [58]	Trivium [129]
SOSEMANUK [56]	

one, the keystream is generated independent of the plaintext and ciphertext and is then combined with the plaintext or ciphertext to encrypt or decrypt, respectively. In the later one, parts of the ciphertext are used to compute the keystream. This allows a receiver to synchronise with the keystream generator, allowing for easier recovery of possible dropped ciphertext bits.
The security of a stream cipher mainly depends on the keystream. The keystream must have a large period and should be free of any biases that would let an adversary be able to distinguish the output from random. Furthermore, the seed to generate the keystream should not be reused. It should also be infeasible for an adversary to recover the internal state or the key of the stream cipher from the keystream as the adversary otherwise could continue generating its own keystream. Stream ciphers are often used in applications with an unknown length of the message input. The advantage of stream ciphers is that they can encrypt individual message bits on-the-fly, which can be advantageous in real-time systems. Using a block cipher smaller messages than the block size need to be padded, resulting in a huge overhead in both the transmission and padding of the messages. Stream ciphers are typically used in the security protocols for wireless networks, such as WEP, WPA and also in TLS.
The most relevant stream ciphers in practical use-cases are ChaCha [59], that will be used also in TLS 1.3, RC-4 [28] which has been widely used until TLS 1.2 and A5/1 [104], A5/2 [104] that are used in mobile phone networks. As of September 2011, the European Union's ECRYPT network announced a portfolio of stream ciphers that had been submitted to a stream cipher competition, called eSTREAM. Those ciphers are split in two use-cases, Profile 1 for software-optimised stream ciphers and Profile 2 for hardware-optimised stream ciphers. The eSTREAM portfolio includes the following algorithms outlined in Table 2.

2.3.3 Cryptographic Hash Functions

A cryptographic hash function maps arbitrary-length input data to a fixed-length output string, also called *hash* or *digest*. It is designed to be an one-way function, that is, a function which is infeasible to invert. Cryptographic hash functions have many applications in information-security to provide integrity of data. These include digital signature, message authentication codes, checksums, storage of passwords, crypto-currencies, and the generation of pseudo-random numbers. An ideal hash function achieves the following main properties:

- *Deterministic:* the same message should always result in the same hash.
- *Efficient:* it should be fast to compute the hash for a given message.
- *One-way:* it should be infeasible to generate the message from a given hash value.
- *Secure:* even a small change to the message should result in a complete change of the hash value, uncorrelated to the old hash value.
- *Unique:* it should be infeasible to find two messages that result in the same hash value.

We can formally define a hash function as follows:

Definition 3. Let n be a positive integer, then a cryptographic hash function $\mathcal{H}$ is given by

$$\begin{aligned} \mathcal{H}(M) : \mathbb{F}_2^* &\to \mathbb{F}_2^n \\ M &\to \tau, \end{aligned}$$

The security of a cryptographic hash function depends on three main properties defined as follows:

- *Collision resistance:* It should be infeasible to find two messages m_1 and m_2 that result in the same hash value, *i.e.,* $\mathrm{hash}(m_1) = \mathrm{hash}(m_2)$, where $m_1 \neq m_2$. If such a message pair can be found it is called a cryptographic hash collision. Due to the birthday paradox, collision attacks can occur with a complexity of $2^{n/2}$.
- *Pre-image resistance:* It should be infeasible to find any message m, that is the pre-image of a given hash value $\tau = \mathrm{hash}(m)$. This property relates to the fact that a hash function should represent an one-way function. Pre-image resistance is given up to a complexity of 2^n.
- *Second pre-image resistance:* Given an arbitrary message m_1, it should be infeasible that an adversary can find a different message m_2, that results in the same hash value, *i.e.,* such that $\mathrm{hash}(m_1) = \mathrm{hash}(m_2)$ but $m_1 \neq m_2$. Second pre-image resistance is given up to a complexity of 2^n. Moreover, if a hash function is collision resistant, this implies that it is also second pre-image resistant. However, this does not hold for pre-image resistance.

The Merkle-Damgård Construction

The Merkle-Damgård construction is a common way to use one-way compression functions inside a cryptographic hash function. The construction has been independently published by Merkle [246] and Damgård [123] in 1989. It is widely used in the MD-5 [283], SHA-1 [148] and SHA-2 [260] families.
Hash functions process arbitrary length messages into a fixed-length output, also called digest. The Merkle-Damgård construction splits the message in several blocks of a fixed block size and then operates on them in sequence feeding the messages to an one-way compression function. This compression function is either specially designed, as it is the case for *i.e.,* MD-5, SHA-1 and SHA-2, or it is based on an

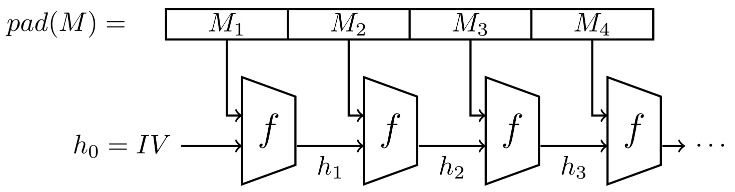
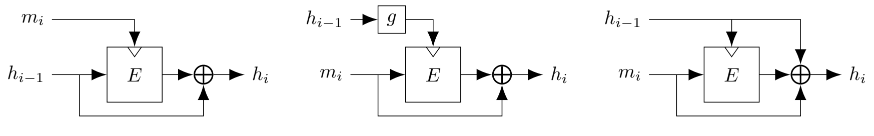


Figure 11: The Merkle-Damgård construction.



(a) Davis-Meyer (b) Matyas-Meyer-Oseas (c) Miyaguchi-Preneel

Figure 12: Modes for hash functions based on block ciphers.

underlying block cipher. The last block also has to be length padded, which is crucial to the security of the Merkle-Damgård construction. We can formally define the functioning of the Merkle-Damgård construction as follows. A compression function F is applied iteratively to the chaining value H_i, for $1 \leqslant i \leqslant l+1$ together with the message input M_i, to obtain the next chaining value H_{i+1}. The initial chaining value $H_0 = IV$ is replaced by an initialization vector. In the end, we obtain H_{l+1} that is the hash digest of our hash function $\mathcal{H}$. Figure 11 illustrates the Merkle-Damgård construction.

Constructions from Block Ciphers

One-way compression functions can be build using block ciphers. A block cipher is a mapping of two fixed-length inputs, *i.e.,* the plaintext and the key, to a fixed-length output ciphertext. A normal block cipher is just partially valid as a one-way compression function, as on the one hand, it is hard to find a key that encrypts the plaintext to the ciphertext but on the other hand, given a ciphertext and a key the plaintext can be recovered by using the decryption algorithm. Nevertheless, there are some modes, Davies–Meyer, Matyas–Meyer–Oseas and Miyaguchi–Preneel, that can be used to turn block ciphers into one-way compression functions.

DAVIES-MEYER CONSTRUCTION. The construction has been first published by Winternitz [348, 349] and is attributed to Davies and further it has been published by Davies and Price [126] where it has been attributed to Meyer. The construction splits the input message in parts of size κ and feeds the message blocks as key to an underlying block cipher. The previous hash value H_{i-1} is used as plaintext for the block cipher. The output ciphertext is then XORed with the previous hash value H_{i-1} to create a new hash value H_i. In the initial round, the previous hash value is replaced by a constant initialisation vector H_0. The construction is illustrated in Figure 12a. We can denote one iteration as:

$$H_i = E_{m_i}(H_{i-1}) \oplus H_{i-1}, \quad 1 \leqslant i \leqslant l. \tag{18}$$

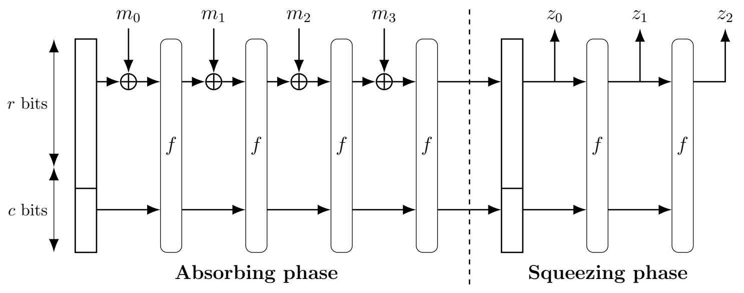


Figure 13: Sponge construction.

The length of the final hash value $\tau = H_l$ is the block size n of the underlying block cipher. The Davis-Meyer construction has one fixed point, that occurs independent of the underlying block ciphers. For any message m, one can find a hash value h such that $E_m(h) \oplus h = h$, by choosing $h = E_m^{-1}(0)$. However, so far no practical attack has been published, based on this fixed point.

MATYAS-MEYER-OSEAS CONSTRUCTION. The construction has first been published by Matyas, Meyer and Oseas [241] in 1985. It can be seen as the dual to the Davies-Meyer construction, as it splits the message in blocks and processes them as the plaintexts to the underlying block cipher. The previous hash value H_{i-1} is used as key to the underlying block cipher. The next hash value H_i is then generated by combining the output of the block cipher with XOR to the input message block. In the initial round, the previous hash value is replaced by a constant initialisation vector H_0. The construction is illustrated in Figure 12b. We can denote one iteration as:

$$H_i = E_{g(H_{i-1})}(m_i) \oplus m_i, \quad 1 \leqslant i \leqslant l, \tag{19}$$

where $g : \mathbb{F}_2^n \to \mathbb{F}_2^\kappa$ is a function that compresses the previous hash value from blocksize n to the key size κ of the underlying block cipher.

MIYAGUCHI-PRENEEL CONSTRUCTION. The construction has been independently proposed by Miyaguchi *et al.* [251] and Preneel *et al.* [278]. The Miyaguchi-Preneel construction is an extended variant of the Matyas-Meyer-Oseas construction. Similar to Matyas-Meyer-Oseas the message is split in blocks that are processed as plaintext to the underlying block cipher. The next hash value H_i is generated by combining the output of the block cipher with XOR to the input message block and with the previous hash value H_{i-1}. The construction is illustrated in Figure 12c. We can denote one iteration as:

$$H_i = E_{g(H_{i-1})}(m_i) \oplus H_{i-1} \oplus m_i, \quad 1 \leqslant i \leqslant l, \tag{20}$$

where $g : \mathbb{F}_2^n \to \mathbb{F}_2^\kappa$ is again a function that compresses the previous hash value from blocksize n to the key size κ of the underlying block cipher.

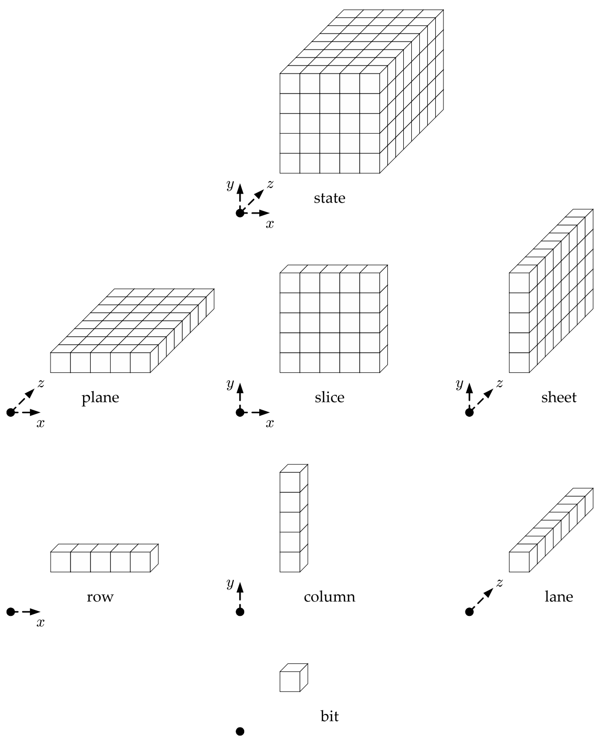


Figure 14: State of KECCAK [5]

Sponge Construction

The Sponge construction is a function that maps variable-length inputs to variable-length outputs. It can be seen as a generalisation of both a hash function and a stream cipher, where the former has a variable-length output and the later has a variable-length input, respectively. The sponge construction is a simple iterated construction based on a state memory, that is divided into two sections: the bitrate r and the capacity c, and further a fixed-length permutation P and a padding rule. The construction works as follows, as depicted in Figure 13. First, the input message is padded with a reversible padding rule. Next, the input message is split into blocks of size r. Then the internal state memory is initialized to zero and the sponge construction continues in two phases:

- *Absorbing Phase:* in the absorbing phase, the r-bit input blocks are combined by XOR to the state memory, interleaved by applications of the permutation P, over the whole state memory $r + c$.
- *Squeezing Phase:* in the squeezing phase, the first r-bits are returned as output blocks, while the state memory is still interleaved with applications to the permutation P.

Note that the capacity is never directly affected by any input or output block in both the absorbing and squeezing phase.

[5] Figure source: `https://keccak.team/figures.html`

Table 3: Instances of KECCAK for SHA-3 and SHAKE by NIST.

Instance	Digest length	Rate r	Capacity c	Security Strength		
				Collision	Preimage	Second-Preimage
SHA-3-224(M)	224	1152	448	112	224	224
SHA-3-256(M)	256	1088	512	128	256	256
SHA-3-384(M)	384	832	768	192	384	384
SHA-3-512(M)	512	576	1024	256	512	512
SHAKE-128(M, d)	d	1344	256	$\min(d/2, 128)$	$\geqslant \min(d, 128)$	$\min(d, 128)$
SHAKE-256(M, d)	d	1088	512	$\min(d/2, 256)$	$\geqslant \min(d, 256)$	$\min(d, 256)$

The sponge construction can be used to construct several cryptographic primitives such as block ciphers, stream ciphers, hash functions, message authentication codes and authenticated encryption schemes. In the following, we focus on sponge constructions when used as hash algorithms. The Secure Hash Algorithm 1 (SHA-1) and the family of Secure Hash Algorithm 2 (SHA-2) are two commonly used hash functions based on the Merkle-Damgård construction, that has been standardised by the National Institute of Standards and Technology (NIST). In 2007, NIST decided that SHA-2 needs to be replaced and organised a public competition [262] for a new hash standard, SHA-3. The winner of the SHA-3 competition was KECCAK [61], which is based on the sponge construction. KECCAK uses a block transformation function f, which is a permutation that uses only the logic operations `XOR`, `AND` and `NOT`. It is designed for simple implementations in both hardware and software. The block transformation function f consists of five steps that operated on the three-dimensional state, which is organised as a $5 \times 5 \times w$ array, where $w = 64$-bits is the word size. The state of KECCAK is illustrated in Figure 14. The steps of the block transformation function f are the following:

- θ*:* is a linear diffusion step. The parity of each column in a slices is calculated and added to a neighbouring column in the same and next higher slice.
- ρ*:* is a linear diffusion step that bitwise rotates the words by a triangular number.
- π*:* is a linear diffusion step that permutes the words in a fixed pattern.
- ξ*:* is the only non-linear function. It is a mapping of degree 2 that operates on each row of the state independently and can be described as $x_i \leftarrow x_i \oplus (\neg y_i \& z_i)$, where x, y, z are rows of the state.
- ι*:* is a simple `XOR` of a round constant RC_i to a lane to break symmetries.

NIST defined six instances of KECCAK as SHA-3 and SHAKE [261] for different message lengths M and output lengths d, as denoted in Table 3.

2.3.4 Message Authentication Codes

Message Authentication Codes (MAC) are used to authenticate a message, *i.e.,* to confirm that the message has not been altered and that the message is indeed from the intended sender. The message authentication code therefore produces a tag that can be sent with the message, and be verified by the receiver. In this context the message authentication code protects the data integrity of a message and ensures the authenticity of the sender by allowing a receiver (who is also in possession of the secret key) to verify that the message was sent by a sender who is also in possession of the secret key. We can formally define a Message Authentication Code as follows:

Definition 4. Let k, n be two positive integers, then a *Message Authentication Code (MAC)* $\mathcal{M}$ is given by

$$\text{MAC } \mathcal{M}_K(\cdot) : \mathbb{F}_2^k \times \mathbb{F}_2^* \to \mathbb{F}_2^n$$

where $K \in \mathbb{F}_2^k$ is a key of fixed length k. The Message Authentication Code consists of three algorithms ($\mathcal{M}.\mathcal{G}$, $\mathcal{M}.\mathcal{S}$, $\mathcal{M}.\mathcal{V}$):

- *Key-Generation* $\mathcal{M}.\mathcal{G}$*:* generates a key $K \in \mathbb{F}_2^k$ of length k.
- *Signing* $\mathcal{M}.\mathcal{S}$*:* generates a tag τ based on the key K and an arbitrary length message M.
- *Verification* $\mathcal{M}.\mathcal{V}$*:* returns either $\begin{cases} \textit{accepted} \text{ if the tag } \tau \text{ is a valid tag,} \\ \textit{rejected} \text{ or } \perp \text{ if the tag } \tau \text{ is not valid,} \end{cases}$.

where a valid tag can only be computed by a sender with possession of the secret key K.

While Message Authentication Codes are similar to cryptographic hash functions they have different security requirements. For a Message Authentication Code to be considered secure it must resist any existential forgery under chosen-plaintext attacks. This can be defined formally in the notation of EUF-CMA according to Bellare *et al.* [52].

Definition 5 (EUF-CMA)**.** A Message Authentication Code (MAC) is considered unforgable if for any efficient adversary $\mathcal{A}$

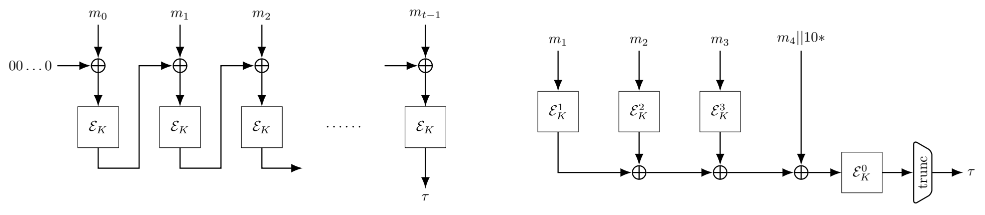
$$\Pr\left[\begin{array}{c} k \to \mathcal{M}.\mathcal{G}(1^n), \\ (x,t) \to \mathcal{A}^{\mathcal{M}.\mathcal{S}(k,\cdot)(1^n)}, x \notin \text{Query}(\mathcal{A}^{\mathcal{M}.\mathcal{S}(k,\cdot)(1^n)}) \\ \mathcal{M}.\mathcal{V}(k,x,t) = \text{accepted} \end{array}\right] = \epsilon \tag{21}$$

An adversary which makes q_m distinct calls to $\mathcal{M}.\mathcal{S}(k,\cdot)$, q_v distinct calls to $\mathcal{M}.\mathcal{V}(k,\cdot,t)$, runs in time t and wins the game with probability at least ϵ is called a (t, q_m, q_v, ϵ)-adversary. A MAC is considered (t, q_m, q_v, ϵ)-secure if there exists no (t, q_m, q_v, ϵ)-adversary.

Message Authentication Codes are used in various applications. These include electronic signatures, payment schemes, commitment schemes and many internet protocols such as TLS/SSH.

Message Authentication Codes from Block Ciphers

Message Authentication Codes can be constructed from other cryptographic primitives, *i.e.,* from block ciphers. In the following, we give an overview of some of the most relevant constructions.



(a) Cipher-Block-Chaining Message Authentication Code (CBC-MAC).

(b) Parallelizable Message Authentication Code (PMAC).

Figure 15: Message Authentication Codes from Block Ciphers.

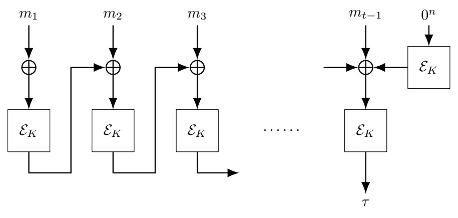
CIPHER-BLOCK-CHAINING MESSAGE AUTHENTICATION CODE (CBC-MAC). CBC-MAC is a MAC that is based on block ciphers in CBC-Mode. The message is encrypted by the block cipher in CBC-mode and instead of producing ciphertexts, the outputs of the block cipher are chained together and are combined with the message inputs using XOR and then fed as input to the next block cipher. The initialisation vector is set to zero. The output of the last block cipher call is then used as the tag τ.

CBC-MAC is secure for fixed-length messages, if the underlying block cipher is secure [53]. However, for variable-length messages CBC-MAC is insecure. This is because, if an adversary can construct two message/tag pairs (m, τ) and (m', τ'), she can generate a new valid message $m'' = m||(m'_1 \oplus \tau)||m_2||\ldots||m'_n$ that will result in tag τ'. This issue can be resolved by either prepending the length of the message in the first message block, or if the message length is unknown by encrypting the last block a second time with a different key, which is also defined as *Encrypt-last-block* CBC-MAC (ECBC-MAC). CBC-MAC is illustrated in Figure 15a.

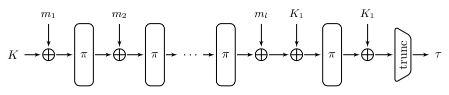
PARALLELIZABLE MESSAGE AUTHENTICATION CODE (PMAC). PMAC addresses the problem that CBC-MAC is inherently serial. Black and Rogaway [86] proposed PMAC that is fully parallelizable, while it is still as efficient as CBC-MAC in serial environments. Parallelism is also highly desirable on modern high-end CPUs, as in dedicated hardware the speed is limited either by the latency or by the amount of parallelism that can be utilised from the underlying block cipher.

PMAC computes the tag by first combining each message block and a mask with XOR. The result is then fed to an underlying block cipher and the outputs of the block cipher are then combined by XOR. Finally, the tag τ is generated by another application of a block cipher. PMAC is illustrated in Figure 15b.

ONE-KEY MESSAGE AUTHENTICATION CODE (OMAC) AND CIPHER-BASED MESSAGE AUTHENTICATION CODE (CMAC). Black and Rogaway [85] initially proposed XCBC, a variant of CBC-MAC that solves the issues with CBC-MAC. However, the variant requires three independent keys. Consequently, Iwata and Kurosawa [181] proposed an improvement called One-Key MAC (OMAC), that required just one key. Iwata and Kurosawa further improved OMAC, and renamed the original proposal to OMAC2, while the new version was called OMAC1. In 2005, CMAC [182] that



(a) Cipher-based Message Authentication Code (CMAC).



(b) Chaskey.

Figure 16: Message Authentication Codes from Block Ciphers.

is identical to OMAC1 became a recommendation by the National Institute of Standards and Technology (NIST).
The tag is computed similar as in CBC-MAC. The main differences are that no initialisation vector is used, and the last block is handled differently. The message blocks are still chained with the outputs of the previous block cipher. For the last block m_n, a temporary value $k_0 = E_k(0)$ is calculated, then two subkeys k_1, k_2 are derived. Depending if the last block is a complete block, the last block is either combined by XOR with k_1 or if it is an incomplete block, the block is padded with $10\ldots0_2$ and then combined by XOR with k_2. Finally, the tag value is generated by applying the XOR of the previous chaining value and the special last block m_n to the underlying block cipher and the l highest bits are used as tag τ. CMAC is illustrated in Figure 16a.

POLY1305. Poly1305 is a message authentication code designed by Bernstein [57] that is based on the evaluation of a polynomial over a finite field. The advantage of this approach is that short keys can be used, key generation is fast and the message authentication can be very efficient. Poly1305 is recommended to be used in combination with AES and a tag is generated by:

$$\begin{aligned} \tau &= \text{Poly}_r(m, \text{AES}_k(n)) && (22)\\ &= (((c_1 r^q + c_2 r^{q-1} + \cdots + c_q r^1) \bmod 2^{130} - 5) + \text{AES}_k(n)) \bmod 2^{128} && (23) \end{aligned}$$

where n is a nonce under the secret key (k, r), and

$$c_i = m[16i - 16] + 2^8 m[16i - 15] + 2^{16} m[16i - 14] + \cdots + 2^{120} m[16i - 1] + 2^{128} \quad (24)$$

are integers that are computed from the message m into a polynomial that is then reduced modulo the prime $2^{130} - 5$ in Poly1305.

CHASKEY. Mouha *et al.* [255] published a very efficient message authentication code that is designed for microcontrollers, called Chaskey. It is intended for applications that still require a security of 128-bit, but need to be implemented in a constrained environment that does not allow the use of any standard MAC algorithm.
Chaskey is based on a 128-bit block cipher that uses ARX operations for efficiency. A MAC tag is computed by first splitting the message in l blocks of 128 bits each.

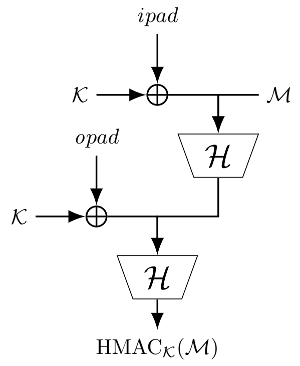


Figure 17: Hash-based Message Authentication Code (HMAC).

The messages are then combined by XOR with the state and interleaved with an application to a permutation π. The initial state is set to the key K. Furthermore, the key K is used to derive two subkeys K_1 and K_2, that are combined with the state after the message is absorbed into the state. Finally, the tag τ is generated by truncating the state to t-bits. CHASKEY is illustrated in Figure 16b.

Message Authentication Codes from Hash Functions

Message authentication codes can further be constructed by using hash functions. The most prominent example is HMAC.

HASH-BASED MESSAGE AUTHENTICATION CODE (HMAC). Hash-based Message Authentication Code (HMAC) was published by Bellare *et al.* [51] in 1996. It involves a cryptographic hash function, such as MD-5, SHA-1 or SHA-2, and a secret key K. The design of HMAC was motivated with the existence of *length-extension attacks* [136]. This occurs when a tag is calculated with a hash function by concatenation the key and the message $\tau = \mathcal{H}(key||message)$. With many hash functions it is possible to append data to the message without knowing the key and one can then create a valid tag value τ.
HMAC initially derives two subkeys k_i, k_o from the secret key. Then it calls a hash function two times, while in the first pass it produces an internal hash derived from the message and the inner key k_i. In the second pass to the hash function, the internal hash and the outer key k_o are used and a final tag τ is produced. Formally, we can define HMAC as:

$$\mathrm{HMAC}(K, M) = \mathcal{H}((K' \oplus opad)||\mathcal{H}((K' \oplus ipad)||M)) \tag{25}$$

where $opad = 0x5C5C\ldots5C$ and $ipad = 0x3636\ldots36$ are the outer and inner padding. HMAC is illustrated in Figure 17.

2.3.5 Authenticated Encryption

Authenticated Encryption (AE) and Authenticated Encryption with Associated Data (AEAD) are a special form of encryption that simultaneously provides message pri-

vacy (*i.e.,* confidentiality) and integrity (*i.e.,* data authenticity). The need for AE schemes emerged when encryption is used in *online* applications such as TLS/SSL, where an adversary can intercept and tamper with ciphertexts. Securing such systems would require the secure combination of cryptographic primitives that provide confidentiality and authenticity. The simplest way to protect against active adversaries is to combine a block cipher that provides confidentiality with a message authentication code that provides integrity of the data. Yet, the generic composition, the combination of a confidentiality mode with an authentication mode, may lead to implementation errors and is often not efficient (*i.e.,* the input stream has to be processed twice, for the encryption scheme and the message authentication code). Recently, some practical attacks [110, 332] have confirmed the vulnerability of these AE schemes. Currently, two block cipher modes are widespread used to provide authenticated encryption. The TLS 1.2 cipher suite includes *Galois/Counter Mode (GCM)* [147] and *Counter with* CBC-MAC *(CCM)* [146] using AES as the underlying block cipher. Moreover, in 2013 the *Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR)* [60] was announced to overcome the recent critical attacks and to achieve fast and efficient AE schemes that should be suitable for widespread adoption.
We can formally define an authenticated encryption scheme with associated data as follows:

Definition 6. Let κ, η, τ be three positive integers, then an *authenticated encryption scheme with associated data (AEAD)* is given by the following three algorithms:

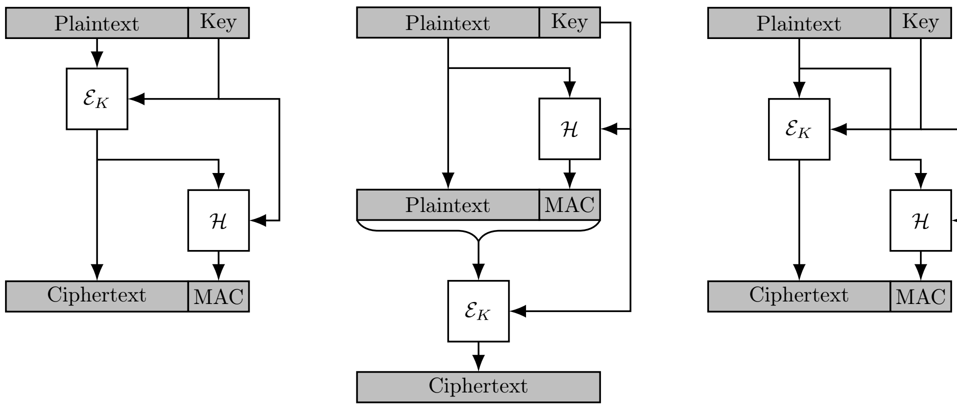
- *Key-Derivation* $\mathsf{AEAD}.\mathcal{K}$*:* that returns a key $K \in \mathbb{F}_2^\kappa$ where $K \overset{\$}{\leftarrow} \mathcal{K}$ returns a randomly sampled key from the keyspace.
- *Encryption* $\mathsf{AEAD}.\mathcal{E}$*:* is the encryption function which is specified as

$$\begin{aligned} \mathcal{E} : \mathbb{F}_2^\kappa \times \mathbb{F}_2^\eta \times \mathbb{F}_2^* \times \mathbb{F}_2^* &\to \mathbb{F}_2^* \times \mathbb{F}_2^\tau && (26) \\ (K, N, A, M) &\to (C, T), && (27) \end{aligned}$$

  where K is the secret key, N is an η-bit nonce, A is the associated data of arbitrary length, which is authenticated but not encrypted, and M is the message, again of arbitrary length. The nonce N is a value that must not be repeated for a fixed key K. The encryption function $\mathcal{E}$ produces a ciphertext C of arbitrary length and a τ-bit tag T.
- *Decryption* $\mathsf{AEAD}.\mathcal{D}$*:* is the decryption function which is specified as

$$\begin{aligned} \mathcal{D} : \mathbb{F}_2^\kappa \times \mathbb{F}_2^\eta \times \mathbb{F}_2^* \times \mathbb{F}_2^* \times \mathbb{F}_2^\tau &\to \mathbb{F}_2^* \cup \{\perp\} && (28) \\ (K, N, A, C, T) &\to \begin{cases} M, & T = T' \\ \perp, & T \neq T' \end{cases}, && (29) \end{aligned}$$

  where K, N, A, C, T are the same as in the encryption algorithm, and T' is the tag that the verifier computes by the decryption algorithm. If the $\perp$ symbol is returned, the verifier can reject the message as a verification error is detected.



(a) Encrypt-then-MAC. (b) MAC-then-Encrypt. (c) Encrypt-and-MAC.

Figure 18: Generic Composition for Authenticated Ciphers.

Constructions for Authenticated Encryption

In the following, we describe three generic compositions of a block cipher and a message authentication code to construct an AE scheme. These construction can simply be changed to AEAD schemes by adding an additional associated data A input, however, for simplicity we ignore the associated data in the following description.

ENCRYPT-THEN-MAC. In the Encrypt-then-MAC mode, the plaintext is first encrypted by a block cipher $\mathcal{E}$ to produce the ciphertext. Then the ciphertext is fed to a message authentication code, and the MAC tag τ is produced. The ciphertext and the MAC tag are then sent together to the verifier. Formally this can be notated as following; First, the sender computes the ciphertext $\mathcal{C}$ and MAC tag τ:

$$C||\tau = \mathcal{E}_K(M)||MAC_K(\mathcal{E}_K(M)). \tag{30}$$

The verifier then first computes:

$$\tau' = MAC_K(C). \tag{31}$$

and checks if $\tau = \tau'$. If the tags match, the verify proceeds to compute

$$M = \mathcal{D}_K(C). \tag{32}$$

to obtain the message M. Figure 18a illustrates the encryption and tag generation in Encrypt-then-MAC. Encrypt-then-MAC is used in *i.e.*, IPsec [307] and TLS 1.2 [282]. Bellare and Namprempre [55] proved that Encrypt-then-MAC is the only generic construction that satisfies all conditions to be considered a secure AE scheme, under the assumption that the message authentication code achieves *strong unforgability*. Encrypt-then-MAC is further highly efficient, as the MAC tag τ can be verified *before* decryption. This allows for faster verification, if an adversary attempts to forge messages, as the ciphertext can be rejected without spending additional effort on decryption.

ENCRYPT-AND-MAC. In the Encrypt-and-MAC mode, the encryption and tag generation are completely separated. The sender first computes the ciphertext C using a block cipher $\mathcal{E}$. The MAC tag τ can be computed in parallel using a message authentication code MAC. The ciphertext C and MAC tag τ are then concatenated and send to the verifier. Formally this can be notated as following; The ciphertext and tag are computed by:

$$C\|\tau = \mathcal{E}_K(M)\|MAC_K(M). \tag{33}$$

and sent to the verifier. The verifier can then compute

$$M = \mathcal{D}_K(C) \text{ and } \tau' = MAC_K(M) \tag{34}$$

and the message M is accepted if and only if the tags match, *i.e.,* $\tau = \tau'$. Figure 18c illustrates the encryption and tag generation in Encrypt-and-MAC. Encrypt-and-MAC is used in SSH [233]. The ciphertext is not protected by the message authentication code and the MAC tag τ' can only be verified after decryption of the ciphertext. Bellare *et al.* [54] pointed out security issues and fixes for the Binary Packet Protocol (BPP) in SSH.

MAC-THEN-ENCRYPT. In the MAC-then-Encrypt mode, the MAC tag τ is first computed by a message authentication code MAC. The message and the tag τ are then fed to a block cipher $\mathcal{E}$ to obtain the ciphertext C. The ciphertext and the tag are then concatenated and sent to the verifier. Formally this can be notated as following; First the sender computes the MAC tag τ by

$$\tau = MAC_K(M) \tag{35}$$

and then computes the ciphertext C by

$$C = \mathcal{E}_K(M\|\tau) \tag{36}$$

and sends the concatenation of the ciphertext C and MAC tag τ to the verifier. The verifier can then decrypt the ciphertext C and check the MAC tag τ' by

$$M'\|\tau = \mathcal{D}_K(C) \text{ and } \tau' = MAC_K(M'). \tag{37}$$

The verifier accepts the message M if the MAC tags $\tau = \tau'$ match. Figure 18b illustrates the encryption and tag generation in MAC-then-Encrypt. MAC-then-Encrypt is used widespread in SSL [157], TLS [282] and DTLS [149]. A major drawback in comparison to Encrypt-then-MAC is that the whole message needs to be decrypted before the tag can be verified. An adversary can use this *i.e.,* in Denial of Service (DoS) attacks by flooding the verifier with wrong message/tag pairs. An even more critical vulnerability is that the tag is part of the plaintext for the block cipher. This attack is known as *padding oracle attacks* [332], when the encryption uses a block cipher in CBC-mode. This attacks have been further exploited in the recent *Lucky Thirteen attack* [8] on TLS and DTLS and *POODLE* [252] on SSL 3.0, that also utilises a *downgrade attack* by forcing an server to use SSL 3.0.

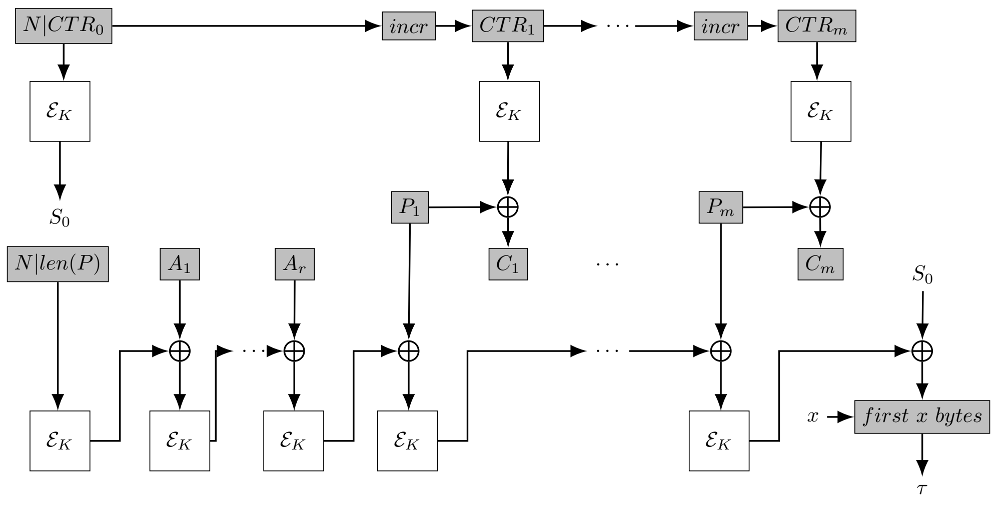


Figure 19: Counter with CBC-MAC (CCM).

Counter with CBC-MAC (CCM)

CCM was designed by Whiting *et al.* [347] and is a widely used mode for authenticated encryption. The application of CCM reaches from the IEEE 802.11i standard, to IPSec [335], TLS 1.2 [296] as well as for Bluetooth Low Energy (*i.e.,* Bluetooth 4.0). The motivation behind the design of CCM was the submission of OCB mode for the IEEE 802.11i standard. OCB in comparison to CCM had a patent pending on the algorithm.
CCM combines the use of counter mode encryption (CTR) with the well-known CBC-MAC for authentication. The primitives are combined in a MAC-then-Encrypt style, where first a MAC tag τ is obtained by processing the message with CBC-MAC, and then the message together with the tag τ are encrypted using counter mode. Compared to MAC-then-Encrypt, CCM makes use of just one key, for both the message authentication code and the encryption part. Figure 19 illustrates CCM. From a performance perspective CCM is two-pass, meaning that the message needs to be processed by the message authentication code, and then a second time for the encryption part. Therefore, it lacks performance in comparison to one-pass modes like GCM or OCB. From a security perspective, Jonsson [191] proved CCM to be secure up to the birthday bound. Rogaway and Wagner [286] published some issues of CCM mainly concerning performance, including that CCM is not online, it disrupts word-alignment, and further CCM can not pre-process static associated data. Apart from that, no serious security issues are published.

Galois/Counter Mode (GCM)

GCM was designed by McGrew and Viega [243] and is currently the most common AEAD block cipher mode of operation. It has been widely adopted because of its efficiency and performance. GCM in general is defined for 128-bit block ciphers, but it is mostly used in combination with AES-128. GCM is used in many applications

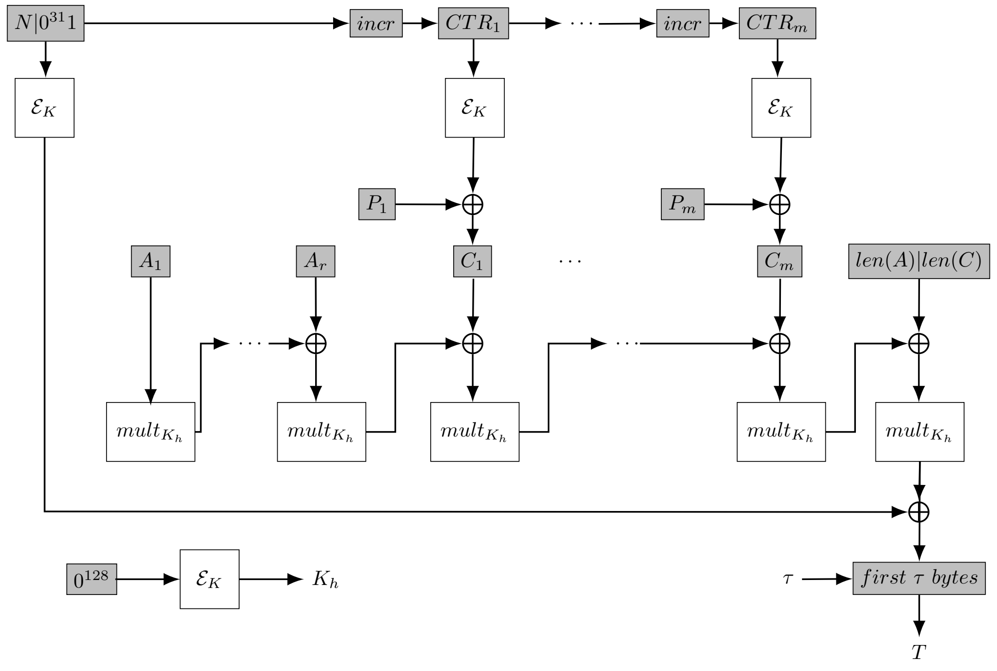


Figure 20: Galois/Counter Mode (GCM).

including TLS 1.2 [296], SSH [178] and IPSec [335]. AES-GCM is further included in the NSA Suite B Cryptography [263].
GCM uses a block cipher in CTR mode for encryption, and further utilises multiplications by elements of the finite field $\mathbb{F}_{2^n}$, specified by an irreducible polynomial. The counter of the block cipher is initialized by a nonce N and incremented for each message. The ciphertexts are then obtained by using the CTR mode. The tag for authentication is computed in a procedure called GHASH. Therefore, the tag is computed over the associated data and the ciphertexts, that are fed to a polynomial multiplication in the finite field over $\mathbb{F}_{2^n}$. The last block is then computed as $T = H \oplus \mathcal{E}_K(X \oplus (|A|||C|))$, where T is the final tag, H is a constant, X is the running polynomial, and $|A|, |C|$ are the length of the associated data and ciphertext. Figure 20 illustrates GCM.
GCM is ideal for data that is sent in packets, such as in SSH and TLS, because it has a very low latency and a minimum operation overhead. Moreover, AES-GCM, as it is used in many applications, achieves very high performance on high-end CPUs. This is mainly due to hardware acceleration on Intel CPUs, that were introduced as *AES New Instructions (AES-NI)* [165]. Due to its widespread use, GCM has also reached the focus of the cryptographic community and there are many cryptanalytic results on the security of GCM. Joux [192] showed that one can recover the hashing key H by using pairs of messages using the same nonce. Saarinen [294] showed weak keys in GHASH that lead to a hashing key of low degree. Procter and Cid [279] generalised the previous attacks based on weak keys according to algebraic properties of the polynomial-based hash function. Abdelraheem *et al.* [4]

showed the first universal forger attacks on GCM, that did not require the misuse of nonces.

Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR)

In 2013, the CAESAR competition [60] was announced to identify a portfolio of authenticated ciphers that offer advantages over AES-GCM and should be suitable for widespread adoption. In March 2014, 57 first round candidates were submitted to the CAESAR competition. Abed *et al.* [6] provided a classification and a general overview of the candidates. Table 4 gives an overview of all first-round candidates. In this thesis, we particularly focused on a few candidates based on the TWEAKEY framework. In the following, we give a more detailed overview for this candidates. A more interested reader should check the individual papers and design specifications for more details on other CAESAR candidates.

KIASU. KIASU is an authenticated encryption scheme designed by Jean *et al.* [187], that is based on ad-hoc tweakable AES integrated in two authenticated encryption modes, COPA [13] and ΘCB3 [211]. The round function is exactly the same as for AES-128, but with a fixed 64-bit tweak value T added to the first two rows of each subkey. KIASU provides 128-bit security (including related-key and related-tweak attacks). From a performance point of view, the speed overhead compared to standard AES-128 is minimal. The design benefits from all optimisations of AES and it is also backwards compatible to AES by setting the tweak to zero. From a security point of view, KIASU has not reached a lot of attention and there is just a little analysis published so far. KIASU was not selected to continue in the CAESAR competition after the first round. Dobraunig and List [141] provide impossible differential and boomerang attacks on eight rounds of KIASU. Moreover, Dobraunig *et al.* [139] published a 7-round square attack on KIASU.

JOLTIK. JOLTIK is a family of authenticated encryption schemes designed by Jean *et al.* [186], that is based on the STK construction of the TWEAKEY framework. The underlying block cipher of JOLTIK, is a lightweight 64-bit ad-hoc tweakable block cipher. The design is AES-like using the 4-bit S-box of PICCOLO. The linear layer consists of an involutory MDS matrix with a low decryption overhead. From a performance point of view, JOLTIK is very lightweight that can be implemented in hardware in ≈ 2100 GE. Moreover, JOLTIK behaves especially well for small messages. From a security point of view, JOLTIK has not reached a lot of attention and there is just a little analysis published so far. JOLTIK was not selected to continue in the CAESAR competition after the second round. The best published attack on JOLTIK is by Zong *et al.* [364] that is based on related-tweakey impossible differentials and reaches 10 rounds.

DEOXYS. DEOXYS is an authenticated encryption scheme designed by Jean *et al.* [189], that is based on the STK construction of the TWEAKEY framework. Similar as in KIASU, DEOXYS also uses exactly the AES round function as an underlying block cipher. However, in contrast to KIASU, DEOXYS employs a more complex, lin-

Table 4: Overview of the 1st round CAESAR candidates as classified by Abed *et al.* [6].

Construction	Name	Primitive	Parallel Enc. / Dec.	Online Enc. / Dec.	Inverse-free	Security proof	Nonce miss use-resistant
Block Cipher based	++AE [281]	AES	✓/✓	✓/✓	-	-	-
	AES-CMCC [330]	AES	-/✓	-/-	✓	✓	✓
	AES-COPA [14]	AES	✓/✓	✓/✓	✓	✓	✓
	AES-CPFB [253]	AES	✓/-	✓/✓	✓	✓	-
	AES-JAMBU [352]	AES	-/-	✓/✓	✓	-	-
	AES-OTR [249]	AES	✓/✓	✓/✓	✓	✓	-
	AEZ [173]	AES [4]	✓/✓	-/-	✓	✓	✓
	AVALANCHE [10]	AES	✓/✓	✓/✓	✓	✓	-
	CBA [175]	AES	✓/✓	✓/✓	✓	-	-
	CLOC [183]	AES, Twine	-/-	✓/✓	✓	✓	-
	Deoxys$^{\neq}$ [189]	Deoxys-BC	✓/✓	✓/✓	-	✓	-
	Deoxys$^{=}$ [189]	Deoxys-BC	✓/✓	✓/✓	-	✓	✓
	ELmD [124]	AES	✓/✓	✓/✓	-	✓	✓
	iFeed [360]	AES	✓/-	✓/✓	✓	✓	-
	iSCREAM [164]	iSCREAM, SPN	✓/✓	✓/✓	✓	-	-
	Joltik$^{\neq}$ [186]	Joltik-BC	✓/✓	✓/✓	-	✓	-
	Joltik$^{=}$ [186]	Joltik-BC	✓/✓	✓/✓	-	✓	✓
	Julius-CTR [36]	AES	✓/✓	-/-	✓	✓	-
	Julius-ECB [36]	AES	✓/✓	-/-	-	✓	✓
	KIASU$^{\neq}$ [187]	AES	✓/✓	✓/✓	-	✓	-
	KIASU$^{=}$ [187]	AES	✓/✓	✓/✓	-	✓	✓
	LAC [359]	LBlock	✓/✓	✓/✓	-	-	-
	OCB [212]	AES	✓/✓	✓/✓	-	✓	-
	POET [5]	AES	-/-	✓/✓	✓	✓	✓
	SCREAM [164]	SCREAM, SPN	✓/✓	✓/✓	✓	-	-
	SHELL [341]	AES	✓/✓	✓/✓	-	✓	✓
	SILC [184]	AES, Present, LED	-/✓	✓/✓	✓	✓	-
	Silver [276]	MAES	✓/✓	✓/✓	-	✓	-
	YAES [103]	AES	✓/✓	✓/✓	✓	-	-
Dedicated	AES-AEGIS [353]	AES	✓/-	✓/✓	✓	-	-
	MORUS [177]	MORUS	-/-	✓/✓	✓	-	-
	Tiaoxin [265]	AES [1]	✓/✓	✓/✓	✓	-	-

continued...

Construction	Name	Primitive	Parallel Enc. / Dec.	Online Enc. / Dec.	Inverse-free	Security proof	Nonce miss use-resistant
Stream Cipher based	ACORN [351]	ACORN	✓/✓	✓/✓	✓	-	-
	Enchilada [169]	ChaCha, Rijndael	✓/✓	✓/✓	✓	✓	-
	HS1-SIV [210]	ChaCha, Poly1305	-/-	-/-	✓	✓	✓
	Raviyoyla [337]	MAGv2	-/-	✓/✓	✓	-	-
	Sablier [357]	Sablier	✓/✓	✓/✓	✓	-	-
	TriviA-ck [112]	Trivium	✓/✓	-/-	✓	✓	-
	Wheesht [242]	Wheesht	-/-	✓/✓	✓	-	-
CF-based	OMD [115]	SHA-2	-/-	✓/✓	✓	✓	-
Permutation based	Minalpher [300]	Minalpher-P	✓/✓	✓/✓	-	✓	✓
	PAEQ [76]	AESQ	✓/✓	✓/✓	✓	✓	✓
	Prøst-COPA [195]	Prøst	✓/✓	✓/✓	✓	✓	✓
	Prøst-OTR [195]	Prøst	✓/✓	✓/✓	✓	✓	-
	Artemia [9]	JHAE	-/-	✓/✓	✓	✓	-
Sponge based	Ascon [140]	Ascon	-/-	✓/✓	✓	✓	✓
	ICEPOLE [254]	Keccack-like	✓/✓	✓/✓	✓	✓	-
	Ketje [62]	Keccack-f	-/-	✓/✓	✓	✓	-
	Keyak [63]	Keccack-f	✓/✓	✓/✓	✓	✓	-
	NORX [33]	NORX	✓/✓	✓/✓	✓	✓	-
	π-cipher [163]	π-cipher	✓/✓	✓/✓	✓	-	-
	PRIMATEs-GIBBON [12]	PRIMATE	-/-	✓/✓	✓	✓	-
	PRIMATEs-HANUMAN [12]	PRIMATE	-/-	✓/✓	✓	✓	-
	PRIMATEs-APE [12]	PRIMATE	-/-	✓/✓	-	✓	✓
	Prøst-APE [195]	Prøst	-/-	✓/✓	-	-	✓
	STRIBOB [274]	Streebog	-/-	✓/✓	✓	✓	-

ear tweakey schedule, instead of adding the same tweak each round. Deoxys also benefits from the performance of optimised implementations and hardware acceleration such as the AES-NI instructions. From a performance point of view, Deoxys achieves good software performance (less than a cycle per byte on recent processors). It also performs very good with small messages and can be implemented in hardware with around 4600 GE. From a security point of view, Deoxys has reached the most attention from all CAESAR candidates based on the Tweakey framework.

At the time of writing, Deoxys is still a candidate of the CAESAR competition, and it is considered for a high security use case. Cid *et al.* [113] published rectangle attacks, reaching 10 from 14 round for Deoxys-256 and 13 from 16 rounds for Deoxys-384. Sasaki improved the complexities of the attacks in [298].

2.4 CRYPTANALYSIS

Cryptanalysis (from the Greek *kryptós* meaning *hidden*, and *analýein* meaning *to loosen*) is the study of information systems and algorithms, in order to find hidden aspects of the system. In a general sense it is used to breach a cryptographic security system to gain access to encrypted data without the knowledge of the secret key. Essentially, an attacker tries to break one or more of the cryptographic goals that a secure cryptosystem shall achieve (*i.e.,* confidentiality, authenticity and integrity). We can distinguish between two forms of cryptanalysis. Firstly, we can consider *mathematical* or *conventional cryptanalysis* which studies the weaknesses of cryptographic algorithms. Secondly, we can consider *implementation attacks* or *side-channel attacks* that rather than searching for weaknesses in the cryptographic algorithms themself, an attacker tries to exploit weaknesses in the implementation of algorithms or exploits side-channel information such as power consumption, timing information, electromagnetic leaks or other side-channel information. In this thesis, we focus on *conventional cryptanalysis*.

2.4.1 Adversarial Models and Goals

When talking about attacks and adversaries we can classify the strengths and the goals of an attacker. This helps us to assess the severity of an successful attack. In particular, depending on available computational power any cryptosystem can be broken (*i.e.,* by using brute-force attacks). Therefore, we can also make a distinction between when an attack is considered a *theoretical* attack or when we talk about *practical* attacks.

Adversarial Models

In the following, we consider several different adversarial models and list them in order of the strength of their assumption (from weaker to stronger).

- *Ciphertext-only attacks:* In this scenario, the attacker can obtain only the ciphertexts C produced by the encryption algorithm $\mathcal{E}_K$ of a cipher. An attacker can observe statistical abnormalities in the ciphertexts to attack the cipher. This scenario can be considered when an attacker is for example able to intercept an encrypted communication channel and observes some of the ciphertexts.
- *Known-plaintext attacks:* In this scenario, the attacker is capable of obtaining a set of plaintext/ciphertext pairs (P_i, C_i), such that $C_i = \mathcal{E}_K(P_i)$. This scenario can be considered, when an attacker is for example intercepts ciphertexts but also gets access to the corresponding plaintexts. An important distinction to

the following models, in the known-plaintext model, the attacker until now just has read access.

- *Chosen-ciphertext attacks:* In this scenario, the attacker is able to obtain a set of plaintexts P_i such that $P_i = \mathcal{D}_K(C_i)$ is the decryption of a set of chosen ciphertexts C_i by the choice of the attacker. This scenario can be considered, when the attacker has physical access to an encryption device and can query it with chosen-ciphertext queries.

- *Chosen-plaintext attacks:* This scenario is the dual to the *chosen-ciphertext attack*, such that an attacker can obtain a set of ciphertexts C_i that are the encryption of a set of chosen plaintexts $C_i = \mathcal{E}_K(P_i)$. Compared to *known-plaintext attacks* the plaintexts are now a choice of the attacker.

- *Adaptive Chosen-ciphertext attacks:* This scenario is similar to normal *chosen-ciphertext attacks*, however, the attacker can now adaptive choose the ciphertexts C_j depending on the obtained plaintext/ciphertext pairs (P_i, C_i) for $1 \leqslant i \leqslant j \leqslant n$, in a set of n plaintext/ciphertext pairs (P_i, C_i).

- *Adaptive Chosen-plaintext attacks:* This scenario is the dual to the *adaptive chosen-ciphertext attacks*, however, the attacker can now adaptive choose the plaintexts P_j depending on the obtained plaintext/ciphertext pairs (P_i, C_i) for $1 \leqslant i \leqslant j \leqslant n$, in a set of n plaintext/ciphertext pairs (P_i, C_i).

Additionally, to the adversarial model of an attack we can also consider different models about the secret-key. Again, we list them in order of the strength of their assumption (from weaker to stronger).

- *Secret-key model:* This is the most common model and also complies with Kerckhoffs' principle as outlined in Section 2.2.3 that states that the security of a cryptographic system should rely on the secrecy of the key-material. In this model, it is assumed that the attack has no knowledge of the secret-key.

- *Related-key model:* In this model, an attacker is allowed to analyse the cryptographic primitive not just with one single secret-key K, but rather with a set of related-keys K_i, where additionally the attacker can choose the relation herself. In Chapter 6 we attack the block cipher SKINNY with a related-tweakey impossible differential attack.

- *Known-key model:* In this model, the attacker is actually allowed to know the secret-key. The model was initially proposed by Knudsen and Rijmen [203] to distinguish ciphertext from random with the knowledge of the key. The aim of this model is to find structural weaknesses in a cipher. While the keys are known to the attacker, this model does not directly compromise the confidentiality of a cipher.

- *Chosen-key model:* This model is similar to the *known-key model*, however, the attacker can also choose the key herself. This is an even stronger assumption in the favour of an attacker.

Adversarial Goals

When considering cryptographic attacks, we can rank the attacks by severity of the impact to a cipher. In the following we list the attacks in order of the severity of the attacks (from weaker to stronger).

- *Distinguisher:* In a distinguishing attack, an attacker tries to efficiently distinguish between a cryptographic primitive and an ideal version of a primitive. In case of a block cipher, the ideal version of a primitive would be a random permutation.
- *Local deduction:* In a local deduction, an attacker is able to determine a single plaintext/ciphertext pair (P, C) such that $C = \mathcal{E}_K(P)$, where neither the plaintext P or the ciphertext C have been previously been observed by the attacker.
- *Global deduction:* In a global deduction, an attacker is able to find a function F that is functionally equivalent to the encryption algorithm $\mathcal{E}_K$, without known the secret key K.
- *Key-recovery:* In a key-recovery attack the attacker is able to recover parts, or the whole secret-key K. If an attacker is able to recover the whole secret-key we can also talk about a *total break*.

Complexity Metrics

Evaluating the complexity of an attack is an important step in an cryptographic attack against a cipher. In general, we want to check if the attack improves upon simple brute-force attacks. Moreover, many designers claim security of their primitives based on some complexity bounds. To consider an attack valid, the attack has to improve against brute-force.

- *Time:* As time complexity we understand the time that is required to perform the attack. The unit for time in that case is sometimes, however, different from the normal units such as seconds, minutes, hours and days. As theoretical attacks often require a complexity of above 2^{64}, the unit is often given in CPU instructions, or evaluations of the encryption function. Furthermore, often we distinguish between an *offline phase*, also called *pre-computation* phase, that is the time when plaintext/ciphertext pairs are observed, and secondly an *online phase*, which is the time were the attack is performed on the obtained data.
- *Memory:* The memory complexity gives the number of read/write accesses to a data structure that is used to store data used in the attack. Memory complexity is normally given in bytes. On a high-end computer a complexity of 2^{40} bytes (*i.e.,* 1TB) for read/write accesses to a data structure is an upper limit before the arithmetic calculations could become a bottleneck.
- *Data:* The data complexity refers to the number of plaintexts, ciphertext or a combination of both that are used as input data to attack the cryptographic primitive. For a valid attack, the data complexity is limited to the block size of the cryptographic primitive. Otherwise, an attacker could just build a so called *full code book* containing all plaintext/ciphertext combinations, under

one key. Therefore, we require that the data complexity is always $< 2^n$ for an n-bit block cipher.

Moreover, we can distinguish between *theoretical attacks* and *practical attacks*. For an attack to be considered practical we can consider several metrics. By considering time/memory/data complexity we can argue that a time complexity of around 2^{50} computations is acceptable on a modern desktop computer. Moreover, a memory complexity of 2^{40} which would be equal to approximately 1 Terabyte read/write accesses to a data structure, and further a data complexity of around 2^{50} which is equal to 1 Petabyte on input data would be acceptable. If we consider costs as a measurement metric, we can argue that around 10000 US\$[6] would be an appropriate limit for an practical attack. Nevertheless, we can also consider more powerful attackers such as nation-state adversaries or organisations with a much higher budget and access to more powerful computer-clusters. An example for such an attack could be the first SHA-1 collision [314] that took about 1 year on 110 high-end GPUs which can be compared to a time complexity of around $2^{63.1}$ computations.

2.4.2 Differential Cryptanalysis

Differential cryptanalysis is one of the most powerful attack vectors in the analysis of symmetric-key primitives. It has been firstly published by Biham and Shamir [73] to analyse DES, and has become one of the prime attack vectors which any modern symmetric-key primitive has to be resistant against. Most differential attacks are applied to block ciphers, but they are also applicable to stream ciphers, hash functions and other symmetric-key primitives. In a nutshell, the idea behind differential cryptanalysis is to find pairs of plaintexts and ciphertexts, where a certain difference between those texts occurs with high probability. The challenge for a cryptanalyst consists of finding differences with a high probability or to show that no such difference exists. In a random permutation such difference occurs with a probability of 2^{-n}, where n is the length of the permutation. In a block cipher we try to find a difference that occurs with a probability $> 2^{-n}$ to get an efficient distinguisher. Later, we can use this distinguisher in a key-recovery attack as we will discuss later in this Section.

Definition 7. Let $P, P' \in \mathbb{F}_2^n$ be two plaintexts of length n. The *XOR-difference* between P and P' can be defined as

$$\alpha = P \oplus P'.$$

In general, in the analysis of symmetric-key primitives we use XOR-differences. However, that is not necessarily a strict requirement, and any other difference can be used (*e.g.*, one might use the difference of modular addition $P \boxplus P'$). The choice of the difference mainly depends on with what operation the secret-key is added to the internal state. More details on the choice of differences can be found in [130, 223].

[6] This would allow one to rent [11] approximately 6400 Intel Xeon CPUs with 2.3GHz, 256TB RAM and 3600TB SSD storage for 1 day. Moreover, this would correspond to a time complexity of 2^{52}, a memory complexity of 2^{48} and a data complexity of 2^{51}.

The idea of differential cryptanalysis is to look at pairs of plaintexts (P, P') and the corresponding ciphertexts (C, C') and try to find some texts with differences α and β, where $\alpha = P \oplus P'$ and $\beta = C \oplus C'$, occurs with a high probability.

Definition 8. Let F be a vectorial Boolean function of the form $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$. A *differential* is a pair of differences $(\alpha, \beta) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$, where we call α the input difference, and β the output difference of the function F.

We further denote a differential (α, β) over F as $\alpha \xrightarrow{F} \beta$. Moreover, in an attack we often encrypt many plaintext pairs (P, P') and want them to achieve the desired input difference α. Therefore, we can denote a pair of plaintexts (P, P') as a *right pair*, if it fulfils the desired difference, or as a *wrong pair* if it does not.

Definition 9. The *differential probability* of a differential (α, β) over a vectorial Boolean function F is given by

$$\mathrm{DP}(\alpha \xrightarrow{F} \beta) = \Pr_X(F(X) \oplus F(X \oplus \alpha) = \beta). \tag{38}$$

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

For ease of notation we define the *weight* of a differential as $-\log_2(\mathrm{DP}(\cdot))$. Any non-zero differential for a random permutation $F_\$: \mathbb{F}_2^n \to \mathbb{F}_2^n$ will have a differential probability close to 2^{-n}. Therefore one is interested in finding any differential with $\mathrm{DP}(\alpha \xrightarrow{F} \beta) \gg 2^{-n}$. In general, it is computationally infeasible to compute the exact value of the DP as this would require to exhaustively search through the whole space of all possible plaintexts. One can use the structure of a block cipher, to obtain a good approximation of the actual DP with less computational effort by tracking the differences through the round functions. Virtually all currently used block ciphers are *iterative* block ciphers, *i.e.,* they are composed of applying a simple round function r times

$$E_K(\cdot) = f_r(\cdot) \circ \ldots \circ f_1(\cdot). \tag{39}$$

Definition 10. A *differential trail* Q is a sequence of differences $\alpha \xrightarrow{F} \beta$ over several iterations of a function F

$$Q = (\alpha_1 \xrightarrow{F_1} \alpha_2 \xrightarrow{F_2} \ldots \xrightarrow{F_{r-1}} \alpha_r). \tag{40}$$

Yet, it is still computationally infeasible to compute the exact value of $\mathrm{DP}(Q)$ and the general approach is to assume independence of the rounds. For most designs it is feasible to compute the exact probability of a differential for a single round.

Definition 11. Let Q be a differential trail over a several iterations of a vectorial Boolean function F, then the *differential trail probability* can be defined as

$$\mathrm{DTP}(Q) = \prod_{i=1}^{r-1} \Pr_X(\alpha_i \xrightarrow[X]{f_i} \alpha_{i+1}). \tag{41}$$

While this assumption of independent rounds is not true in general, it has been shown to serve as a good approximation in practice [116, 172]. However, if an adversary wants to construct a distinguisher, she actually does not care about any intermediate differences and is only interested in the probability of the differential. The adversary can therefore collect all differential trails sharing the same input and output difference to get a better estimate

$$\Pr(\alpha_1 \xrightarrow{E} \alpha_r) = \sum_{\alpha_2,\dots,\alpha_{r-1}} \Pr_X(\alpha_1 \xrightarrow[X]{f_1} \alpha_2 \xrightarrow[f_1(X)]{f_2} \cdots \alpha_{r-1} \xrightarrow[f_{r-1}\circ\dots\circ f_1(X)]{f_{r-1}} \alpha_r). \quad (42)$$

It is often assumed that the probability of the differential is close to the probability of the best single trail. While this might hold for some ciphers this assumption has been shown to be inaccurate in several cases and does not hold for many modern block ciphers [75, 137]. We will show later in Chapter 3 that this assumption fails particularly often for some recently designed lightweight block ciphers.
We consider two different criteria for a design: *differential trail resistant* (DTR), which means that no single differential trail exists with a probability larger than 2^{-n} and *differential resistant* (DR) which means that it should be difficult to find a differential with a probability larger than 2^{-n}. Note that we typically cannot avoid that there are differentials with DP $> 2^{-n}$, as if we fix the input difference to α_1 then $\sum_{\alpha_r \neq 0} \Pr(\alpha_1 \xrightarrow{E} \alpha_r) = 1$. This implies that there exists at least one differential with a probability DP $> 2^{-n}$. In the *Wide-Trail Strategy* which was used to design the AES and subsequently many other ciphers, Daemen and Rijmen suggest that it is a sound design strategy to restrict the probability of difference propagation [120]. Nevertheless, this does not result in a proof for security.
Note that in the definitions so far the influence of the keys was ignored. However, the DP for a specific differential strongly depends on the choice of the secret key and it is therefore of interest how this distribution looks like. To solve this problem we could compute the probabilities of a differential over the whole key space, however this is again practically infeasible which leads one to use the *expected differential probability*.

Definition 12. The *expected differential probability* of a block cipher E_k of an r-round differential (α, β), with a key-size of κ-bits is defined as

$$\mathrm{EDP}(\alpha \xrightarrow{E} \beta) = 2^{\kappa} \sum_{k \in \mathbb{F}_2^{\kappa}} \Pr_X(\alpha \xrightarrow[X]{E_k} \beta). \quad (43)$$

Lai *et al.* [215] showed that for an iterated cipher, if the round function satisfies that the differential probability is independent of the choice of one of the component plaintexts under a difference, it can be considered a Markov cipher.

Definition 13 ([215], Def. page 22.)**.** Let $E_K(\cdot)$ be an iterated cipher of r rounds. A *Markov cipher* can then be defined if it there is a group operation $\otimes$ for defining differences such that, for all choices of α and β,

$$\Pr(\Delta Y = \beta | \Delta X = \alpha, X = \gamma)$$

is independent of γ, when the key K is drawn uniform at random, *i.e.*, $K \xleftarrow{\$} \mathbb{F}_2^{\kappa}$, or, equivalently, if

$$\Pr(\Delta Y = \beta | \Delta X = \alpha, X = \gamma) = \Pr(\Delta Y(1) = \beta_1 | \Delta X = \alpha)$$

for all choices of γ when the key K is drawn uniform at random.

Moreover, for a Markov cipher that has independent round-keys, the sequence of round differences forms a Markov chain.

Definition 14 ([215], Theorem 2, page 22.)**.** If an r-round iterated cipher $E_K(\cdot)$ is a *Markov cipher* and the r round keys are independent and uniformly random, then the differential trail $Q(\alpha_1 \xrightarrow{E} \alpha_r) = \alpha_1 \xrightarrow[X]{f_1} \alpha_2 \xrightarrow[f_1(X)]{f_2} \cdots \alpha_{r-1} \xrightarrow[f_{r-1}\circ\ldots\circ f_1(X)]{f_{r-1}} \alpha_r$ is a homogeneous *Markov chain*.

In order to derive a security proof against differential cryptanalysis often the *Hypothesis of Stochastic Equivalence* [215] is used which states that for all differentials Q it holds that for most keys K the differential probability of a trail is close to the expected differential probability, $DP_K(Q) \approx EDP(Q)$.

Definition 15. Let $E_K(\cdot)$ be an iterated cipher of r rounds. The *Hypothesis of Stochastic Equivalence* states that for all r-round differentials (α, β) it holds that

$$\Pr(\alpha \xrightarrow{E_K} \beta) \approx \Pr(\alpha \xrightarrow{E_{K'}} \beta). \tag{44}$$

Moreover, this holds for almost all subkey values $K \in \mathbb{F}_2^{\kappa}$.

In practice this hypothesis does not always hold [107], which we will also see later in Chapter 3.

Distinguisher

In a cryptographic attack an attacker first tries to find a distinguisher for the cryptographic primitive and then to mount a key-recovery based on the distinguisher. Let's focus now on a block cipher E, and let (α, β) be a differential of E with a probability $DP_E(\alpha, \beta)$. For a valid attack, we require that the $DP_E(\alpha, \beta)$ of the distinguisher is $\gg 2^{-n}$, as the DP of a random permutation is 2^n.
We can then consider an oracle $\mathcal{O}$, that the attacker can interact with. The attacker can query the oracle with either plaintexts or ciphertexts, depending on the attack model as defined in Section 2.4.1, and gets the corresponding outputs. The oracle then either returns the output of the block cipher E or the output of a random permutation π. The goal of the attacker then is to distinguish if the output originated from the block cipher or the random permutation. In the case of differential cryptanalysis, the attacker can therefore query the oracle $\mathcal{O}$ with a pair of inputs $P, P \oplus \alpha$, where α is the input difference. If the oracle then returns two ciphertexts (C, C') where $C' = C \oplus \beta$, and β is the output difference, then the attacker knows with a high probability that $(C, C') = \mathcal{O}(P), \mathcal{O}(P \oplus \alpha)$ and the oracle returned the output of the block cipher E. If any other output difference is returned, then with a high probability the oracle $\mathcal{O}$ returned an output of the random permutation π.
Finding differential distinguishers efficiently is a hard problem. However, there are many automated tools that help with the search for various cryptographic primitives. Mouha *et al.* [257] used Mixed Integer Linear Programming (MILP) to count active S-boxes and to compute provable bounds. Furthermore, there have been a few approaches of using automated tools to find optimal trails, and to collect many

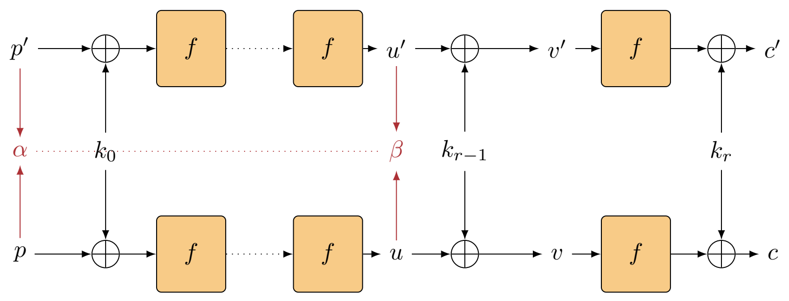


Figure 21: Example of an r-round key-recovery attack based on an $(r-1)$-round differential distinguisher.

trails with the same input/output differences. This idea was first introduced by Sun *et al.* [317] who used MILP. Likewise, tools using SAT/SMT solvers are used where the results were applied to SALSA-20 [256], NORX [32], and SIMON [206]. Chapter 3 gives more details about the search for optimal differential trails based on a SAT/SMT-based tool.

Key-Recovery Attacks

Let $\mathcal{E}_K[r]$ be an r-round key-alternating block cipher, with a secret key $K \in \mathbb{F}_2^\kappa$ consisting of an invertible round function f, so that the ciphertext C_i can be expressed as $C_i = f(x_i) \oplus k_r^*$, where x_i denotes the state at $(r-1)$ rounds and k_r^* is the actual final r^{th} round subkey. In a key-recovery attack, we use an $r-1$ round differential (α, β) with a differential probability $DP_E(\alpha, \beta) \gg 2^{-n}$, where n is the block size of $\mathcal{E}_K[r]$. In the following, we describe an r-round attack on the block cipher $\mathcal{E}_K[r]$.
Let's assume we found an $(r-1)$-round differential (α, β) for the block cipher $\mathcal{E}_K[r]$, with high differential probability, as outlined in Figure 21. We can turn the $(r-1)$-round differential into a key-recovery attack on r-rounds.

ATTACK STEPS. We assume a *chosen-plaintext attack (CPA)* scenario, where the attacker is able to obtain N ciphertext pairs (C_i, C_i'), that correspond to the encryption of N chosen plaintext pairs (P_i, P_i') under a key $K \in \mathbb{F}_2^\kappa$ and $1 \leqslant i \leqslant N$.
The attack consists of the following steps:

1. Initialise a list $\mathcal{K}$ of 2^κ key counters and set them all to zero, for each possible guess of key k_r.
2. For each of the 2^κ guesses of k_r:
   - 2.1 Decrypt both ciphertexts (C_i, C_i') for one round, *i.e.*, for $i = 1, \dots, N$ compute $v = f^{-1}(C_i \oplus k_r)$ and $v' = f^{-1}(C_i' \oplus k_r)$.
   - 2.2 Increase the key counter in $\mathcal{K}$, at the position of k_r for each pair that satisfies $(v \oplus v') = \beta$.
3. Output the key candidate in $\mathcal{K}$ in descending order of their corresponding counters.

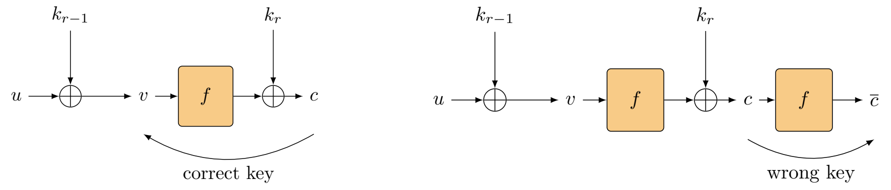


Figure 22: **(Left:)** A *correct* key-guess decrypts one round to the end of the differential distinguisher, **(Right:)** A *wrong* key-guess randomizes the state by encrypting one more round with a random-key (*i.e.,* the wrong key) value.

A differential (α, β) is assumed to have a differential probability p, for a correct key. Therefore, the expected key counter for the key candidate is Np, where N denotes the number of chosen-plaintexts. Statistically, we will also get false-positive candidates, and let's denote by p' the probability of a right pair, with a wrong key $k'_r \neq k_r$, for which the key counter is also Np'. Yet, assuming that the attacker found a differential with high probability, we expect that $p \gg p'$ and the counter for a correct key guess is significantly higher then the one for wrong key guesses. This assumption is based on the fact that for wrong pairs, we assume that the counters of the key candidates are distributed uniform at random. This is also characterised in the *hypothesis of wrong-key randomisation*. Basically, we can further say that a correct key guess decrypts one round to the state of round $r - 1$ as illustrated in Figure 22. However, a wrong key guess would randomise the state by encrypting for another round with the wrongly guessed key candidate, as depicted in Figure 22.

COMPLEXITY ESTIMATION AND CORRECTNESS. The success probability of a differential attack can be quantified with the *signal-to-noise ratio*. We can call the event that a *right pair* suggests the correct key value a *signal*, while we call a *false-positive pair* that suggests a wrong key value as *noise*. For a successful attack the *signal-to-noise ratio* has to diverge from 1. To achieve a good ratio, an attacker therefore has to increase the number of chosen-text pairs, which we can quantify by $c/\operatorname{DP}_E(\alpha, \beta)$ pairs (P_i, P'_i) that follow the differential (α, β), and c denotes a small constant.
We can then estimate the data complexity of the key-recovery attack by $c/\operatorname{DP}_E(\alpha, \beta)$ chosen-text pairs. The time complexity is then dependent on this number of chosen-texts and any additional computational overhead. The memory complexity stems from the number of potential key-counters one has to store. To improve either time or memory complexity, time-memory trade-off techniques [171] can be used.

2.4.3 Variants of Differential Cryptanalysis

Since differential cryptanalysis has firstly been published, many different variants have been developed. These variants simplify parts of the analysis and therefore often lower the attack complexities or allow to find distinguishers for a larger number of rounds. In the following, we are going to describe some of the more popular variants of differential cryptanalysis, including: *higher-order differentials*, *truncated differentials*, *impossible differentials* and *boomerang/rectangle* attacks.

Higher-order Differential Cryptanalysis

Higher-order differential cryptanalysis is a generalisation of differential cryptanalysis. While standard differential cryptanalysis uses differences between two messages, either plaintexts or ciphertexts, in higher-order differential cryptanalysis the propagation of a set of differences in a larger set of messages is used. Lai [213] firstly showed that differentials can be seen as a special case of higher order derivates. Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be a block cipher that maps n-bit strings to n-bit strings, for a fixed key K. Then in standard differential cryptanalysis we are interested in finding a pair of messages with an input difference α that results in an output difference β such that

$$F(x \oplus \alpha) \oplus F(x) = \beta$$

holds for many messages $x \in \mathbb{F}_2^n$. However, we can also describe this differential equation as

$$\Delta_\alpha F(x) = F(x \oplus \alpha) \oplus F(x),$$

where $\Delta_\alpha F(\cdot)$ denotes the first-order derivation of F at a point α. Moreover, we can then define the d^{th}-order derivative recursively as

$$\Delta^{(d)}_{\alpha_1,\dots,\alpha_d} F(x) = \Delta_{\alpha_d}(\Delta^{(d-1)}_{\alpha_1,\dots,\alpha_{d-1}} F(x)).$$

Based on this observation, Knudsen [201] was able to show that the concept of higher order derivatives can be used in attacks against block ciphers. In general, this works by considering a block cipher as a Boolean function and the fact that taking the derivative of a function decreases the algebraic degree of the function. Therefore, if we consider a block cipher $\mathcal{E}_K$ that has an algebraic degree of d, then the $(d+1)^{th}$ order derivative of $\mathcal{E}_K$ is zero, for any given input $x \in \mathbb{F}_2^n$. We will further describe this *zero-sum* property in Section 2.4.6, when we describe *integral attacks*.
While many modern block ciphers ensure that the algebraic degree is growing fast enough, so the set of messages required, exceeds the data complexity there is one published attack by Jakobsen and Knudsen [185] breaking the KN-CIPHER [273], which was designed to be resistant against standard differential cryptanalysis, but can be broken using higher order differential attacks.

Truncated Differential Cryptanalysis

Truncated Differentials were firstly introduced by Knudsen [201] in 1994. It is a generalisation of standard differential cryptanalysis. In comparison to standard cryptanalysis, that analyses *full* differences between two messages over a word or the whole blocksize, in truncated differential cryptanalysis differences are just partially determined, while leaving parts of the difference unspecified. We can formally define a *truncated differential* as:

Definition 16. Let α be an t-bit *truncated difference*, where $\alpha \in \{0, 1, *\}^t$. In this bitwise notation 1 denotes that there is a difference, 0 denotes there is no difference, and $*$ is a wildcard that indicates either a 0 or a 1.

For example, given a truncated difference $\alpha =$ `0*1`, with 3-bit, then this would allow differences $\alpha =$ `001,011`.

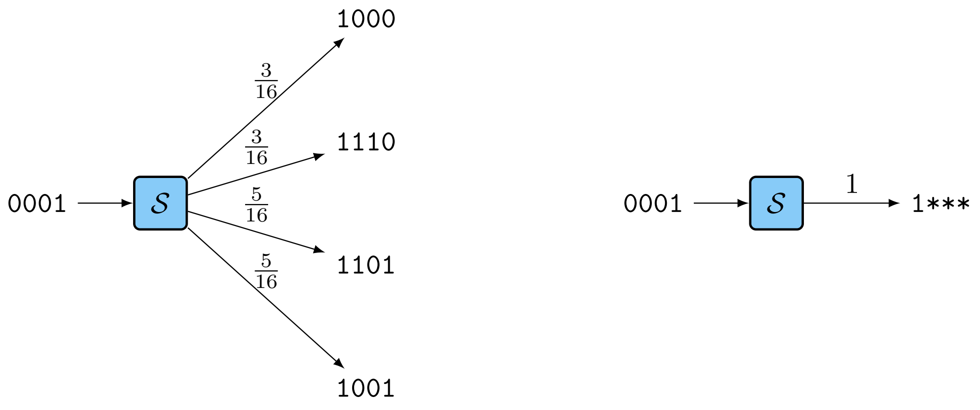


Figure 23: **(Left):** Differential probabilities using standard differential cryptanalysis, **(Right):** Differential probability using truncated differentials.

The advantage of using *truncated differentials* becomes distinct when looking at the differential probability of a truncated differential. In general, truncated differentials are a more flexible approach to express expected differences between two texts. In the following, lets consider the differential transition of a difference through a S-box S. Then for example an input difference $\alpha = \texttt{0001}$ results in output differences $\beta_1 = \texttt{1000}$ with probability $3/16$, $\beta_2 = \texttt{1110}$ with probability $3/16$, $\beta_3 = \texttt{1101}$ with probability $5/16$, $\beta_3 = \texttt{1001}$ with probability $5/16$ and to all other possible output differences with probability 0. Then instead of considering just a fixed differential, *i.e.*, $\alpha \xrightarrow{S} \beta_2$ with probability $5/16$, we can also consider the truncated difference $\gamma = \texttt{1}{*}{*}{*}$ where the differential $\alpha \xrightarrow{S} \gamma$ results in a probability 1. Figure 23 further illustrates the improved differential probability if we consider a truncated differential $\gamma = \texttt{1}{*}{*}{*}$ instead of standard differential cryptanalysis.
Truncated differentials have been used to improve many attacks [194, 202, 204]. Moreover, in Chapter 7 we show a truncated differential attack on a round-reduced version of the block cipher Sparx.

Impossible Differentials

Impossible differential attacks have been introduced independently by Biham *et al.* [69, 70] and Knudsen [200], and they are widely used as an important cryptanalytic technique. It is a special form of standard differential cryptanalysis. While in standard differential cryptanalysis an attacker is interested in finding a differential with a probability higher than the expected probability of a random permutation, in impossible differential cryptanalysis the attacker tries to find an impossible differential, meaning the probability should be exactly zero. We can formally define an *impossible differential* by

Definition 17. Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be a Boolean function, and let $(\alpha, \beta) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$ be a differential. The differential (α, β) is called an *impossible differential* if the differential probability $\mathrm{DP}(\alpha, \beta) = 0$.

We can find impossible differentials by considering a differential (α, β) that covers a few rounds in encryption direction, and a second differential (γ, δ) that covers a few rounds in decryption direction. We require that both differentials have a probability $\mathrm{DP}(\alpha, \beta) = \mathrm{DP}(\gamma, \delta) = 1$ and we choose the differences in such way that $\beta \neq \gamma$ contradicts at at least one bit. Then, when considering the differential (α, δ) the probability is exactly zero which would then be an impossible differential.
For a key-recovery attack we can add rounds before and/or after the impossible differential. An attacker can then collect pairs with certain plaintext and ciphertext differences. If there exists a pair that meets the input and output values of the impossible differential under some subkey, these subkeys must be wrong, as the differential has probability zero. In this way, we filter as many wrong keys as possible and exhaustively search through the rest of the key-candidates.
In Chapter 6 we show an related-key impossible differential attack on a round-reduced version of the block cipher Skinny.

Boomerang Attacks

Boomerang attacks, as proposed by Wagner [338], allow an attacker to concatenate two short differentials with high probability when long differentials with sufficient probability are absent or hard to find. In the basic setting, an adversary decomposes an encryption function $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ into two sub-ciphers $E = E_2 \circ E_1$, such that $E(P) \overset{\text{def}}{=} E_2(E_1(P))$. Then, it considers a first differential $\alpha \to \beta$ with probability p over E_1 and a second differential $\gamma \to \delta$ with probability q over E_2. These are often called upper and lower differentials or trails, respectively. They can then be combined in a chosen-plaintext and adaptive chosen-ciphertext attack to construct a boomerang distinguisher that consists of the following steps:

1. Choose a plaintext pair (P, P') with difference $\alpha = P \oplus P'$ and encrypt it through E to obtain its ciphertext pair (C, C') with difference β.
2. Derive $D = C \oplus \delta$ and $D' = C' \oplus \delta$ (the δ-shift) and decrypt D and D' through E^{-1} to obtain the corresponding plaintext pair (Q, Q').
3. If the plaintext pair (Q, Q') has difference $\alpha = Q \oplus Q'$, then (P, P', Q, Q') form a correct quartet.

Proposition 1. For a quartet (P, P', Q, Q'), there exists a differential with an input difference α for $P' = P \oplus \alpha$, $Q' = Q \oplus \alpha$, and a corresponding output difference β for $U' = U \oplus \beta$, $V' = V \oplus \beta$ with probability p. If we consider a differential $\delta \to \gamma$ with input difference $D = C \oplus \delta$, $D' = C' \oplus \delta$ and a corresponding output difference γ for $V = U \oplus \gamma$, it holds with probability q that $V' = U' \oplus \gamma$. Then, we can connect both differentials if we consider $V = U \oplus \gamma$, it follows that $V' = V \oplus \beta = (U \oplus \gamma) \oplus \beta = (U \oplus \beta) \oplus \gamma = U' \oplus \gamma$.

A boomerang distinguisher is outlined in Figure 24. Calculating the probabilities for a correct quartet requires to consider both plaintext pairs (P, P') and (Q, Q') and results in a probability of $(pq)^2$. In the attack published by Wagner [338], the probability of the boomerang distinguisher is asserted as

$$\Pr[E^{-1}(E(x \oplus \alpha) \oplus \delta) \oplus E^{-1}(E(x) \oplus \delta) = \alpha] = (pq)^2.$$

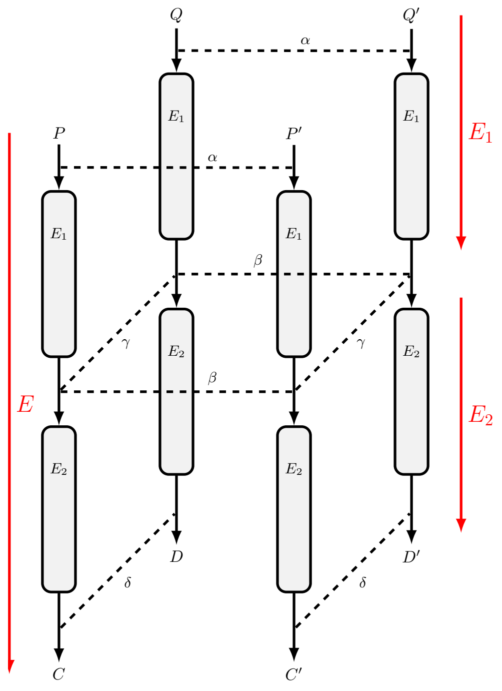


Figure 24: Boomerang distinguisher.

The initial assumption for boomerang attacks was that the two differential trails for E_1 and E_2 can be chosen independently. However, Murphy [258] showed that for SPN ciphers, independently chosen differential trails can be incompatible, meaning that the probability of a right quartet can be zero. Therefore, for a successful boomerang attack, the rounds in the middle construction have to be verified for correctness. This further lead to the development of the *sandwich attack* [144].
Moreover, the probability of a correct quartet can be improved if one fixes input differences α and δ but allows all possible differences for β and γ, with the only requirement that $\beta \neq \gamma$. A boomerang distinguisher would then consider all trails of the form $\alpha \to \beta'$ for the upper trail and $\delta \to \gamma'$ for the lower trail. This increases the probability to $(\widehat{p}\widehat{q})^2$ where $\widehat{p} = \sqrt{\sum_{\beta'} \Pr^2[\alpha \to \beta']}$ and $\widehat{q} = \sqrt{\sum_{\gamma'} \Pr^2[\delta \to \gamma']}$ where $\widehat{p}$ is evaluated over E_1 and $\widehat{q}$ over E_2^{-1}, respectively.

CONNECTING BOOMERANG TRAILS. There exist a few approaches to increase the transitional probability of boomerang trails in the middle, *i.e.,* in the switching phase between top and bottom trail. Three well-established approaches are the *Feistel switch*, *ladder switch* and *S-box switch* [77]. Recently, Cid *et al.*[114] proposed a tool called *Boomerang Connectivity Table* that calculates the probabilities for the connecting round in the middle.

Rectangle Attacks

In boomerang attacks, the adversary needs to query its oracles with chosen plaintexts and adaptively chosen ciphertexts. One can turn a boomerang attack into a entirely chosen-plaintext attack by considering *Rectangle* attacks. *Rectangle* attacks [71] have been derived from the *amplified boomerang* [197], both of which transform the boomerang into a purely chosen-plaintext attack (or chosen-ciphertext, if the adversary starts from the opposite direction). The core idea is to encrypt many pairs (P, P') with difference $P' \oplus P = \alpha$ in the hope that some of those will form a quartet with the desired differences in the middle with probability 2^{-n}. Given N plaintext pairs, the number of correct quartets is reduced to $N^2 \cdot 2^{-n} \cdot (\widehat{p}\widehat{q})^2$. Note that two pairs (U, U') and (V, V') can be combined in two distinct ways to a quartet in the middle: $U \oplus V = U' \oplus V' = \beta$ or $U \oplus U' = V \oplus V' = \beta$. [72] presented further improvements to the technique. The disadvantages of rectangle compared to boomerang attacks are the increased data complexity and the large number of potential quartets that have to be filtered to find correct quartets.

2.4.4 Linear Cryptanalysis

Linear cryptanalysis is based on finding linear approximations to the partly nonlinear behaviour of the components of a cipher. It is together with differential cryptanalysis one of the most widely used tools in the analysis of block ciphers. Linear cryptanalysis was initially discovered by Matsui and Yamagishi [240], who applied the attack against the FEAL cipher in 1992. Subsequently, Matsui [238] applied the attack on the DES block cipher. While differential cryptanalysis works as a chosen-plaintext attack, linear cryptanalysis is considered a known-plaintext attack. Moreover, differential cryptanalysis considers pairs of plaintext and ciphertexts, while in linear cryptanalysis we just consider plaintexts and ciphertexts but not in pairs. In a nutshell, in linear cryptanalysis we try to approximate a linear Boolean function of the output of the block cipher $\mathcal{E}_K(x)$ by a linear function of the input x. Therefore, an attacker searches for a pair of input and output masks α, γ such that the bias of the linear approximation

$$\langle \gamma, \mathcal{E}_K(x) \rangle \approx \langle \alpha, x \rangle$$

becomes large. We can define the bias $\epsilon_{\mathcal{E}_K}(\alpha, \gamma)$ by

$$\Pr_X[\langle \gamma, \mathcal{E}_K(x) \rangle = \langle \alpha, x \rangle] = \frac{1}{2} + \epsilon_{\mathcal{E}_K}(\alpha, \gamma).$$

Definition 18. Let a *linear mask* be a value $\alpha \in \mathbb{F}_2^n$. We can define a bit that is 1 in a mask as an *active* bit. Moreover, we use a mask to select/partition bits from an n-bit value $X \in \mathbb{F}_2^n$. We can then use the *linear mask* $\alpha = \alpha_{n-1} \| \ldots \| \alpha_0$ to select certain bits at the positions with *active* bits from the value $X = X_{n-1} \| \ldots \| X_0$, where the parity of the *masked bits* is computed by

$$\langle \alpha, X \rangle = \bigoplus_{i=0}^{n-1} \alpha_i X_i.$$

In comparison to differential cryptanalysis, a *linear mask* can be compared to a *difference*.

Definition 19. Let $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^m$ be a vectorial Boolean function. If we evaluate this function at positions $\alpha \in \mathbb{F}_2^n$ and $\gamma \in \mathbb{F}_2^m$, where α is the input mask and γ is the output mask of the function f, we can define the *Fourier coefficient* as

$$\widehat{f}(\alpha, \gamma) = \sum_{x \in \mathbb{F}_2^n} (-1)^{\langle \alpha, x \rangle + \langle \gamma, f(x) \rangle}.$$

The Fourier coefficient of a block cipher $\mathcal{E}_K$ can be used to approximate the bias of a linear approximation. However, it is often more convenient to work with the correlation $c_{\mathcal{E}_K}(\alpha, \gamma) = 2\epsilon_{\mathcal{E}_K}(\alpha, \gamma)$. We can then define the Fourier coefficient of a block cipher as

$$\widehat{\mathcal{E}_K}(\alpha, \gamma) = 2^n c_{\mathcal{E}_K}(\alpha, \gamma) = 2^{n+1} \epsilon_{\mathcal{E}_K}(\alpha, \gamma).$$

Yet, it is still computationally infeasible to compute the (exact) Fourier coefficient of a block cipher, we make use of the fact that nearly every block cipher is round based. Therefore we can calculate the *linear trail* of a block cipher, that is in comparison to differential cryptanalysis, the same as a *differential trail*.

Definition 20. Let $F_r : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ denote a vectorial Boolean function, and let $F = F_{r-1} \circ \cdots \circ F_0$. We can then define a tuple of the form

$$(\alpha_0, \ldots, \alpha_r) \in (\mathbb{F}_2^n)^{(r+1)}$$

as a *linear trail*, for the function F. Moreover, we refer to α_0 as input mask, and to α_r as output mask, respectively. Furthermore, we can call the masks α_i for $0 < i < r$ intermediate masks.

When using a *linear trail* in a distinguishing or key-recovery attack, we are interested in the probability that such a linear approximation holds. Therefore, we can define the *linear trail correlation*.

Definition 21. Let $F_r : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n$ denote a vectorial Boolean function, and let $F = F_{r-1} \circ \cdots \circ F_0$. Moreover, let $(\alpha_0, \ldots, \alpha_r)$ denote a r-round linear trail over F. We can then denote the *linear trail correlation* for $i = 0, \ldots, r-1$ rounds over F_r by

$$\mathrm{Corr}_{F_r}(\alpha_0, \alpha_r) = 2 \cdot \Pr_X[\langle X, \alpha_i \rangle = \langle F_i(X), \alpha_{i+1} \rangle] - 1,$$

where the probability is taken over $X \in \mathbb{F}_2^n$. Moreover, when we consider F_i as the round function of an iterated block cipher $\mathcal{E}_K$ the probability is also taken over the round keys.

Matsui [238] derived the *linear trail correlation* in his attacks against DES with a method called the *piling-up lemma*, where he assumes that the linear approximations of different rounds behave as independent Boolean random variables. We can then write the combined linear trail correlation over F as

$$\mathrm{Corr}_F(\alpha_0, \ldots, \alpha_r) = \prod_{i=0}^{r-1} \mathrm{Corr}_{F_i}(\alpha_i, \alpha_{i+1}).$$

Later, Nyberg [271] introduced the concept of *linear hulls* that are important tools to understand how the correlation is composed. Moreover, Nyberg also showed that Matsui in his attack against DES, actually also used *linear hulls* and not just *single linear trails*. This has also been shown for iterated block ciphers, with the concept of correlation matrices [117]. In comparison to differential cryptanalysis, we can think of *linear hulls* as the same concept as *differentials*. With *linear hulls* Nyberg basically showed that the Fourier coefficient of a block cipher $\mathcal{E}_K(x) = F(x, k)$ corresponds to the signed sum of the Fourier coefficients of a Boolean function $F : \mathbb{F}_2^n \times \mathbb{F}_2^m \to \mathbb{F}_2^n$, with an m-bit key k, over all possible masks for the key-input, *i.e.*,

$$2^m \widehat{\mathcal{E}_K}(\alpha, \gamma) = \sum_{\beta \in \mathbb{F}_2^m} (-1)^{\langle \beta, k \rangle} \widehat{F}((\alpha, \beta), \gamma)$$

Definition 22. Let α_0 and α_r be linear masks over a Boolean function F. We can define a *linear hull*, with α_0 as input mask and α_r as output mask, as a set $\mathcal{L}_r(\alpha_0, \alpha_r)$ that consists of the input mask α_0 and the output mask α_r, but with intermediate masks that can take any value.

For a linear attack, we are then further interested in the correlation of the linear hull over a block cipher $\mathcal{E}$.

Definition 23. Let $\mathcal{L}_r(\alpha_0, \alpha_r)$ denote a *linear hull* as a set with α_0 as input mask and α_r as output mask, and let Q be the linear trail. We can denote $\mathrm{Corr}_Q = |\mathrm{Corr}_Q^K|$ the correlation for the linear trail for any key, and $\mathrm{sign}(Q)$ is a sign bit indicating the sign of Corr_Q^K for the all-zero expanded key. We can then denote the correlation for a linear hull as

$$\mathrm{Corr}_{\alpha_0, \alpha_r}^K = \sum_{q0 = \alpha_0, qr = \alpha_r} (-1)^{Q \cdot K' \oplus \mathrm{sign}(Q)} \mathrm{Corr}_Q.$$

The correlation of linear hulls is highly dependent on the keys, which is further discussed in [90, 98, 209].
In a linear attack, an attacker is generally interested in determining a linear hull (α, γ) that maximizes the absolute correlation $|\mathrm{Corr}_{\mathcal{E}}(\mathcal{L}_r(\alpha, \gamma))|$. Similar as with differential cryptanalysis, there are some automated tools that can help an attacker with the search for optimal linear hulls such as [138, 317].

Distinguisher

Similar as in differential cryptanalysis, we can use linear trails for distinguishing attacks and based on that distinguisher we can mount key-recovery attacks. Lets again consider the scenario were an attacker interacts with an oracle $\mathcal{O}$ by providing queries with inputs and requesting an output from the oracle. The oracle $\mathcal{O}$ returns either the output of a block cipher $\mathcal{E}_K$ with a fixed key K, or returns the output of an ideal permutation π, whose outputs are sample uniform at random. The goal of the attacker is to distinguish if she is interacting with the block cipher $\mathcal{E}$ or with the random permutation π. The attacker therefore initializes two counters T_0 and T_1 to zero, and sends a plaintext P_i to the oracle. Then the attacker computes $\langle P_i, \alpha \rangle \oplus \langle C_i, \gamma \rangle$ and if the output is zero she increase T_0, otherwise T_1. If the output

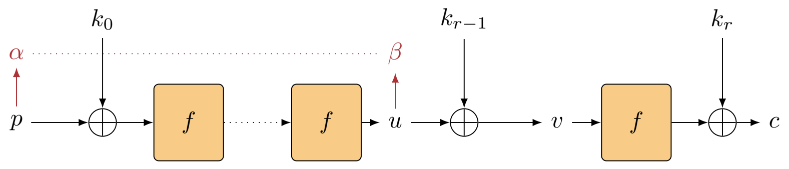


Figure 25: Example of an r-round key-recovery attack based on an $(r-1)$-round linear distinguisher.

comes from the ideal permutation π, we can expect that both T_0 and T_1 have values close to $N/2$, where N is the number of queries. However, if one of the counters is close to $1/2N(\mathrm{Corr}_{\mathcal{E}}+1)$ we assume the output stems from the block cipher $\mathcal{E}_K$.

Key-Recovery Attacks

Let $\mathcal{E}_K[r]$ be a r-round key-alternating block cipher, with a secret-key $K \in \mathbb{F}_2^\kappa$ consisting of an invertible round function f, so that the ciphertext C_i can be expressed as $C_i = f(x_i) \oplus k_r^*$, where x_i denotes the state at $(r-1)$ rounds and k_r^* is the actual final r^{th} round subkey. In a key-recovery attack, we use a $r-1$ round linear hull $\mathcal{L}_r(\alpha,\beta)$ with a high absolute correlation $|\mathrm{Corr}_{\mathcal{E}}(\mathcal{L}_r(\alpha,\beta))|$. In the following, we describe an r-round attack on the block cipher $\mathcal{E}_K[r]$.
Let's assume we found a $(r-1)$-round linear hull (α,β) for the block cipher $\mathcal{E}_K[r]$, with high absolute correlation, as outlined in Figure 25. We can turn the $(r-1)$-round linear hull into a key-recovery attack on r-rounds.

ATTACK STEPS. We assume a *known-plaintext attack (KPA)* scenario, where the attacker is able to obtain N ciphertexts C_i, that correspond to the encryption of N plaintext pairs P_i under a key $K \in \mathbb{F}_2^\kappa$ and $1 \leqslant i \leqslant N$.
The attack consists of the following steps:

1. Initialize two lists $\mathcal{K}_{r,0}, \mathcal{K}_{r,1}$ of 2^κ key counters and set them all to zero, for each possible guess of key k_r.
2. For each of the 2^κ guesses of k_r:
   2.1 Decrypt each ciphertext C_i for one round, *i.e.*, for $i = 1,\dots,N$ and compute $v = f^{-1}(C_i \oplus k_r)$.
   2.2 Compute $\langle P_i,\alpha\rangle \oplus \langle v,\beta\rangle$. If it equals zero, then increase the key counter in list $\mathcal{K}_{r,0}$, otherwise increase the counter in $\mathcal{K}_{r,1}$.
3. Output the key candidate $k' = \max\{\mathcal{K}_{r,0}, \mathcal{K}_{r,1}\}$.

Similar as in the differential cryptanalysis case, we can assume that for a wrong key guess, the distribution of the counters $\mathcal{K}_{r,0}, \mathcal{K}_{r,1}$ will be uniform at $N/2$, where N is the number of plaintexts queried to the oracle $\mathcal{O}$. By repeating the attack for many plaintext/ciphertext pairs (P_i, C_i), either one of the key counters $\mathcal{K}_{r,0}, \mathcal{K}_{r,1}$ will significantly derivate from $N/2$ allowing an attacker to uniquely identify the correct key.

2.4.5 Variants of Linear Cryptanalysis

Compared to differential cryptanalysis, there are not that many variants for linear cryptanalysis. One variant of linear cryptanalysis was proposed by Kaliski and Robshaw [193], that suggested to use *multiple linear approximations* for the same key bits. This allows an attacker to reduce the data complexity of an attack, as with each linear approximation we can get more suggested bits for the key-candidates. Moreover, as a counterpart of *impossible differentials*, Bogdanov and Rijmen introduced *zero-correlation linear attacks* [97], which we are going to explain in more details in the following.

Zero-Correlation Attacks

Zero-correlation linear cryptanalysis was introduced by Bogdanov and Rijmen [97] in 2014. Let α and β be the linear mask for the plaintext and ciphertext, then zero-correlations exploits the pair (α, β) with correlation exactly zero.

Definition 24. Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be a Boolean function, and let $\alpha, \beta \in \mathbb{F}_2^n$ be two linear masks. Moreover, let $\mathcal{L}_r(\alpha, \beta)$ be the linear hull, that is spanned by using α as the input mask and β as output mask. We say that the linear hull $\mathcal{L}_r(\alpha, \beta)$ has *zero-correlation*, if the correlation of the linear hull $\mathcal{L}_r(\alpha, \beta)$ is exactly zero.

One clear drawback of basic zero-correlation linear cryptanalysis is its huge data complexity. In order to detect that the correlation is exactly zero, it is a priori necessary to encrypt (almost) every possible message. Later, the data complexity was reduced by exploiting multiple or multidimensional zero-correlation linear approximations [96, 99]. When there are ℓ zero-correlation linear approximations for an n-bit block cipher, the required data complexity can be roughly estimated as $\mathcal{O}(2^n/\sqrt{\ell})$. Nevertheless, zero-correlation attacks have been successfully applied to many ciphers, such as [329] on round-reduced Sparx, [93] on round-reduced versions of Camellia and Clefia and [295] against round-reduced Skinny.
The main technique to derive zero-correlation linear approximations is very similar to deriving impossible differentials, that is a miss-in-the-middle approach. In a nutshell, starting with a given input and output mask, one propagates the input mask forward and the output mask backwards through the encryption (respectively decryption) process. This propagation usually does not capture all linear trails with non-zero correlation in both direction exactly as this might easily get very difficult to handle, but rather captures an easy to describe super-set of all those trails. The fact that the linear approximation is then derived by deducing that those supersets of forward and backward linear trails have an empty intersection. As an illustration, we recall the well known zero-correlation hull on 4 rounds of the AES [97]. Here, all bytes of the input mask are non-zero except for one diagonal, and the output linear mask is non-zero for only one byte. This then causes a contradiction in the second round `MixColumns` operation because of its branch number of 5.
Moreover, in Chapter 5 we use zero-correlation linear distinguishers to construct integral attacks on tweakable block ciphers, with applications to reduced-round versions of Qarma, Mantis and Skinny.

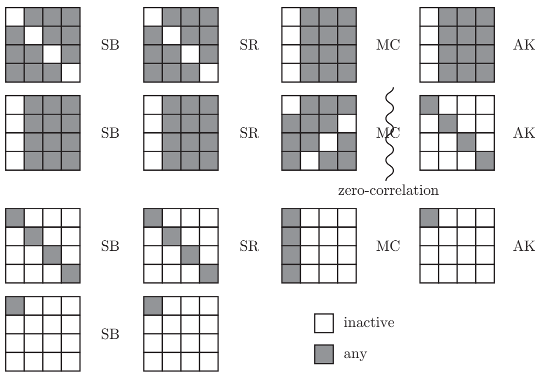


Figure 26: Zero-correlation linear hull on 4-round AES

2.4.6 Integral Cryptanalysis

Integral attacks were first introduced by Daemen *et al.* [118] as a dedicated attack against the block cipher Square and later extended to integral attacks [205]. These attacks have been shown extremely power against AES-like ciphers [139, 155, 321]. Similar as with *higher-order differentials*, in integral attacks an attacker prepares a Λ-set of chosen plaintexts so that particular cells of the state are held constant, while other cells vary trough all possible values.
Let a Λ-set be a set of ω states, where ω is the word size (in bits) of the block cipher, where all states are different in some bytes (*i.e.,* active), and equal in some other state bytes (*i.e.,* constant). Further properties of a Λ-set are defined below. Let λ denote the subset of active bytes in the Λ-set. Then it holds for the Λ-set that

$$\forall x, y \in \Lambda : \begin{cases} x_{i,j} \neq y_{i,j} \text{ for } (i,j) \in \lambda \\ x_{i,j} = y_{i,j} \text{ for } (i,j) \notin \lambda \end{cases}$$

Then an attacker considers some properties of the set, when propagating the set through several rounds of the cipher. The common properties used in standard integral attacks are:

- *Active (A):* The value of a cell takes all possible values in the set.
- *Constant (C):* The value of a cell is fixed to a constant value.
- *Unknown (?):* The value of a cell is unknown.
- *Balanced (B):* The XOR-sum of all values in a cell is zero.

There are different rules for the propagation of the integral properties through the components of an SPN-cipher. The linear layer consisting of simple word-rotations or shifts, just changes the positions in a multiset, but does not change the set itself. A multiplication with an MDS matrix, as in MixColumns in AES affects the output multiset just if there are more then one *active*, *balanced* or *unknown* word included. For example, if there are two or more *active* words the output multiset becomes *balanced*. Moreover, the propagation through an S-box changes the integral property.

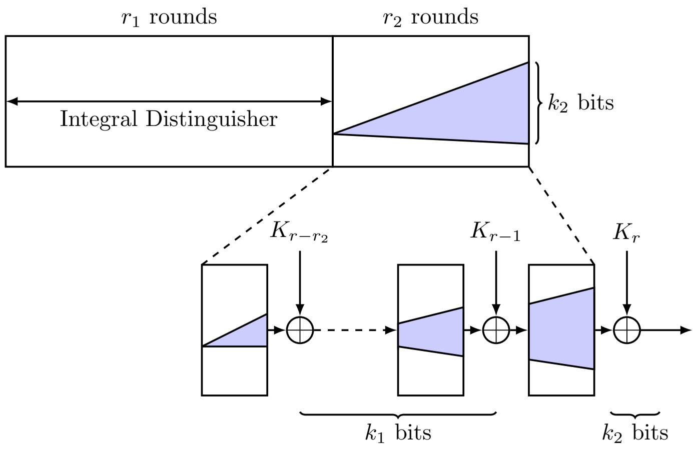


Figure 27: Key-recovery on a key-alternating cipher for integral attacks.

An *active* or *constant* input set through an S-box remains *active* or *constant*, respectively. However, an *balanced* input set becomes *unknown* when propagated through an S-box.

KEY-RECOVERY FOR INTEGRAL ATTACKS. After finding an integral distinguisher over several rounds, by propagating the input set through several rounds of the cipher until it reaches a state where there is at least one cell that is *balanced* left, we can mount an key-recovery attack based on the integral distinguisher. In a nutshell, this can be done by further propagating the *balanced* cell for a few rounds and then guessing round keys and calculating backwards to check if for a particular guess of an round key, the *balanced* property is still present.
In more details, lets consider a key-alternating cipher that can be expressed as:

$$C = K_r \oplus F_r(\cdots \oplus F_2(K_1 \oplus F_1(K_0 \oplus P)))$$

where, F_i is the round function of a block cipher (*i.e.,* a permutation), $K_0, K_1, \dots, K_r$ are the round keys, and C, P are the ciphertext, and the plaintext, respectively. The number of rounds, that are covered by an integral attack, can be split into two parts $r = r_1 + r_2$. Then, r_1 covers the integral distinguisher with N chosen plaintexts, and r_2 are the number of rounds when the balanced bits on the output of the distinguisher are diffused to k_2-bits by additional r_2 rounds. We can then guess k_1-bit keys, from $K_{r-r_2}, K_{r-r_2+1}, \dots, K_{r-1}$, and k_2-bit keys from key K_r, and calculate back r_2 rounds to check for an *balanced* state. The attack is further outlined in Figure 27.
Further improvements for the key-recovery have been published by Ferguson *et al.* that used a *partial-sum* technique [155], and by Todo and Aoki that developed the *FFT Key-Recovery* technique [324].

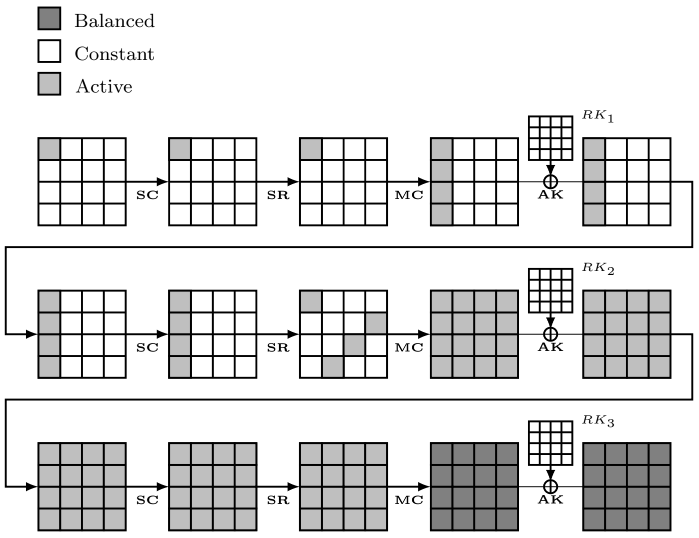


Figure 28: Integral distinguisher on 3-round AES.

INTEGRAL ATTACKS ON AES. Integral attacks were initially introduced by Knudsen as a dedicated attack against the block cipher SQUARE [118]. In the following, we give details on the attack applied to 4, 5 and 6 rounds of AES.
Figure 28 shows a 3-round integral distinguisher on AES. The attacker starts by preparing a set of plaintexts that has one *active* byte (*i.e.,* in our case at position 0, however, the position can be any), while the other 15 bytes remain *constant*. The first application of `SubBytes` does not change any of the bytes. `ShiftRows` does not shift the first row, so again nothing changes. The first application of `MixColumns` *activates* the whole first row. The final key addition does not change the set. In the second round, `ShiftRows` shifts the *active* row so that there is an *active* cell in each column. *MixColumns* then *activates* the entire state. In the third round, *MixColumns* transforms the entire *active* state to an entire *balanced* state. In the fourth round, the application of the S-box layer change the entire set to become *unknown*. We can turn the three round integral distinguisher in a 4, 5, and 6 round key-recovery attack.
For the four round attack we can consider the rounds:

$$4 \text{ round attack} = \texttt{AddRoundKey} + 3 \text{ round distinguisher} + \text{final round},$$

as outlined in Figure 29. The initial `AddRoundKey` does not change the set of plaintexts. The attack works as follows:

1. Encrypt a Λ-set with one *active* position.
2. For each of the 2^8 guesses of RK_4:
   - 2.1 Decrypt each ciphertext pair (C_i, C_i') for one round to the output of round 3.
   - 2.2 Verify the *balanced* property.
     - i. For a correct guess of the key, the *balanced* property must hold.

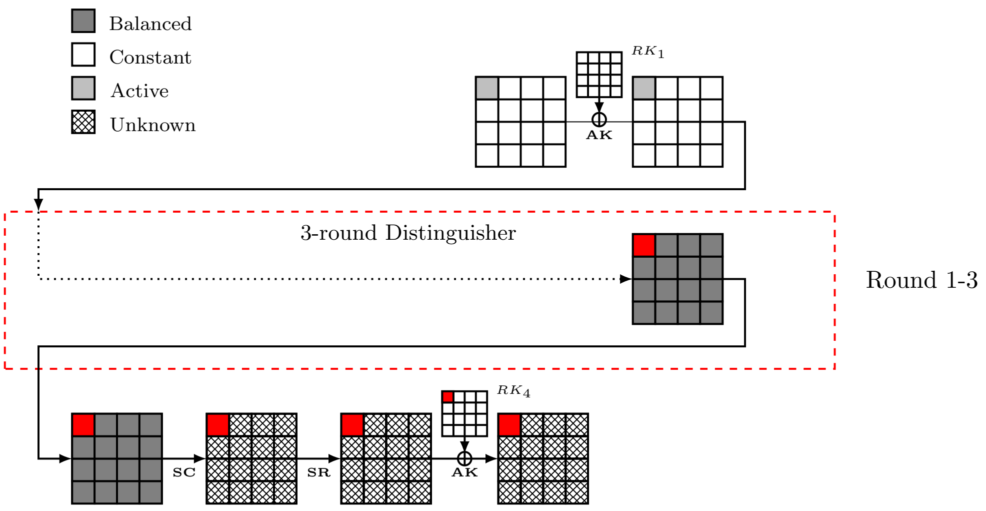


Figure 29: Integral attack on 4 round AES.

ii. For an incorrect guess of the key, the *balanced* property holds with a probability of 1/256.

We can repeat the attack for all 16 positions in the state, and then invert the key-schedule to recover the master key.
For the five round attack we can consider the rounds:

$$5 \text{ round attack} = \texttt{AddRoundKey} + 3 \text{ round distinguisher} + 1 \text{ round} + \text{final round},$$

as outlined in Figure 30. An attacker guesses the row-shifted column of the key in round 5, and then decrypts 1 byte of the output of round 4. Then the attacker can simply apply the 4 round attack as described above. The attack requires approximately 6Λ-sets and 2^{40} steps.
For the six round attack we can consider the rounds:

$$6 \text{ round attack} = \texttt{AK} + 1 \text{ round} + 3 \text{ round distinguisher} + 1 \text{ round} + \text{final round},$$

as outlined in Figure 31. An attacker guesses the inverse-row shifted column of the initial key RK_0. Then the attacker chooses plaintexts, such that the output of round 1 forms a Λ-set. Therefore, the attacker needs 2^{32} plaintexts for one column, that forms a Λ set after 1-round. Again, the attacker can apply the previous 5 round attack, as outlined above. The attack requires approximately $2^{35}\Lambda$-sets and a complexity of 2^{72} steps. The 6-round attack can be improved by the *partial-sum* technique of Ferguson *et al.* [155] that improves the time complexity to 2^{45} computations.

2.4.7 Variants of Integral Cryptanalysis

Since *integral cryptanalysis* has been firstly published as a dedicated attack against the block cipher SQUARE, it has been further developed. Compared to the *square attack*, in *integral cryptanalysis* we also consider backward propagation from the start

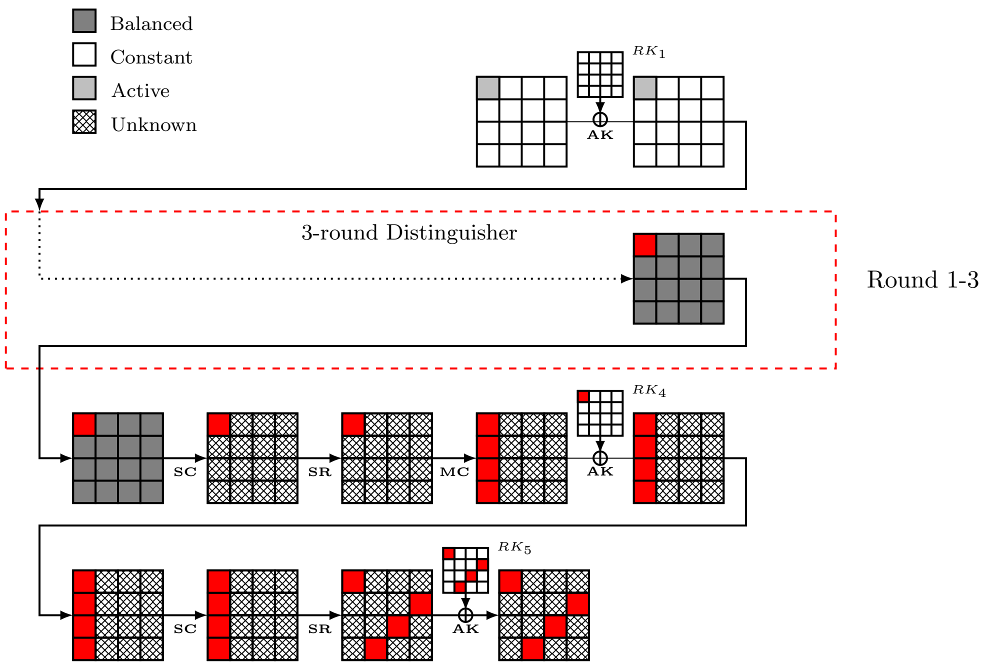


Figure 30: Integral attack on 5 round AES.

of the integral distinguisher. Another improvement is the *division property* that additionally considers the algebraic degree of an S-box and can be seen as a combination of *integral attacks* and *higher-order differentials*.

Division Property

The division property is a generalisation of integral attacks and higher-order differential distinguishers, and has recently been introduced by Todo [322]. Later, the division property was used in an attack on full Misty-1 by Todo [321]. The division property was initially proposed against word-based ciphers, but further extended to the bit-based division property by Todo and Morii [326].
The division property of a multi-set uses bit-product functions over $\mathbb{F}_2^n$.

Definition 25 ([322])**.** Let $\pi_u : \mathbb{F}_2^n \to \mathbb{F}_2$ be a bit-product function for any $u \in \mathbb{F}_2^n$. For any input $x \in \mathbb{F}_2^n$ $\pi_u(x)$ is the product of $x[i]$ satisfying that $u[i] = 1$, *i.e.*, we can define

$$\pi_u(x) = \prod_{i=1}^{n} x[i]^{u[i]}.$$

Note, we interpret $0^0 = 1$, for correctness.

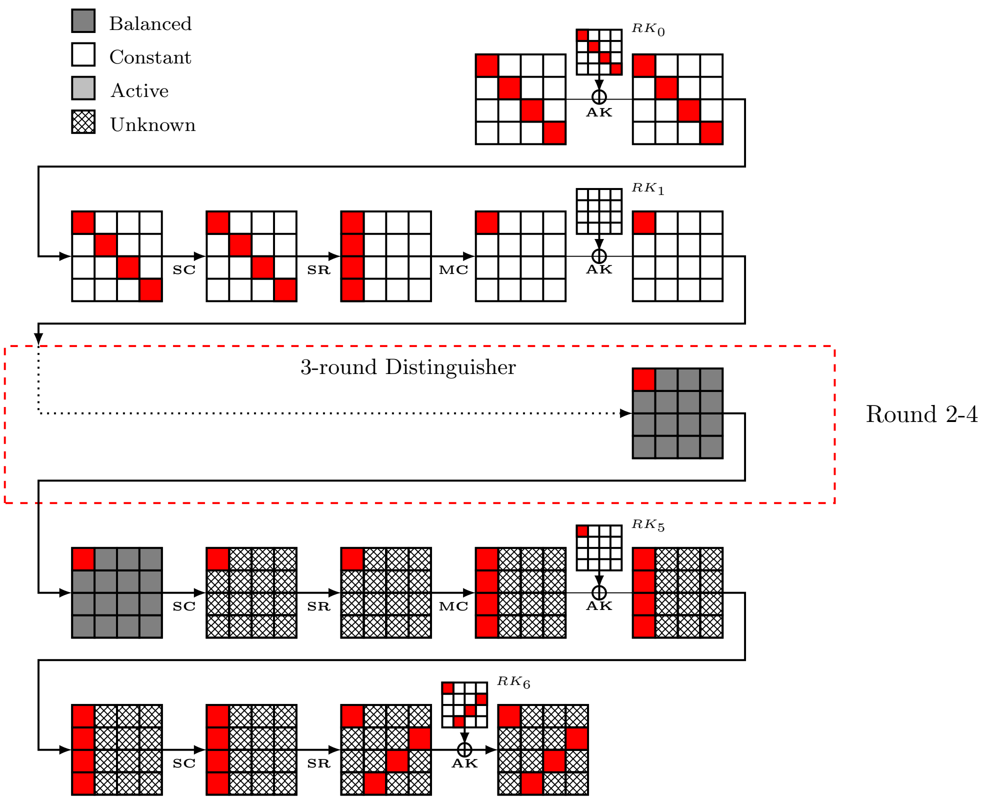


Figure 31: Integral attack on 6-round AES.

Definition 26 ([322])**.** Let $\pi_{\mathbf{u}} : (\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m}) \to \mathbb{F}_2$ be a bit-product function for any $\mathbf{u} \in (\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m})$. For any input $\mathbf{x} \in (\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m})$ $\pi_{\mathbf{u}}(\mathbf{x})$ is defined as

$$\pi_{\mathbf{u}} = \prod_{i=1}^{n} \pi_{u_i}(x_i).$$

The division property makes also used of the algebraic structure of an S-box. Therefore, we can further define the *algebraic degree* of an S-box, which is represented as the highest exponent in the univariate polynomials of the *Algebraic Normal Form (ANF)* of an S-box.

Definition 27 ([322])**.** Let $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be a Boolean function, then the ANF is represented as

$$f(x) = \bigoplus_{u \in \mathbb{F}_2^n} a_u^f \left(\prod_{i=1}^{n} x[i]^{u[i]} \right) = \bigoplus_{u \in \mathbb{F}_2^n} a_u^f \pi_u(x),$$

where $a_u^f \in \mathbb{F}_2$ is a constant value depending on f and u.

Compared to standard integral attacks, the division property further exploits the algebraic degree of an S-box. If we consider an input multiset $\mathbb{X}$ that is *active*, then

Table 5: Example calculation of $\pi_u(x)$ for input set $\mathbb{X} = \{0x0, 0x3, 0x3, x3, 0x5, 0x5, 0x6, 0x8, 0xB, 0xD, 0xE\}$.

	0x0 0000	0x3 0011	0x3 0011	0x3 0011	0x5 0101	0x6 0110	0x8 1000	0xB 1011	0xD 1101	0xE 1110	$\sum \pi_u(x)$ ($\bigoplus \pi_u(x)$)
u=0000	1	1	1	1	1	1	1	1	1	1	10 (0)
u=0001	0	1	1	1	1	0	0	1	1	0	6 (0)
u=0010	0	1	1	1	0	1	0	1	0	1	6 (0)
u=0011	0	1	1	1	0	0	0	1	0	0	4 (0)
u=0100	0	0	0	0	1	1	0	0	1	1	4 (0)
u=0101	0	0	0	0	1	0	0	0	1	0	2 (0)
u=0110	0	0	0	0	0	1	0	0	0	1	2 (0)
u=0111	0	0	0	0	0	0	0	0	0	0	0 (0)
u=1000	0	0	0	0	0	0	1	1	1	1	4 (0)
u=1001	0	0	0	0	0	0	0	1	1	0	2 (0)
u=1010	0	0	0	0	0	0	0	1	0	1	2 (0)
u=1011	0	0	0	0	0	0	0	1	0	0	1 (1)
u=1100	0	0	0	0	0	0	0	0	1	1	2 (0)
u=1101	0	0	0	0	0	0	0	0	1	0	1 (1)
u=1110	0	0	0	0	0	0	0	0	0	1	1 (1)
u=1111	0	0	0	0	0	0	0	0	0	0	0 (0)

the output multiset is also *active* propagated through an S-box. Moreover, if an input multiset $\mathbb{X}$ is *balanced* then the output multiset becomes *unknown*. Additionally, if we consider an input multiset with 2^{d+1} chosen texts, the output multiset becomes *balanced*, as the $(d+1)^{st}$ derivative of an S-box with degree d is zero. In standard integral cryptanalysis this property is not further exploited.

Definition 28 (Division property [322])**.** Let $\mathbb{X}$ be a multiset where all elements in the set take a value of $(\mathbb{F}_2^{n_1} \times \mathbb{F}_2^{n_2} \times \cdots \times \mathbb{F}_2^{n_m})$. When the multiset has the *division property* $\mathcal{D}_{\mathbb{K}}^{n_1,n_2,\ldots,n_m}$, where $\mathbb{K}$ denotes a set of m-dimensional vectors whose i^{th} element takes a value between 0 and n_i, it fulfils the following conditions:

$$\bigoplus_{x \in \mathbb{X}} \pi_{\mathbf{u}}(\mathbf{x}) = \begin{cases} \text{unknown}, & \text{if there are } \mathbf{k} \in \mathbb{K} \text{ s.t. } wt(\mathbf{u}) \succeq \mathbf{k}, \\ 0, & \text{otherwise.} \end{cases}$$

Lets consider an example to show how the division property is calculated for an input set $\mathbb{X} \in \mathbb{F}_2^4$. Let the input set $\mathbb{X}$ be:

$$\mathbb{X} = \{0x0, 0x3, 0x3, x3, 0x5, 0x5, 0x6, 0x8, 0xB, 0xD, 0xE\}.$$

From the example shown in Table 5, we can see that all u satisfying that the *hamming weight* $w(u) < 3$ and the XOR-sum $\bigoplus_{x \in \mathbb{X}} \pi_u(x) = 0$. Therefore, the multiset $\mathbb{X}$ has division property $\mathcal{D}_3^4$.

The division property can then be used to construct integral distinguishers, where the propagation of the division property through the different components of an SPN cipher are explained in the following. Let's consider an (l, d, m)-SPN cipher, where l is the bit-size of the S-boxes, m is the number of S-boxes in the S-box layer, and d is the algebraic degree of an S-box. We can evaluate the propagation of the division property through the different components of an SPN cipher as follows:

- *S-box Layer:* Assume that the input multiset of the S-box layer has division property $\mathcal{D}^{l,m}_{\mathbf{k}}$. As the S-box layer consists of $m \times l$-bit S-boxes with algebraic degree d, the output set of the S-box layer has division property $\mathcal{D}^{l,m}_{\mathbf{k'}}$, where $k'_i = \lceil k_i/d \rceil$ if $k_i < l$ and $k'_i = l$ if $k_i = l$.
- *Concatenation:* Since the output of the S-box layer is a value of the form $(\mathbb{F}_2^l)^m$ but the input of the Diffusion layer is of the form $\mathbb{F}_2^{lm}$ we have to convert in between. Let $\mathbb{X}$ be the output of the S-box layer and $\mathbb{Y}$ be the input of the Diffusion layer. The parity of $\pi_v(y)$ for all $y \in \mathbb{Y}$ becomes unknown if and only if we choose v satisfying the hamming weight $w(v) \geqslant \sum_{i=1}^{m} k_i$. Therefore, it holds that the input set $\mathbb{Y}$ of the Diffusion layer has the division property $\mathcal{D}^{lm}_{k'}$.
- *Diffusion Layer:* The diffusion layer normally consists of an (l, m)-bit linear function. As the algebraic degree of an linear function is exactly 1, there is no change in the division property.
- *Partition:* We have a similar problem of converting the format of the set between Diffusion layer back to the S-boxes layer. Again, let $\mathbb{X}$ be the output of the Diffusion layer and $\mathbb{Y}$ be the input of the S-boxes layer. When the output set $\mathbb{X}$ of the Diffusion layer has division property $\mathcal{D}^{l,m}_{k}$, the sufficient condition that the parity of $\pi_u(x)$ for all $x \in \mathbb{X}$ becomes unknown is $k \leqslant w(u)$. Therefore, the input set $\mathbb{Y}$ of the S-box layer has the collective division property $\mathcal{D}^{l,m}_{\mathbf{k'^{(1)}},\mathbf{k'^{(2)}},\dots,\mathbf{k'^{(q)}}}$, where q denotes the number of all possible vectors satisfying $k'^{(j)}_1 + k'^{(j)}_2 + dots + k'^{(j)}_m = k$ for $1 \leqslant j \leqslant q$.

We can then construct integral distinguishers using the division property by evaluating the propagation characteristic of the collective division property.

2.4.8 Yoyo Cryptanalysis

Yoyo attacks are closely related to boomerangs attacks. In both techniques, the adversary first lets the oracle encrypt chosen-texts, and then observes the corresponding encryptions and adaptively chooses new ciphertexts, that are then decrypted in the hope for a certain property in their corresponding plaintexts. Initially, yoyo attacks have been proposed by Biham *et al.* on SKIPJACK-3XOR [68], and have been revived by Biryukov *et al.* [79] for analysing Feistel networks with secret round functions. Recently, Rønjom *et al.* [290] showed yoyo-based distinguishers and key-recovery attacks on generic SPNs and applications to round-reduced AES.
Assume, $E : \mathbb{F}_2^{m \cdot b} \to \mathbb{F}_2^{m \cdot b}$ is a permutation over elements of $n = m \cdot b$ bits that can be divided into m words of b bits each. The core idea of yoyo attacks is to encrypt chosen-plaintext pairs (P^0, P^1) with a certain input difference through E

and collect the corresponding ciphertexts (C^0, C^1). The cryptanalyst crafts a set of new ciphertexts from the recombination of words from C^0 and C^1. Let $C^0[i]$ denote the i-th b-bit word of C^0. Let $v = (v_0, \dots, v_{m-1})$ denote a Boolean vector, *i.e.*, $v \in \mathbb{F}_2^m$, and let $\rho : \mathbb{F}_2^{m \cdot b} \times \mathbb{F}_2^{m \cdot b} \times \mathbb{F}_2^m \to \mathbb{F}_2^{m \cdot b}$ be defined as

$$\rho\left(C^0, C^1, v\right) \overset{\text{def}}{=} (C^{v_0}[0] \| \dots \| C^{v_{m-1}}[m-1]) .$$

Given $v \neq (0, 0, \dots, 0)$, one can derive two modified ciphertexts $C^2 = \rho(C^0, C^1, v)$ and $C^3 = \rho(C^0, C^1, \bar{v})$, where $\bar{v}$ denotes the vector of the element-wise inverse Boolean vector, *i.e.*, $\bar{v}_i = 1 \oplus v_i$ for $0 \leqslant i < m$. Then, C^2 consists of the words of C^0 at all positions i where $v_i = 0$ and of the words of C^1 at the remaining positions. The situation is vice versa for C^3. The cryptanalyst then queries the so-derived ciphertexts to obtain the corresponding plaintexts (P^2, P^3). The core observation is that the vector v defines a word-wise difference pattern: C^2 differs from C^0 in exactly the words where $v_i = 1$ holds. This difference pattern is preserved through any partial map of E that operates on the b-bit words separately, *e.g.*, an S-box layer in an SPN.

2.5 DESIGN PRINCIPLES

In this section, we discuss a few design principles and properties of Boolean functions. In general, we can define every symmetric cryptographic primitive as a Boolean function and therefore we can also inherit the properties of Boolean functions for symmetric cryptographic primitives. Concretely, we focus in this section on cryptographic S-boxes and give a more detailed overview on energy-efficient S-boxes in Chapter 4.

2.5.1 Boolean functions

Definition 29. A Boolean Function of n variables is a function $f : \mathbb{F}_2^n \to \mathbb{F}_2^n$. The set of all Boolean functions is denoted as $\mathcal{BF}_n$, which has cardinality $|\mathcal{BF}_n| = 2^{2^n}$.

Definition 30. The Hamming weight $w_H(x)$ of a binary vector $x \in \mathbb{F}_2^n$ is the number of non-zero coordinates. The Hamming weight $w_H(f)$ of a Boolean function f over $\mathbb{F}_2^n$ is the size of the support of the function, where the support is defined as $\mathrm{Supp}(f) \leftarrow \{x \in \mathbb{F}_2^n \mid f(x) \neq 0\}$.

A Boolean function is usually defined by its truth table that gives the images of all its possible inputs in $\mathbb{F}_2$. However, in cryptography and coding theory the natural representation is the n-variable polynomial representation over $\mathbb{F}_2$, also called *Algebraic Normal Form* (ANF):

Definition 31. Let f be a Boolean function of n variables. Then there exists a unique multivariate polynomial in $\mathbb{F}_2[x_1, \dots, x_n]/(x_1^2 + x_1, \dots, x_n^2 + x_n)$ such that

$$f(x_1, \dots, x_n) = \bigoplus_{I \in \mathcal{P}(N)} a_I \left(\prod_{i \in I} x_i \right) = \bigoplus_{I \in \mathcal{P}(N)} a_I x^I ,$$

where $\mathcal{P}(N)$ denotes the power set of $N = \{1, \dots, n\}$.

In lightweight cryptography, we are interested in reducing the computational overhead. This can mathematically be achieved by designing involutory functions, that enable the use of the same function for encryption and decryption.

Definition 32. Let f be a function. Then f is an involution, or an involutory function if and only if it is its own inverse such that:

$$f(f(x)) = x,$$

for all x in the domain of f.

The Fourier transform of the *sign function* $f_\chi = (-1)^{f(x)}$ is the *Walsh transform*.

Definition 33. Let f be a Boolean function. Then $\widehat{f_\chi}(u)$ is called the Walsh coefficient if:

$$\widehat{f_\chi}(u) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x) \oplus x \cdot u}$$

2.5.2 Cryptographic S-boxes

Many cryptographic primitives use Boolean functions as building blocks. In Substitution Permutation Network (SPN) ciphers, the S-boxes are a set of m Boolean functions from $\mathbb{F}_2^n \to \mathbb{F}_2^m$ (with n inputs and m outputs) of n variables, defined as the coordinates of f.

Definition 34. Let S be an S-box from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$. The Boolean functions, also called coordinates, of S are n-variable Boolean functions such that:

$$S_\lambda : x \rightarrowtail \lambda \cdot S(x)$$

for any $\lambda \in \mathbb{F}_2^m$.

Cryptographic S-boxes require further properties, such as we are only interested in *non-linear* Boolean functions. Moreover, we want an S-box to be invertible (i.e. for decryption) so we require an S-box to be a bijection.

Definition 35 ([108, page 24])**.** Let S be an S-box from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$. S is a permutation if and only if all its non-trivial coordinate functions are balanced.

Cryptographic functions must be *balanced* functions to avoid any statistical dependence between input and output. That is the output must be uniformly distributed. A Boolean function f is balanced if and only if $\widehat{f_\chi}(0) = \mathcal{F}(f) = 0$.

Property 1 ([111, page 24])**.** *Let f be a Boolean function from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$. Then f is a balanced function if and only if:*

$$\widehat{f_\chi}(0) = \mathcal{F}(f) = 0,$$

where $\widehat{f_\chi}(u) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x) \oplus x \cdot u}$ is the Walsh transform of f, and $\mathcal{F}(f) = \widehat{f_\chi}(0) = \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x)} = 0$.

An even stronger relation was introduced by Siegenthaler [311] who defined that a Boolean function f has to be m-resilient, then there exists no correlation between the output of a function and (at most) m coordinates of its input.

Definition 36 ([111, page 56], [311, page 777])**.** Let f be a Boolean function from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$. Than f is called an m-resilient function, if any of it restrictions obtained by fixing at most m of its input coordinates x_i is balanced.

Cryptographic functions need to have a high algebraic degree. Algebraic attacks exploit low algebraic degrees of an S-box or low degrees on a non-trivial component function. The algebraic degree of an S-box is usually defined by the algebraic degrees of its coordinate functions.

Definition 37. Let S be an S-box from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$ and let $f_1, \dots, f_n$ be its coordinate functions. Then if we consider the boolean functions in algebraic normal form, as in Definition 31, then the (algebraic) degree of f is defined as:

$$\deg f = \max\{w_H(I) : I \in \mathbb{F}_2^n, a_I \neq 0\},$$

The (algebraic) degree for an S-box is then the maximal algebraic degree of its coordinate functions.

The branch number of a Boolean function f defines the minimum number of active S-boxes at the input and output of f. Moreover, it can be used to measure diffusion, especially for linear and differential trails as defined by Daemen and Rijmen [121].

Definition 38 ([121, page 131])**.** Let f be a Boolean function. Then the differential branch number of f is defined by:

$$\mathcal{B}_d(f) = \min_{a,b \neq a} \{w_H(a \oplus b) + w_H(f(a) \oplus f(b))\}.$$

Similar to the differential branch number, we can further define the linear branch number.

Definition 39 ([121, page 132])**.** Let f be a Boolean function. Then the linear branch number of f is defined by:

$$\mathcal{B}_l(f, \alpha) = \min_{\alpha,\beta,C(\alpha^T x, \beta^T f(x)) \neq 0} \{w_H(\alpha) + w_H(\beta)\}.$$

2.5.3 Properties of cryptographic S-boxes

Cryptographic S-boxes play an important role in the resistance against attacks such as differential and linear cryptanalysis.
Differential Cryptanalysis has been introduced by Biham and Shamir [74]. It basically studies the propagation of a difference between two plaintexts over several rounds in a block cipher. The resistance against differential cryptanalysis can be quantified as the maximal probability that a given non-zero input difference leads to a given output difference for an S-box and is determined by the *differential uniformity* of the S-box.

Definition 40 ([270, page 58])**.** Let S be an S-box from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$. For any $a, b \in \mathbb{F}_2^n$ we can define:

$$\delta(\Delta_{in}, \Delta_{out}) = \#\{x \in \mathbb{F}_2^n \mid S(x \oplus \Delta_{in}) \oplus S(x) = \Delta_{out}\}$$

and the maximum

$$\delta_S = \max_{a \neq 0, b} \delta(a, b)$$

is the *differential uniformity* of S.

The minimum differential uniformity that can be achieved was shown by Nyberg and Knudsen [272] is $\delta(S) \geqslant 2$, where S is an S-box from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$. For S-boxes that fulfil this equation with equality are called *Almost Perfect Non-linear* (APN) functions.
The differential probability of an S-box measures the probability for a given non-zero input difference leads to a given output difference for an S-box.

Definition 41. Let S be an S-box from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$. For any $a, b \in \mathbb{F}_2^n$ we can define the differential probability as:

$$DP(a \to b) = 2^{-n} \cdot \delta(a, b)$$

Linear cryptanalysis has been first introduced by Matsui [238]. In a nutshell, it is based on finding *affine approximations* for the non-linear components of a cipher. The resistance against linear cryptanalysis can be determined with the linearity of a cipher which evaluates the deviation of a Boolean function from being linear or affine.

Definition 42 ([269, page 93])**.** Let S be an S-box from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$. Then the linearity of the S-box can be defined as:

$$\mathcal{L}(S) = \max_{a \in \mathbb{F}_2^n, b \in \mathbb{F}_2^n \setminus \{0\}} |\mathcal{L}(S_b(a))| = \max_{a \in \mathbb{F}_2^n, b \in \mathbb{F}_2^n \setminus \{0\}} \left| \widehat{S_b}(a) \right|$$

where $\widehat{S_b}(a)$ is the Walsh coefficient from S.

A lower bound for the linearity of a Boolean function can be derived using Parseval's relation $\sum_{a \in \mathbb{F}_2^n} (\widehat{f_X}(u))^2 = 2^{2n}$ and is bound by $\mathcal{L}(f) \geqslant 2^{\frac{n}{2}}$. Boolean functions that fulfil this equation with equality were firstly shown by Rothaus [291]. Those functions are called *bent functions* and exist only for even n and are not balanced.
The *linear probability bias* (ϵ) of an S-box measures the correlation and gives the deviation from a uniform probability regarding affine or linear functions.

Definition 43. Let S be an S-box from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$. Then the linear probability bias of S can be defined as:

$$\epsilon \leqslant \left| \frac{\mathcal{L}(S)}{2^{n+1}} \right|.$$

When designing a cryptographic primitive, it is important to reduce the number of possible S-boxes and consider just S-boxes with similar cryptographic properties to study further implementation relevant properties. Some well known classification tools are the notion of linear and affine equivalence.

Definition 44 ([128, page 76])**.** Let S_1, S_2 be two invertible S-boxes from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$. Let $A, B \in GL_n(\mathbb{F}_2^n)$ be two invertible $n \times n$ matrices. Then S_1 is *linear equivalent* to S_2 if the following holds:

$$S_1(x) = B^{-1} \cdot S_2(A \cdot x)$$

for all $x \in \mathbb{F}_2^n$.

The notion of *affine equivalence* even further reduces the number of different equivalence classes.

Definition 45 ([128, page 78])**.** Let S_1, S_2 be two invertible S-boxes from $\mathbb{F}_2^n$ to $\mathbb{F}_2^n$. Let $A, B \in GL_n(\mathbb{F}_2^n)$ be two invertible $n \times n$ matrices and let $a, b \in \mathbb{F}_2^n$. Then S_1 is *affine equivalent* to S_2 if the following holds:

$$S_1(x) = B^{-1} \cdot S_2(A \cdot x \oplus a) \oplus b$$

for all $x \in \mathbb{F}_2^n$.

Part I

Foundations

3 DIFFERENTIAL CRYPTANALYSIS OF LIGHTWEIGHT BLOCK CIPHERS

CONTENTS

EXECUTIVE SUMMARY. In this chapter, we study the effects of differential cryptanalysis of several recently proposed lightweight block ciphers. Resistance against differential cryptanalysis is an important design criterion for any modern block cipher and most designs rely on finding some upper bound on probability of single differential trails. However, already at EUROCRYPT'91, Lai *et al.* conjectured that differential cryptanalysis rather uses *differentials* instead of single *trails*.
We consider exactly the gap between these two approaches and investigate this gap in the context of recent lightweight cryptographic primitives. This shows that for many recent designs like Midori, Skinny or Sparx one has to be careful as bounds from counting the number of active S-boxes only give an inaccurate evaluation of the best differential distinguishers. For several designs we found new differential distinguishers and show how this gap evolves. We found an 8-round differential distinguisher for Skinny-64 with a probability of $2^{-56.93}$, while the best single trail only suggests a probability of 2^{-72}. Our approach is integrated into publicly available tools and can easily be used when developing new cryptographic primitives.
Moreover, as differential cryptanalysis is critically dependent on the distribution over the keys for the probability of differentials, we provide experiments for some of these new differentials found, in order to confirm that our estimates for the probability are correct. While for Skinny-64 the distribution over the keys follows

a Poisson distribution, as one would expect, we noticed that SPECK-64 follows a bimodal distribution, and the distribution of MIDORI-64 suggests a large class of weak keys.

DECLARATION OF AUTHORSHIP. The work described in this chapter is based on the paper [24]: *Mind the Gap - A Closer Look at the Security of Block Ciphers against Differential Cryptanalysis* and was presented at *The 25th Conference on Selected Areas in Cryptography (SAC'18)* in Calgary, Canada. The paper is joint work with Stefan Kölbl. All authors contributed equally to the results of the paper. The contributions of the author are the following:

- Implementation of several block ciphers in an automated search tool.
- Search for single differential trails and differentials for several block ciphers.
- Adapting/Adding new functionalities to the differential search tool (*i.e.,* support for S-boxes).

3.1 INTRODUCTION

Differential cryptanalysis, first published by Biham and Shamir [73] to analyse the DES, has become one of the prime attack vectors which any modern symmetric-key primitive has to be resistant against. The idea behind differential cryptanalysis is to find pairs of plaintexts and ciphertexts, where a certain difference between those texts occurs with high probability. The challenge for a cryptanalyst consists of finding differences with a high probability or to show that no such difference exists. A popular approach is to design a cipher in such a way that one can find a bound on the best differential trails, either directly *e.g.,* the wide-trail strategy deployed in AES or using methods based on Matsui's algorithm, MILP or SAT.
A differential trail specifies all the intermediate differences after each round of the primitive. However, when constructing a differential distinguisher one only cares about the input and output difference. It is often assumed that a single trail dominates the probability of such a differential, however this is not true in general and leads to imprecise estimates of the probability in many cases [75, 137].
In the work by Lai *et al.* [215] they showed that for an iterated cipher, the round function satisfies that the differential probability is independent of the choice of one of the component plaintexts under a difference, it can be considered a Markov cipher. Moreover, for a Markov cipher that has independent round-keys, the sequence of round differences forms a Markov chain. As differential cryptanalysis considers just the first and last difference and ignores the intermediate values, the probability of such a *differential* can then be computed as the sum of all trails, that are formed by the differentials. While this assumes that the rounds are independent, it provides a more precise estimate and the probability of the most probable *differential* will always be greater than the probability of the most probable *trail*.
We provide a broad study covering different design strategies and investigate the differential gap between single *trails* and *differentials* for the block ciphers LBLOCK, MIDORI, PRESENT, PRINCE, RECTANGLE, SIMON, SKINNY, SPARX, SPECK and TWINE. In

order to do this, we use an automated approach for enumerating the trails with the highest probability contributing to a differential based on SMT solvers [256], which we adopt to different design strategies. This allows us to efficiently enumerate a large set of trails contributing to the probability of a differential resulting in a precise estimate for the probability of differentials.
For Skinny-64 we present an 8-round differential distinguisher with a probability of $2^{-56.93}$, while the best single trail only suggests a probability of 2^{-72}. For Midori-64 we show that the best trail for 8 rounds, with a probability of 2^{-76} can be used to find a differential with a probability of $2^{-60.86}$. Our results show that in the case of many new lightweight ciphers like Midori-64, Skinny-64, and Sparx-64 the probabilities improve significantly and that we can find differential distinguishers which are able to cover more rounds. This suggests that one should be particularly careful with lightweight block ciphers when using simpler approximations like counting the number of active S-boxes.
Our method is generic and can easily be applied to other designs as one only needs to describe the differential behaviour of the round function and can re-use all the components we implemented for doing so. This allows both to find optimal differential trails and to enumerate all trails contributing to a differential.
Furthermore, we provide experiments to verify that our estimates of the differential probability provide a good approximation. However, we also noticed that the distribution over the choice of keys varies significantly for some design strategies and that commonly made assumptions do not hold for reduced-round versions. While for Skinny-64 the distribution over the keys follows relatively closely what one would expect we noticed that for Midori-64 for a large class of keys there are no pairs following the differential at all, while for very few keys the probability is significantly higher.

RELATED WORK. Daemen and Rijmen firstly studied the probability of differentials for AES in their work on Plateau Characteristics [122]. In their work, they analysed AES on the distribution of differential probability over the choice of keys and showed that all 2-round trails have either a zero probability or for a small subset of keys the probability is non-zero. However, they only considered AES, but further proved that other ciphers with 4-uniform S-boxes will show a similar result. In the case of AES and AES-like ciphers, there has also been a lot of research in studying the expected differential/linear probability (MEDP/MELP) [119, 196], that is used to provable bound the security of a block cipher against differential/linear cryptanalysis.
In recent years, many automated tools were proposed that could help designers to prove bounds against differential/linear attacks. Mouha *et al.* [257] used Mixed Integer Linear Programming (MILP) to count active S-boxes and compute provable bounds. Furthermore, there have been a few approaches of using automated tools to find optimal trails, and to collect many trails with the same input/output differences. This idea was first introduced by Sun *et al.* [317] who used MILP. Likewise, tools using SAT/SMT solvers are used where the results were applied to Salsa-20 [256], Norx [32], and Simon [206].
Moreover, there exist several design and attack papers that study the effect of numerous trails contributing to the probability of a differential: Mantis [137],

Noekeon [190], Salsa [256], Simon/Speck [83, 206], Rectangle [361] and Twine [75]. Yet, these are often based on truncated differentials or dedicated algorithms for finding large numbers of trails. For example in [151], Eichlseder and Kales attack Mantis-6 by finding a large cluster of differential trails. Contrary to the attack on Mantis-5 by Dobraunig *et al.* [137] where the cluster was found manually, in the attack on Mantis-6, Eichlseder and Kales used a tool based on truncated-differentials. Similar effects have also been observed in the case of linear cryptanalysis, where Abdelraheem *et al.* [3] showed that the security margins based on the distribution of linear biases are not always accurate. Their work has further been studied and improved by Blondeau and Nyberg [89].

software. All the models for enumerating the differential trails are publicly available at `https://github.com/TheBananaMan/cryptosmt`.

differentials and differential trails A detailed description about differential cryptanalysis can be found in Chapter 2.4.2.

3.2 DESIGN STRATEGIES FOR LIGHTWEIGHT BLOCK CIPHERS

We arrange the ciphers in several classes according to different design strategies. In general one can distinguish between two main design families for block ciphers, *Substitution-Permutation Networks (SPN)* and *Feistel Networks*. Within these families we can gather ciphers according to other structural properties. These are for SPN ciphers: *AES-like*, *Bit-sliced S-boxes*, *Bit-based Permutation Layers*, *Reflection Ciphers*, *ARX-based* and for Feistel Networks: *ARX-based*, *Generalized Feistel Networks*, *Two-branched Feistel Networks*.

3.2.1 Substitution-Permutation Networks

Substitution-Permutation Networks provide confusion and diffusion in two distinct operations. Confusion is achieved by a layer of normally small concatenated S-boxes, while diffusion is provided by a linear permutation of the whole blocksize.

AES-*like Ciphers*

AES-like ciphers closely follow the structure of AES, as all ciphers in that category incorporate their internal state as 4×4 matrices, with one entry arranged as a 4-bit *nibble* or a 8-bit *byte* entry. An S-box is applied to each of the entries to provide confusion, and the diffusion layer normally consists of an MDS-matrix applied to the columns of the state matrix.

midori [37] is a family of energy efficient lightweight block ciphers with 16 rounds. The family offers two blocksizes of 64/128 bits and a key size of 128 bits. Midori-64 uses lightweight 4-bit S-boxes to provide confusion and an involutory bi-

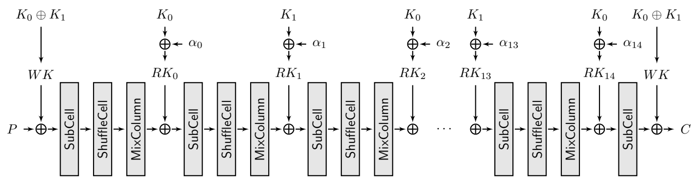


Figure 32: Midori encryption function

nary almost-MDS matrix for diffusion. The round function of Midori is illustrated in Figure 32.

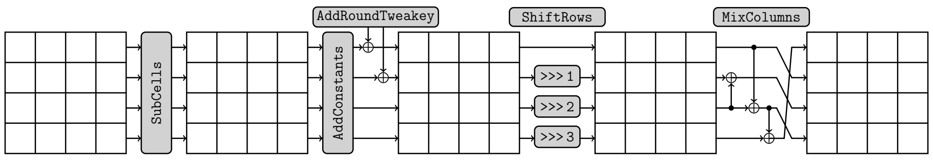


Figure 33: Skinny round function

SKINNY [50] is a family of lightweight tweakable block ciphers based on the Tweakey framework [188]. The family offers block sizes n of 64 and 128 bits and tweakey sizes from $n/2n/3n$ bits. Skinny-64 has an AES-like round function and uses a simple 4-bit S-box for confusion. Diffusion is provided by a very sparse binary matrix, that can be implemented with just three XOR gates. Skinny recommends 32/36/40 rounds depending on the tweakey size. The round function of Skinny is illustrated in Figure 33. We study the security of Skinny in Chapter 5 against zero-correlation attacks and in Chapter 6 against impossible differential attacks.

Bit-sliced S-boxes

Ciphers in this category are based on a non-linear layer that consists of parallel applications of several small S-boxes which can be implemented using basic bitwise operations such as XOR, AND and OR gates.

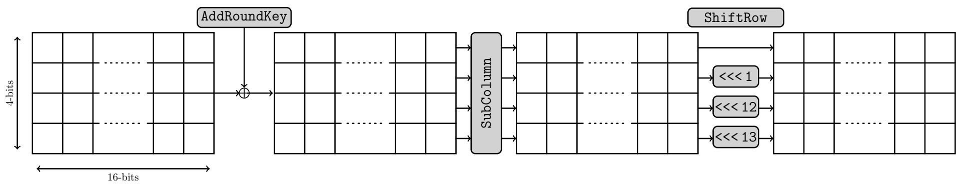


Figure 34: Rectangle round function

RECTANGLE [361] is an ultra lightweight block cipher that is optimised for software and hardware. It operates on a 64-bit state that is arranged in a 4×16 matrix and has 25 rounds. The non-linear layer consists of 4-bit S-boxes that are executed in parallel on the columns to provide confusion. The diffusion layer consists of simple fixed row-wise rotations by a different value for each row. The round function is illustrated in 34.

Bit-based Permutation Layer

Bit-based permutation layer SPNs ciphers can be separated by their diffusion layer. All of them have a simple 4/8—bit S-box layer, and a bit-permutation as diffusion layer. The advantage of those designs is that the bit-permutation can be efficiently implemented in hardware by just wiring the outputs. The diffusion of those designs depends highly on the differential/linear branch numbers of the S-boxes.

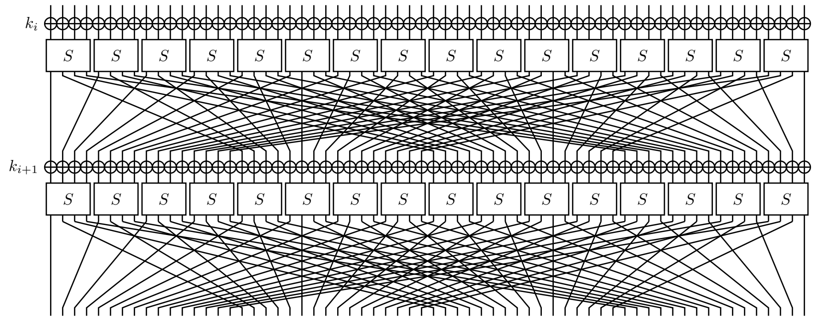


Figure 35: Two round functions of PRESENT

PRESENT [95] is a very lightweight cipher that is optimised for small area. It uses a simple bit-permutation, that can be implemented in hardware just by wiring, to provide diffusion. PRESENT has a block size of 64 bits and supports key sizes of 80 and 128 bits. It uses simple 4-bit S-boxes to provide confusion and a bit-permutation for diffusion. PRESENT recommends 31 rounds where two rounds are illustrated in Figure 51.

Reflection Ciphers

Reflection ciphers have the α reflection property, which means that decryption with one key corresponds to encryption with a related key. This can be achieved by keeping the cipher symmetric around the middle and varying the round constants by a constant α.

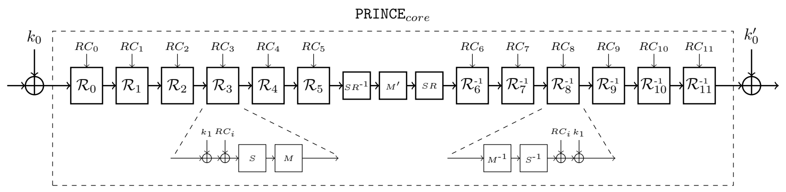

$\mathtt{PRINCE}_{core}$
k_0
k_0'
RC_0 RC_1 RC_2 RC_3 RC_4 RC_5 RC_6 RC_7 RC_8 RC_9 RC_{10} RC_{11}
$\mathcal{R}_0$ $\mathcal{R}_1$ $\mathcal{R}_2$ $\mathcal{R}_3$ $\mathcal{R}_4$ $\mathcal{R}_5$
SR^{-1} M' SR
$\mathcal{R}_6^{-1}$ $\mathcal{R}_7^{-1}$ $\mathcal{R}_8^{-1}$ $\mathcal{R}_9^{-1}$ $\mathcal{R}_{10}^{-1}$ $\mathcal{R}_{11}^{-1}$
k_1 RC_i S M
M^{-1} S^{-1} RC_i k_1

Figure 36: PRINCE round function

PRINCE [101] is a lightweight block cipher that is optimised for latency when implemented in hardware. PRINCE has a 64-bit block size and a 128-bit key size. It uses a 4-bit S-box for confusion and a combination of AES `ShiftRows` and an involutory binary matrix for diffusion. PRINCE has 12 rounds as illustrated in Figure 53.

ARX-based Ciphers

ARX-based SPNs use an ARX-Box to provide confusion instead of an S-box. The use of an ARX-Box allows to provide provable bounds for linear and differential cryptanalysis, which is not possible in other ARX-based designs. Moreover, ARX-based ciphers are very efficient in software and hardware as they just use modular additions, rotations and `XOR` gates.

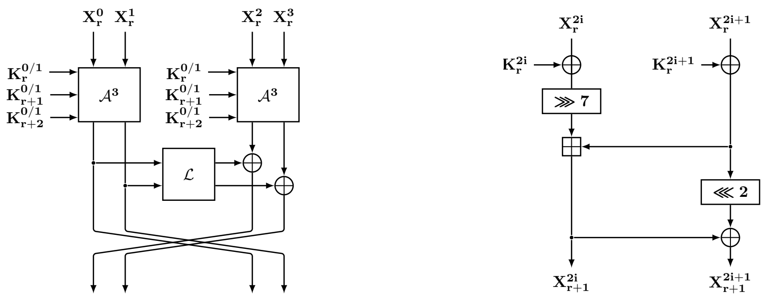


Figure 37: **(Left):** SPARX step function, **(Right):** SPECKEY which is inside of one ARX-box

SPARX [135] is a lightweight block cipher that was designed according to the *long trail strategy* [135]. In contrast to the Wide-Trail strategy that uses small strong S-boxes, in the long-trail strategy a large and weaker ARX-box is used. This allows to bound the differential and linear probabilities, compared to other ARX designs. SPARX-64 uses a 32-bit block cipher that is similar to SPECK with 32-bit block size as an ARX-based S-box to provide confusion. Diffusion is provided by a linear layer that is similar to the linear layer of NOEKEON [190] in a Feistel structure. We further analyse the security of SPARX in Chapter 7 against truncated-differential and rectangle attacks.

3.2.2 Feistel Networks

In a Feistel construction, the input is split in two parts, that interact with each other. The advantage of the structure of an Feistel network is that encryption and decryption operations are very similar or even identical in some ciphers.

ARX-based Feistel Networks

ARX-based Feistel networks use only modular additions, rotations and `XOR` in their round functions. The non-linearity is provided by the modular additions, while diffusion is provided by the `XOR` and rotations.

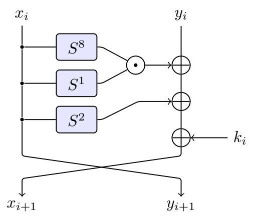


Figure 38: Round function of SIMON

SIMON [46] is a family of AND-RX block ciphers optimized for hardware. The cipher was designed by the NSA and performs exceptionally well in hardware and software. It is a classical Feistel-network which uses an AND gate to provide non-linearity for confusion and rotations and XOR for diffusion. The SIMON family offers block sizes of 32 to 128 bits and key sizes of 64 or 256 bits. Depending on the block size and key size SIMON recommends between 32 and 72 rounds. The round function of SIMON is illustrated in Figure 38.

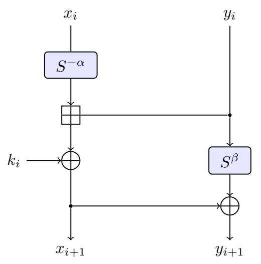


Figure 39: Round function of SPECK

SPECK [46] is a family of ADD-RX bock ciphers optimized for software. The cipher was designed by the NSA and follows a typical ARX cipher where the round function is inspired by the underlying block cipher THREEFISH of the hash function SKEIN [156]. The SPECK family offers block sizes of 32 to 128 bits and key sizes of 64 or 256 bits. Depending on the block size and the key size SPECK recommends between 22 and 34 rounds. The round function of SPECK is illustrated in Figure 39.

Generalized Feistel Networks

Compared to classical Feistel networks such as DES, generalized Feistel networks split the state in two or more words, and use the permutation of the Feistel network for diffusion.

TWINE [318] is a lightweight block cipher based on a generalized Feistel structure. The cipher has 16 branches of 4-bit each. The Feistel function, that is iterated 8 times per round, consists of XORing the round key and a 4-bit S-box to provide confusion. The diffusion layer is a sophisticated permutation. TWINE has a 64-bit block size and a key size of 80 or 128 bits. The round function of TWINE is illustrated in Figure 40.

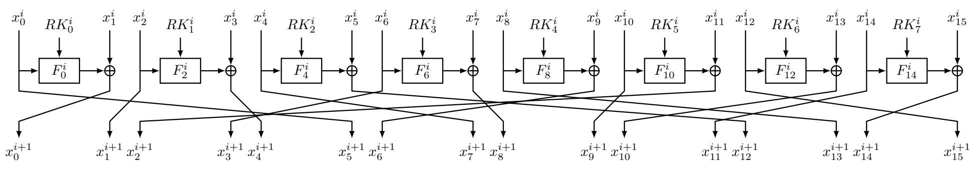

x_0^i RK_0^i x_1^i x_2^i RK_1^i x_3^i x_4^i RK_2^i x_5^i x_6^i RK_3^i x_7^i x_8^i RK_4^i x_9^i x_{10}^i RK_5^i x_{11}^i x_{12}^i RK_6^i x_{13}^i x_{14}^i RK_7^i x_{15}^i
F_0^i F_2^i F_4^i F_6^i F_8^i F_{10}^i F_{12}^i F_{14}^i
x_0^{i+1} x_1^{i+1} x_2^{i+1} x_3^{i+1} x_4^{i+1} x_5^{i+1} x_6^{i+1} x_7^{i+1} x_8^{i+1} x_9^{i+1} x_{10}^{i+1} x_{11}^{i+1} x_{12}^{i+1} x_{13}^{i+1} x_{14}^{i+1} x_{15}^{i+1}

Figure 40: TWINE round function

Two-branched Feistel Networks

Two-branched Feistel networks split the state in two sub-words. The round function is then applied to one of the words and the output is XORed to the other sub-word.

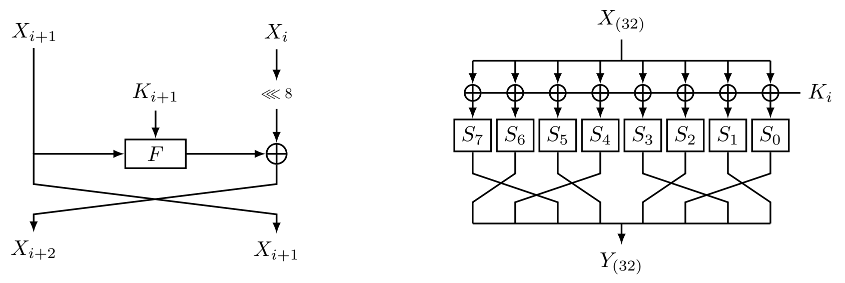


Figure 41: (Left) Round function of LBLOCK, (Right) Feistel function of LBLOCK

LBLOCK [318] is a lightweight block cipher optimised for both hardware and software. It is based on a two-branched Feistel network of 32 bits each. The Feistel function of LBLOCK consists of a layer of 8 distinct 4-bit S-boxes and a word permutation that is similar to TWINE. LBLOCK has a block size of 64 bits and a key length of 80 bits. It consists of 32 Feistel rounds as illustrated in Figure 41.

3.3 FINDING DIFFERENTIAL TRAILS EFFICIENTLY

While there are many methods based on SAT, MILP or Matsui's algorithm to find differential trails and even prove an upper bound on the probability of the best single trail, it remains a hard problem to find a good estimate on the probability of the best differential. Even finding those differential trails remains a difficult problem for some design strategies and cryptanalysts had to search manually for differentials in some attacks [345]. Nowadays a variety of automated tools [84, 222, 313] is available which are constantly improved and help cryptanalysts in finding good differential trails.

3.3.1 SAT/SMT Solvers

SAT solvers are used to solve the Boolean satisfiability problem (SAT) and are based on heuristic algorithms. A solver starts from an initial assignment for the literals and then builds a search tree by using systematic backtracking until all conflicting clauses are resolved and either an assignment of variables for a satisfiable set of clauses is returned or the solver decides that this instance is unsatisfiable. The most commonly algorithms used in SAT solvers are based on the original idea of DPLL [127].

SMT solvers are more powerful than SAT solvers in the sense that they can express constraints on a higher abstraction layer and allow simple first-order logic. In general, SMT solvers often translate the problem to SAT and then use an improved version of the DPLL algorithm and backtracking to infer when theory conflicts arise. Moreover, the solver checks the feasibility of conjunctions from the first-order logic predicates as it interacts with the Boolean formulas that are returned by the SAT solver.

There exists a few SAT/SMT solvers that are suitable for our use cases. STP [336] is an SMT solver that uses the CVC and SMTLIB2 language to encode the constraints and then invokes a SAT solver to check for satisfiability of the model. CryptoMiniSat [236] is an advanced SAT solver that supports features like `XOR` recovery[1] to simplify clauses. As `XOR` operations are commonly used in cryptography this can be an advantage and potentially reduces the solving time. We also considered other solvers like Boolector [264], which for some instances provide a better performance, however in general this only provides an improvement by a small constant factor and it is hard to identify for which instances one obtains any advantage.

3.3.2 From Differential Cryptanalysis to Satisfiability Modulo Theories

When using automated tool like SAT/SMT solvers, one can simplify the search for differential trails and differentials by modelling the differential behaviour of the block cipher. For this we represent all intermediate states of our block cipher as variables which corresponds to the differences and encode the transitions of differences through the round functions as constraints that can be processed by the SMT/SAT solver. An advantage of using SMT over SAT for the modelling is that most SMT solvers support reasoning over bit-vectors which are commonly used in block cipher designs, especially when considering word-oriented ciphers. This both simplifies the modelling of the constraints and can lead to an improved time for solving the given problem instances compared to an encoding in SAT.

Constructing an SMT Model

We focus on a tool that uses the CVC language[2] for encoding the differential behaviour of block ciphers. Therefore, we encode the constraints imposed by the round function for each round of the block cipher and the probability of the resulting differential transitions. Our main goal here is to construct an SMT model which decides whether

$$\exists Q : \mathrm{DP}(Q) = 2^{-t}, \tag{45}$$

[1] See `https://www.msoos.org/2011/03/recovering-xors-from-a-cnf/`

[2] A list of all bitwise and word level functions in CVC is available at: `http://stp.github.io/cvc-input-language/`

which allows us to find the best differential trail Q for a cipher by finding the minimum value t for which the model is satisfiable, where t is a non-negative integer.
In order to represent the differential behaviour of a cipher we consider any operation in the cipher, *e.g.*, the application of an S-box, matrix multiplication, word-wise operation or bit operation, and add constraints for a valid transition from an input to an output difference such that any valid assignment to the variables corresponds to a valid differential trail in the actual operation. For any non-linear component we introduce additional variables w^j which represent the $\log_2$ probability of the differential transition. The probability of Q is then given by $\sum w^j$. This means that a valid assignment for all these variables directly gives us the differential trail Q with all intermediate differences and $DP(Q) = p$.
In the following we give an overview on how the different components of the ciphers can be modelled in the SMT model. The algorithms to find the optimal differential trails and consequently good estimates for the differentials are described in the following.

S-boxes

Substitution Permutation Network (SPN) ciphers typically use S-boxes, which are non-linear functions operating on a small number of bits. These are often 4- or 8-bit functions and therefore we can compute the differential probability by simply constructing the Difference Distribution Table (DDT), which is a full lookup table of all possible pairs of input/output differences, for each S-box. In our SMT model we represent the input difference to an n-bit S-box as $\alpha = \alpha_1, \dots, \alpha_n$ respectively the output as $\beta = \beta_1, \dots, \beta_n$. These variables correspond to the input/output difference to this S-box and we want to constrain them to only allow non-zero probability combinations of input and output differences. We further introduce additional variables $w = w_1, \dots, w_n$ which are used to represent the probability of the transition. The probability of the transition is encoded as $2^{-wt(w)}$, where $wt(\cdot)$ denotes the Hamming weight of w.
In order to construct the constraints on the variables, we first find all valid transitions and their corresponding probability. We want to construct a CNF which is satisfiable if and only if the assignment corresponds to such a valid trail. One simple way to this is by just considering all assignments which are impossible. If a transition is defined as $(a \xrightarrow{S} b)$ and has a probability c then we add the following clause

$$\begin{aligned} T = {} & N(a_1, \alpha_1) \vee \dots \vee N(a_n, \alpha_n) \vee \\ & N(b_1, \beta_1) \vee \dots \vee N(b_1, \beta_n) \vee \\ & N(c_1, w_1) \vee \dots \vee N(c_n, w_n) \end{aligned} \tag{46}$$

where

$$N(x_i, y_i) = \begin{cases} \neg y_i, \text{if } x_i = 0 \\ \ \ y_i, \text{if } x_i = 1 \end{cases} . \tag{47}$$

This clause is only satisfiable if the variables of the corresponding S-box are not set to the invalid assignment. For example let $a = (1,0,1,1)$, $b = (0,0,0,0)$ and $c = (0,0,0,0)$ then we add the clause

$$(\neg\alpha_0 \vee \alpha_1 \vee \neg\alpha_2 \vee \neg\alpha_3 \vee \beta_0 \vee \beta_1 \vee \beta_2 \vee \beta_3 \vee w_0 \vee w_1 \vee w_2 \vee w_3). \qquad (48)$$

We implemented this approach to generate the SMT models for 4- and 8-bit S-boxes, where most of the lightweight ciphers actually use 4-bit S-boxes which allows a very compact description (*i.e.,* to represent the 4-bit S-box of Skinny we need 12 variables and about 3999 clauses in CNF). Note that our method is limited to S-boxes which have a DDT with entries that are a power of 2. For other S-boxes a similar method could be used by using l additional variables for encoding probabilities of the form $2^{-0.5}, 2^{-0.25}, \ldots$ to get an approximation of the actual probability.

Linear Layers

The diffusion layers of Substitution Permutation Networks in lightweight ciphers are often constructed with simple bit-permutations (*e.g.,* Present) or by multiplication with matrices having only binary coefficients (*e.g.,* Midori, Skinny). ARX-based ciphers (*e.g.,* Speck) use the diffusion properties of XOR combined with rotations. Feistel networks (*e.g.,* Simon, LBlock, Twine) also mix the state by switching parts of the states on every Feistel switch.
For modelling rotations and bit-permutations in an SAT/SMT solver, we simply have to re-index the variables accordingly before they are input to another function. This can be achieved using SMT predicates (ASSERT and equality) in the CVC language. Rotations can be realised using predicates for *shifting* words and the word-wise *or* function that are available in the CVC language. The multiplication by a binary matrix can be modelled using the *xor* predicate at the word-level.

ARX Designs

ARX designs use modular additions (modulo 2^n), XOR and rotations. As modular addition is the only non-linear component, that is not already available in the SMT solver, we use an algorithm proposed by Lipmaa and Moriai [226] to efficiently compute the differential probability of modular addition. Let $\mathrm{xdp}^+(\alpha, \beta \to \gamma)$ be the XOR differential probability of modular addition, where α, β are input differences and γ is the output difference, then it holds that a differential is valid if and only if:

$$\mathrm{eq}(\alpha \lll 1, \beta \lll 1, \gamma \lll 1) \wedge (\alpha \oplus \beta \oplus \gamma \oplus (\beta \lll 1)) = 0 \qquad (49)$$

where

$$\mathrm{eq}(x, y, z) := (\neg x \oplus y) \wedge (\neg x \oplus z). \qquad (50)$$

The weight of a valid differential is defined as:

$$w(\alpha, \beta, \gamma) := -\log_2(\mathrm{xdp}^+(\alpha, \beta \to \gamma)) = wt'(\neg\mathrm{eq}(\alpha, \beta, \gamma)). \qquad (51)$$

where $wt'(\cdot)$ denotes the Hamming weight omitting the most significant bit. We implemented this algorithm to calculate the differential probability of modular additions.

3.3.3 Finding the Best Trail and Differentials

We use the open-source tool CryptoSMT [313] for the automated search of differential trails and implemented several missing functionalities for block ciphers (*i.e.,* support for S-boxes as described in Section 4.3, and binary diffusion matrices). CryptoSMT is based on the state-of-the-art SAT/SMT solvers, CryptoMiniSat [236] and STP [336].
The tool offers a simple API that allows cryptanalysts and designers to formulate various cryptanalytic problems and solve them with the underlying SAT/SMT solver. We added the models for the block ciphers Skinny, Midori, Rectangle, Present, Prince, Sparx, Twine and LBlock (Note that some of these are block cipher families and we focused on a subset of parameters) to CryptoSMT and use the following two functionalities provided by the tool:

- Decide if a differential trail with probability p exists.
- Enumerate all differential trails with a probability of p.

Based on this we can achieve our two goals, namely finding the best differential trail and estimating the probability of the differential.

Best Differential Trail

In order to find the trail Q with maximum probability p_{max} for r rounds of a block cipher we start by checking whether our model is satisfiable for a probability of p, starting at $p = 1$. If our model is not satisfiable we continue by checking whether there is a valid assignment for $p = 2^{-1}$. Note that for all our block ciphers the probability of the differential transitions are powers of two and therefore there does not exist any differential trail which has a probability p' such that $2^{-(t+1)} < p' < 2^{-t}$ for any integer t. We continue this process until we reach a model which is satisfiable, which gives us an assignment of all variables of the state forming a valid differential trail with probability $p_{\text{max}} = 2^{-t}$. Considering that we start with probability $p = 1$ and then we constantly increase the weight, and finish as soon as we found an valid assignment, we can ensure that we found the best differential trail. Our experiments have shown that this search strategy is considerable fast. As it is normally easier for an SAT solver to detect if a problem is satisfiable, then to detect if it is unsatisfiable (*i.e.,* which in the worst case would require to exhaustively go through the whole search space), another good search strategy would be a top-down approach, where one could search for a valid trail for a large t, and then decrease t with a constant that becomes smaller for each step. This algorithm is also known as *gradient descent* and is often used to find a maximum/minimum in optimisation problems.

Estimate for the Probability of a Differential

In order to find a good differential we can use a tool assisted approach to compute an approximation for Equation 42, as shown in [256]. We first obtain the best single trail Q with probability $p = 2^{-t}$ which gives us the input difference α_1 and output difference α_r. Subsequently we modify our model and fix the input and output

difference to α_1 respectively α_r. Note that this restricts the search space significantly and results in a much faster time for solving any subsequent SMT instances.
The next step is to find all differential trails Q, such that $DP(Q) = 2^{-u}$, for $u = t, t+1, \ldots$, under this new constraints. This allows us to collect more and more terms of the sum in Equation 42, improving our approximation for the differential. By doing this process we always search for those differential trails which contribute the most to the probability of the differential first.
Here we assume that the input and output difference imposed by the best differential trail correspond to a good differential. While this assumption might not always hold and some of the differentials we found significantly improve the best differential distinguishers there could still exist better starting points for our search, for example as shown in [207] against the block cipher Simeck.

3.4 ANALYSIS OF THE GAP IN LIGHTWEIGHT CIPHERS

The construction of cryptographic primitives optimised for resource constrained devices has received a lot of attention over the last decade and various design strategies and optimisation targets have been explored. All these primitives exhibit the idea of using simpler operations in order to save costs and therefore often exhibit a simpler algebraic structure compared to other symmetric-key algorithms.
For some design strategies this leads to a significant larger gap between single trails and differentials. This gap becomes especially relevant for aggressively optimised designs with minor security margins. Table 6 gives an overview of all the block ciphers we analysed with the methodology outlined in Section 3.3 and their security margins as well as the best known differential attacks.
In the following, we provide some detailed analysis of the differential gaps of several lightweight block ciphers. We focus in more detail on Skinny, Midori and Sparx as there are no results published on the differential effects of those ciphers to the best of our knowledge.

3.4.1 Skinny

Skinny [50] is an AES-like tweakable block cipher, based on the Tweakey framework [188]. The aim of Skinny is to achieve the hardware performance of the AND-RX-cipher Simon and have strong security bounds against differential/linear attacks (this includes the related-key scenario), while also having competitive software performance. The resistance against differential/linear attacks in Skinny is based on counting the minimal number of active S-boxes, in the single-key and related-tweakey models. As the design of Skinny is based on a few very simple but highly efficient cryptographic building blocks, it seems intuitive that one can expect that a large number of differential trails will contribute to a differential. Recent attacks [20, 230] exploited the low branch number of the binary diffusion matrix, as well as properties of the tweakey schedule.
Using our tool-assisted approach we analysed this gap in Skinny-64 (see Figure 42) and can provide some new insights to the security of Skinny-64. For example the

Table 6: Best attacks and security margins (active S-boxes) for various design strategies for symmetric cryptographic primitives. D/MD/RK/ID/R/TD = differential, multiple differential, related-key, impossible differential, rectangle, truncated differential

Group	Design Strategy	Cipher	Block Size	Key Size	Rounds	Margin (active S-boxes)	Best Differential Attack	Exploit Differentials
SPN	AES-like	MIDORI	64	128	16	9 rounds	full rounds (RK) [162]	✗
		SKINNY	64	64	32	24 rounds	19 rounds (ID) [230]	✓
		SKINNY	64	128	36	28 rounds	23 rounds (ID) [20, 230]	✓
		SKINNY	64	192	40	32 rounds	27 rounds (R) [230]	✓
	Bit-sliced	RECTANGLE	64	80/128	25	-	18 rounds (D) [320, 361]	Sec.3.4.6
	PRESENT-like	PRESENT	64	80/128	31	12 rounds	26 rounds (D) [229, 343]	✓
	Reflection	PRINCE	64	128	12	-	10 rounds (MD) [109]	✓
	ARX	SPARX	64	128	24	9 rounds	16 rounds (TD) [26]	✓
Feistel	AND-RX	SIMON	64	96	42	-	26 rounds (D) [7]	✓
		SIMON	64	128	44	-	26 rounds (D) [7]	✓
	ARX	SPECK	64	96	26	-	19 rounds (D) [312]	✓
		SPECK	64	128	27	-	20 rounds (D) [312]	✓
	GFN	TWINE	64	80	36	21 rounds	23 rounds (ID) [75]	✗
		TWINE	64	128	36	21 rounds	25 rounds (ID) [75]	✗
	Two-branched	LBLOCK	64	80	32	17 rounds	24 rounds (ID) [344]	✗

best 8-round single differential trail $Q^8_{\max}$ suggests a probability of 2^{-72} while the differential D^8 defined by the input/output difference of $Q^8_{\max}$ consists of a large cluster of trails leading to the differential

$$\texttt{0x0104401000C01C00} \xrightarrow{8\text{—round SKINNY-64}} \texttt{0x0606060000060666} \qquad (52)$$

with a probability of larger than $2^{-56.55}$ by taking all 821896 trails[3] into account which have $\mathrm{DP} > 2^{-99}$. Note that the probabilities and the number of characteristics are obtained with a fixed input/output difference as noted in Equation 52. This suggests that estimates from active S-boxes should be taken with care as the gap is fairly large. However, the number of rounds in SKINNY-64 is chosen very conservatively and it provides a large security margin.

In particular the probability of the differential improves very quickly when adding more trails, as the distribution of the number of trails with a probability 2^{-t} is very flat over the choice of t (see Figure 42). For example there are 39699 trails with $\mathrm{DP} = 2^{-75}$ and 25413 trails with $\mathrm{DP} = 2^{-76}$ and the probability of the differential only improves marginally by considering more trails with a lower probability. On the contrary, for designs like SIMON (see Figure 46) this distribution grows exponentially as the probability of the single trails decreases as has also been noted in [206], and one has to take a much larger number of trails into account before getting a

[3] This process took in total 23.5 hours on a single core, however after 1 hour the estimate for the differential probability improves by less than $2^{-0.9}$.

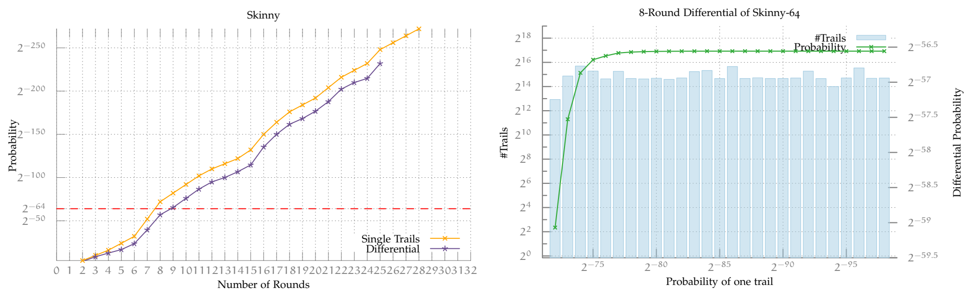


Figure 42: Probability for the best single trails and differentials for Skinny-64 (left), and the distribution of the number of trails with a fixed probability contributing to the best 8-round differential for Skinny-64 (right). The green line indicates the probability of the differential when summing up the probability of all trails up to this probability, which highlights the small improvement when adding all lower probability trails.

good approximation. For a detailed overview over how many trails contribute to each differential see Table 7.

3.4.2 Midori

Midori is an AES-like lightweight block cipher optimised for low-energy usage using a binary near-MDS matrix combined with a generic cell permutation for diffusion. Despite that Midori-64 has a large number of 2^{32} weak keys, for which Midori-64 can be practically broken with invariant subspace attacks [166], there has been no differential attacks on even reduced versions of Midori, apart from a related-key attack by Gérault and Lafourcade [162].
The gap between the differential probability of a single trail and a differential behaves similar to Skinny-64, *i.e.*, counting the active S-boxes gives an inaccurate bound against differential distinguishers. For example we found new differentials for Midori-64 where the 8-round single differential trail suggests a probability of 2^{-76} and the corresponding 8-round differential

$$\texttt{0x0A000000A0000005} \xrightarrow{8-\text{round Midori-64}} \texttt{0x000000000000A0AA} \tag{53}$$

has a probability of larger than $2^{-60.86}$ by summing all 693730 trails up to a probability of 2^{-114}. Similar to Skinny the distribution of the contributing trails is very flat, which means that we quickly approach a good estimate for the probability of the differential (see Figure 43).

3.4.3 Sparx

Sparx [135] is based on the *long-trail strategy*, introduced alongside with Sparx, which can be seen as combining the ARX approach with an SPN, allowing to provide bounds on the differential resistance of an ARX cipher by counting the active S-boxes. While it is also feasible to prove such a bound using the methodology

Table 7: Detailed results on the differentials found for Skinny-64. We first provide the best differential trail for r rounds. Next we give an estimate of the differential with the input/output difference of the best differential trail found. #Trails gives the number of differential trails we used for the estimate followed by the maximum *weight* of the differential trails we use for the estimate. $\text{Time}_{\text{trail}}$ gives the search time to find the best single differential trail and $\text{Time}_{\text{diff}}$ provides the time to search all the differential trails for the best differential.

r	$\Pr_{\text{Trail}}$	$\Pr_{\text{Differential}}$	#Trails	Max weight	$\text{Time}_{\text{trail}}$	$\text{Time}_{\text{diff}}$
6	2^{-32}	$2^{-23.51}$	100319	45	22m54s	1h38m
7	2^{-52}	$2^{-39.49}$	141800	58	1h03m	5h13m
8	2^{-72}	$2^{-56.55}$	821896	98	1h24m	23h20m
9	2^{-82}	$2^{-65.36}$	277464	89	1h06m	29h25m
10	2^{-92}	$2^{-75.98}$	66438	92	1h42m	2h59m
11	2^{-102}	$2^{-86.63}$	64339	103	2h36m	3h14m
12	2^{-110}	$2^{-95.00}$	62382	113	3h12m	3h37m
13	2^{-116}	$2^{-100.06}$	165079	124	2h42m	24h42m
14	2^{-122}	$2^{-106.71}$	100457	127	3h30m	10h25m
15	2^{-132}	$2^{-114.65}$	326404	142	7h23m	37h21m
16	2^{-150}	$2^{-135.41}$	24598	150	30h35m	1h44m
17	2^{-164}	$2^{-150.07}$	21524	165	60h09m	1h53m
18	2^{-176}	$2^{-161.64}$	20903	177	92h04m	1h54m
19	2^{-184}	$2^{-168.27}$	54245	185	60h22m	3h38m
20	2^{-192}	$2^{-176.74}$	39169	193	60h10m	2h59m
...						

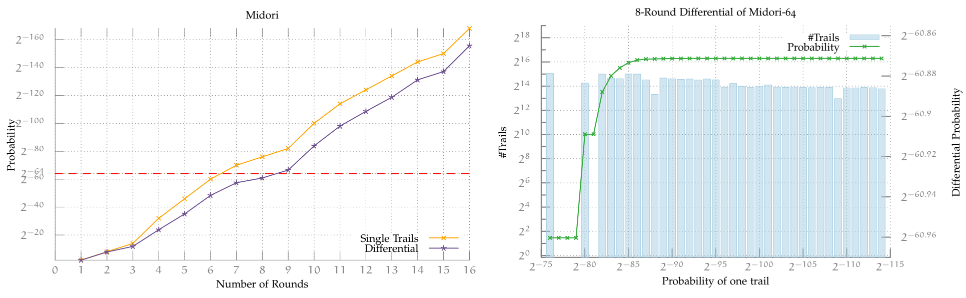


Figure 43: Probability for the best single trails and differentials for various rounds of Midori-64 (left), and Distribution of the trails contributing to the best 8-round Differential for Midori-64 (right).

from Section 3.3, it is often computationally infeasible or the bounds are not very tight [256]. The designers of Sparx used the YAARX toolkit [84] to show truncated

Table 8: Detailed results on the differentials found for MIDORI-64.

r	Pr_{Trail}	$Pr_{Differential}$	#Trails	Max weight	$Time_{trail}$	$Time_{diff}$
4	2^{-32}	$2^{-23.79}$	896	36	31m36s	2m4s
5	2^{-46}	$2^{-35.13}$	55168	54	56m42s	1h10m
6	2^{-60}	$2^{-48.36}$	11072	71	1h54m	29m
7	2^{-70}	$2^{-57.43}$	28588	99	3h12m	1h32m
8	2^{-76}	$2^{-60.87}$	693730	114	1h6m	23h36m
9	2^{-82}	$2^{-66.52}$	104694	90	56m	3h12m
10	2^{-100}	$2^{-83.86}$	120181	106	5h12m	4h36m
11	2^{-114}	$2^{-98.04}$	87055	119	10h56m	3h18m
12	2^{-124}	$2^{-108.59}$	88373	131	1d02h	4h54m
13	2^{-134}	$2^{-118.70}$	56596	139	22h02m	3h06m
14	2^{-144}	$2^{-131.18}$	13932	149	1d16h	9h36m
15	2^{-150}	$2^{-137.07}$	25680	155	20h30m	1h48m
16	2^{-168}	$2^{-155.58}$	11815	172	3d21h	1h12m

trails, that they used to compute the differential bounds. One of the main design motivations of SPARX was that it should be very difficult to find differential trails for a large number of rounds for ARX-based ciphers with a state of more than 32 bits [134].
Our experiments have shown that ARX ciphers do not have a very strong differential effect compared to the previous lightweight SPN constructions. However, as SPARX is in-between those it is an interesting target. Our results suggest that SPARX-64 has a differential effect comparable to other ARX designs like SPECK-64 (see Figure 44). The major limitation for applying our approach to SPARX is that the search for optimal differential trails on SPARX is computationally very costly. While trails up to 6 rounds can be found in less then 5 minutes, the 10-round trail took already 32 days, on a single core[4].

3.4.4 Results for other Lightweight Ciphers

Table 13 summarises the gaps between single-trails and differentials for all lightweight block ciphers we analysed. We observed that for most ciphers a large gap between the probability for single-trails and differentials exists and that a higher number of rounds is required for the block ciphers to be *differential resistant*. The gaps also increase significantly with the number of rounds, which is not surprising as with more rounds there are more valid differential trails for a given input/output difference.

[4] Note that this process can not easily be parallelised as most SAT solvers are inherently serial.

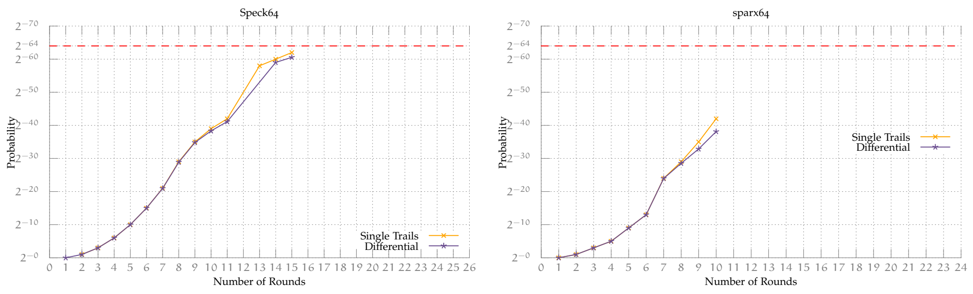


Figure 44: Comparison of the best single trails and differentials for various rounds of SPECK-64 (left), and SPARX-64 (right).

Table 9: Detailed results on the differentials found for SPARX-64.

r	Best Trail	Differential	#Trails	Max weight	$\text{Time}_{\text{trail}}$	$\text{Time}_{\text{diff}}$
1	1	1	1	1	0.02s	0.03s
2	2^{-1}	2^{-1}	1	2	0.1s	0.07s
3	2^{-3}	2^{-3}	1	4	0.5s	0.09s
4	2^{-5}	$2^{-4.99}$	8	49	2.4s	3.36s
5	2^{-9}	$2^{-8.99}$	12944	58	25s	2m12s
6	2^{-13}	$2^{-12.99}$	70133	51	3m48s	3h06m
7	2^{-24}	$2^{-23.95}$	56301	60	47h48m	28m
8	2^{-29}	$2^{-28.53}$	37124	60	15d5h	17m
9	2^{-35}	$2^{-32.87}$	233155	58	22d7h	7h42m
10	2^{-42}	$2^{-38.12}$	1294158	73	32d12h	35h18m
....						

The biggest gap, in term of number of rounds, occurs for SIMON-64 with a gap of five rounds. There is also a 2-round gap for ciphers like PRESENT, MIDORI and TWINE. However, it seems that the gap for SIMON-64 grows faster, considering that the differentials and trails seem to follow an exponential growth as also observed in [206]. In comparison PRESENT, MIDORI and TWINE seem to grow in a linear way. In relation to the number of rounds, the gap for MIDORI also has quite a significant impact and allows to extend the distinguisher by two rounds. Further we observed that there seem to be nearly no gaps for ciphers like RECTANGLE and SPECK. We illustrate the gaps for the analysed ciphers in Figure 45 and we provide Figure 46 showing the distribution of valid differential trails that contribute to the probability of the best differential for each cipher.

3.4.5 Differentials for Midori, Skinny and Sparx

In the following we give the best differentials that we found for Midori, Skinny and Sparx. The differentials for many other lightweight ciphers together with the source code to generate the differential models is publicly available at: `https://github.com/TheBananaMan/cryptosmt`

Table 10: The best differentials that we found for various rounds of Midori-64.

r	Differential	$\Pr_{\text{Differential}}$
4	`0x0000020000022000` $\rightarrow$ `0x0020220002022002`	$2^{-23.79}$
5	`0x0004100000000100` $\rightarrow$ `0x0222220222222022`	$2^{-35.13}$
6	`0x0550000000005000` $\rightarrow$ `0x0000AA0000007707`	$2^{-48.36}$
7	`0x0AA00500700A0000` $\rightarrow$ `0x00005AFF0000AAA0`	$2^{-57.43}$
8	`0x0A000000A0000005` $\rightarrow$ `0x000000000000A0AA`	$2^{-60.87}$
9	`0x0000000A050000A0` $\rightarrow$ `0x770700000AAAA0AA`	$2^{-66.52}$
10	`0x0500005050000000` $\rightarrow$ `0xDD7A7D0D25727A7D`	$2^{-83.86}$
11	`0x0000A00000500500` $\rightarrow$ `0xAAA0AAA50AAAAA0A`	$2^{-98.04}$
12	`0xA0A00A0A00007000` $\rightarrow$ `0x0000DD7A00007077`	$2^{-108.59}$
13	`0x0000A0070A000AA0` $\rightarrow$ `0x00000555A5AFAF5F`	$2^{-118.70}$
14	`0x0000000000000500` $\rightarrow$ `0x000070777707AAA0`	$2^{-131.18}$
15	`0x0A0000A00000000A` $\rightarrow$ `0x05550000AA0AAAA0`	$2^{-137.07}$
16	`0xAA00A0A0AAA00A70` $\rightarrow$ `0x00007077AA0A7770`	$2^{-155.58}$

3.4.6 Application of the Differential Gaps to the Best Published Differential Attacks

In the following, we analyse the best published differential attacks and discuss improvements of the attacks by considering the differential gaps between single-trails and differentials.

midori-64. Gérault and Lafourcade [162] proposed related-key differential attacks on full-round Midori-64, where they use $16 \cdot 15$-round and $4 \cdot 14$-round related-key differential trails to recover the key. In their attacks they do not exploit differentials. In comparison, the best differential that we found reaches 8 rounds with a probability of $2^{-60.86}$.

skinny-64. Liu *et al.* [230] propose related-tweakey rectangle attacks on 26 rounds of Skinny-64-192 and they use optimal single differential trails based on truncated-differential trails. The authors exploit the differential gap of Skinny by using 5000 single differential trails to compute the differential for a 22-round distinguisher. In

Table 11: The best differentials that we found for various rounds of SKINNY-64.

r	Differential	$\Pr_{\text{Differential}}$
6	0x0041C00001000000 $\rightarrow$ 0x4044400400404444	$2^{-23.51}$
7	0x002220222B222000 $\rightarrow$ 0x0444004404004444	$2^{-39.49}$
8	0x0104401000C01C00 $\rightarrow$ 0x0606060000060666	$2^{-56.55}$
9	0x0020000200020200 $\rightarrow$ 0x0060000100600160	$2^{-65.36}$
10	0x0008200020000020 $\rightarrow$ 0x0008808000880088	$2^{-75.98}$
11	0x0002200000000200 $\rightarrow$ 0x0444004404004444	$2^{-86.63}$
12	0x0004000000000000 $\rightarrow$ 0x0001000100000001	$2^{-95.00}$
13	0x0200000000002000 $\rightarrow$ 0x0001001100000001	$2^{-100.06}$
14	0x4000040000400000 $\rightarrow$ 0x0404040000040444	$2^{-106.71}$
15	0x8008080000800000 $\rightarrow$ 0x1066100600601666	$2^{-114.65}$
16	0x0020000220000000 $\rightarrow$ 0x8880088080008888	$2^{-135.41}$
17	0x004C400004000000 $\rightarrow$ 0x2002022022020022	$2^{-150.07}$
18	0x400C0000C00C0000 $\rightarrow$ 0x0077001100660077	$2^{-161.64}$
19	0x2200000000002008 $\rightarrow$ 0x0077001100660077	$2^{-168.27}$
20	0x8800000000008009 $\rightarrow$ 0x8800080900008800	$2^{-176.74}$
...		

Table 12: The best differentials that we found for various rounds of SPARX-64.

r	Differential	$\Pr_{\text{Differential}}$
1	(0x0040,0x8000,0x0000,0x0000) $\rightarrow$ (0x0000,0x0002,0x0000,0x0000)	1
2	(0x0010,0x2000,0x0000,0x0000) $\rightarrow$ (0x8000,0x8002,0x0000,0x0000)	2^{-1}
3	(0x2800,0x0010,0x0000,0x0000) $\rightarrow$ (0x8300,0x8302,0x8100,0x8102)	2^{-3}
4	(0x0000,0x0000,0x2800,0x0010) $\rightarrow$ (0x8000,0x840A,0x0000,0x0000)	$2^{-4.99}$
5	(0x0000,0x0000,0x0211,0x0A04) $\rightarrow$ (0x8000,0x840A,0x0000,0x0000)	$2^{-8.99}$
6	(0x0000,0x0000,0x0211,0x0A04) $\rightarrow$ (0xAF1A,0xBF30,0x850A,0x9520)	$2^{-12.99}$
7	(0x0000,0x0000,0x7448,0xB0F8) $\rightarrow$ (0x8004,0x8C0E,0x8000,0x840A)	$2^{-23.95}$
8	(0x0000,0x0000,0x0050,0x8402) $\rightarrow$ (0x0040,0x0542,0x0040,0x0542)	$2^{-28.53}$
9	(0x2800,0x0010,0x2800,0x0010) $\rightarrow$ (0x5761,0x1764,0x5221,0x1224)	$2^{-32.87}$
10	(0x2800,0x0010,0x2800,0x0010) $\rightarrow$ (0x8081,0x8283,0x8000,0x8002)	$2^{-38.12}$
...		

comparison, the best differential trail with no differences in the tweak/key that we found reaches 8 rounds with a probability of $2^{-56.55}$.

Table 13: Gap between the number of rounds required for a cipher to be *differential trail resistant* (DTR) and *differential resistant* (DR). Note that DR is only a lower bound and there might still exist better differentials.

Group	Design Strategy	Cipher	Block Size	Key Size	Rounds	DTR	DR
SPN	AES-like	Midori	64	128	16	7	9
		Skinny	64	64/128/192	32	8	9
	Bit-sliced	Rectangle	64	80/128	25	15	15
	Present-like	Present	64	80/128	31	15	17
	Reflection	Prince	64	128	12	6	8
	ARX-based	Sparx	64	128	24	15	15[5]
Feistel	AND-RX	Simon	64	96/128	42	19	24[6]
	ARX	Speck	64	96/128	26	> 15	> 15[7]
	GFN	Twine	64	80/128	36	14	16
	Two-branched	LBlock	64	80	32	15	16

RECTANGLE. Zhang *et al.* [361] studied the differential effect and showed an 18-round differential attack, where they used a 14-round differential with a probability of $2^{-62.83}$. In our analysis we found a better differential for 14 rounds with probability of $2^{-60.63}$ by summing up 40627 single-trails which would improve the complexity of these attacks. For more rounds the distinguisher are below 2^{-64}.

PRESENT. Liu and Jin [229] presented an 18-round attack based on slender-sets. Wang *et al.* [343] further presented normal differential attacks on 16-round Present where they used a differential with probability $2^{-62.13}$ by summing up 91 differential trails which is comparable to our differentials.

PRINCE. Canteaut *et al.* [109] showed differential attacks on 10 rounds of Prince, by considering multiple differential trails. In their attack they use 12 differentials, for six rounds with a probability of $2^{-56.42}$ by summing up 1536 single-trails. The differential we found for 6 rounds only has a probability of about 2^{-62}, but does not lead to further improvements of the attack.

SPARX-64. Ankele and List [26] (also see Chapter 7 for further details) studied truncated-differential attacks on 16 rounds of Sparx-64/128 and used single differential trails, for the first part of the 14-round distinguisher, and truncated the second part of the distinguisher. The designers of Sparx-64 claim that Sparx is differential secure for 15 rounds, however, by considering the differential effect of Sparx-64, also in comparison with Speck-64, it seems likely that there exist differentials with more than 15 rounds with a data complexity below using the full codebook.

[5] Single-Trail differentials of Sparx [135] are proven to reach 15 rounds, while the authors mention that they don't expect the bound to be tight.
[6] The best differentials for Simon-64 reach 23 rounds with $2^{-63.91}$ [232].
[7] The best differentials for Speck-64 reach 15 rounds with $2^{-60.56}$ [312].

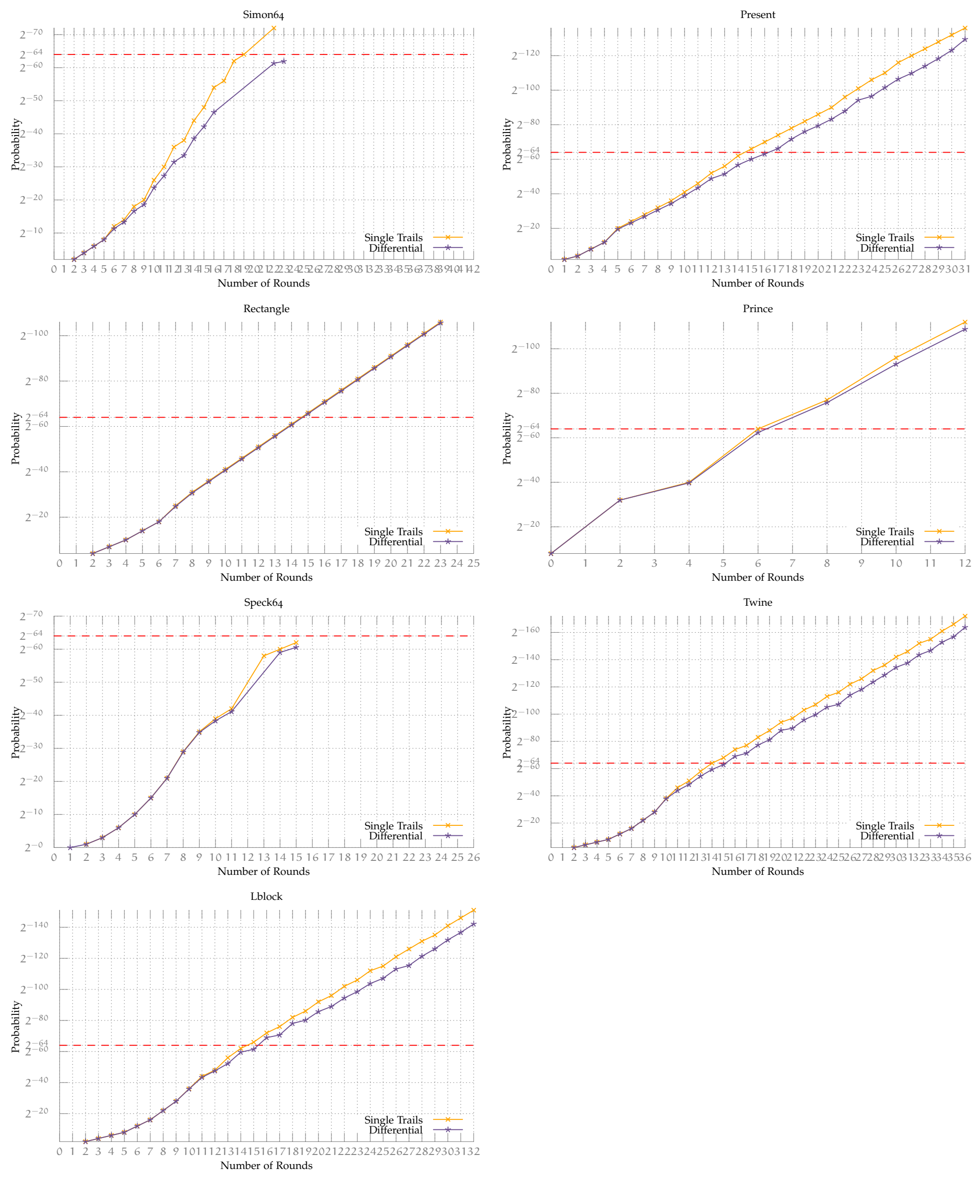


Figure 45: Probability for the best single trails and differentials for various rounds of different block ciphers. 1st row: SIMON-64 (left) and PRESENT (right), 2nd row: RECTANGLE (left) and PRINCE (right), 3rd row: SPECK-64 (left) and TWINE (right), 4th row: LBLOCK (left)

SIMON-64. Abed *et al.* [7] presented differential attacks on SIMON-64, where they used a 21-round distinguisher with a probability of $2^{-61.01}$. Better distinguishers are reported by [232] for 23 rounds with a probability of $2^{-63.91}$. The differentials we found are in line with previous results.

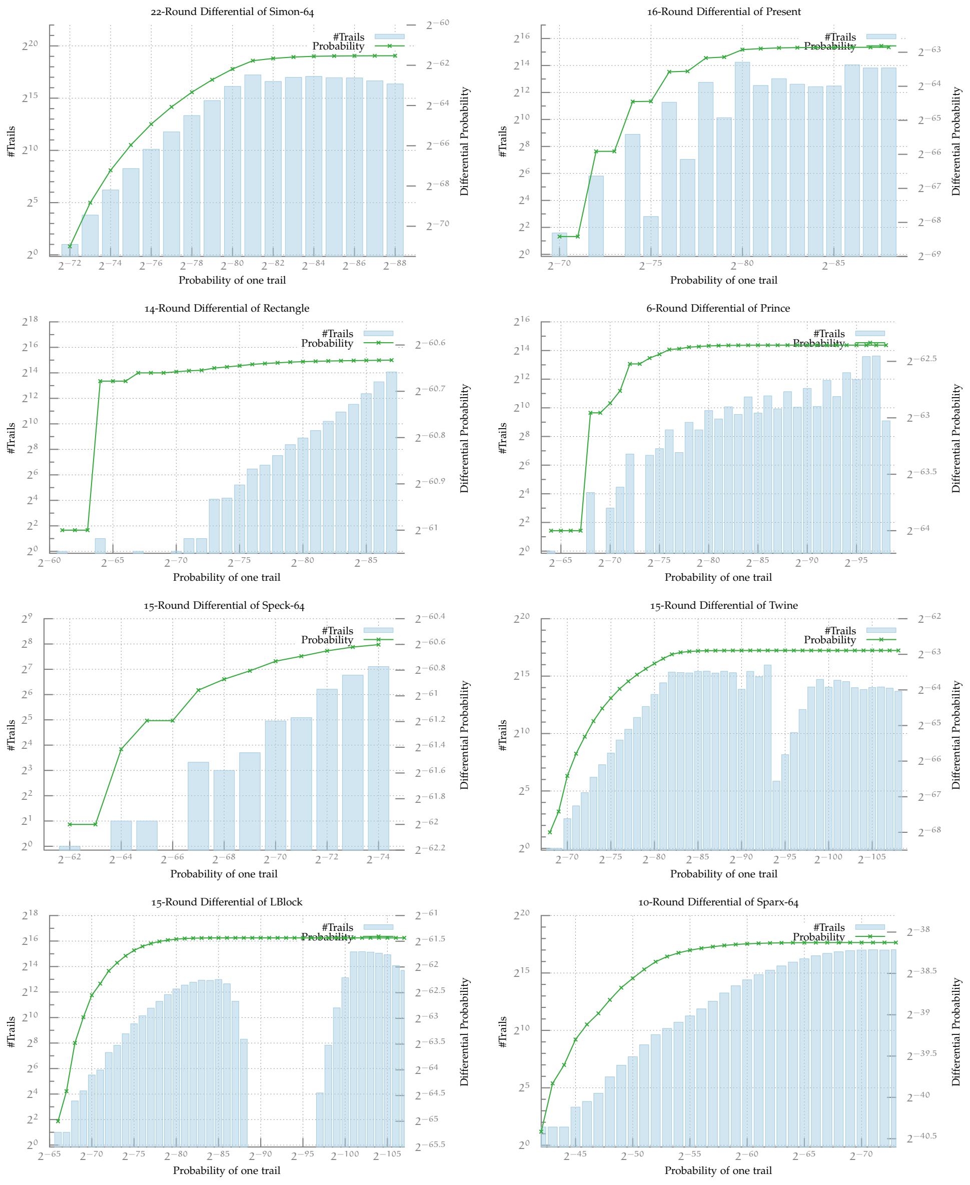


Figure 46: Distribution of the trails contributing to the best Differential for various block ciphers. 1st row: Simon-64 (left) and Present (right), 2nd row: Rectangle (left) and Prince (right), 3rd row: Speck-64 (left) and Twine (right), 4th row: LBlock (left) and Sparx-64 (right)

speck-64. Song *et al.* [312] presented 20-round attacks on Speck-64 by constructing a distinguisher from two short trails where they concatenated the two trails to a 15-round trail with probability $2^{-60.56}$. The distinguishers used in the attack are already based on differentials and the differentials we found do not lead to any improvement.

TWINE. Biryukov *et al.* [75] showed a 25-round impossible differential attack and a truncated differential attack on 23 rounds by chaining several iterated 4-round trails together. In the paper the authors also considered differentials for 12 rounds with a probability of $2^{-52.08}$ and 16 rounds with probability $2^{-67.59}$. The best differential that we found reaches 15 rounds with a probability of $2^{-62.89}$.

LBLOCK. Wang *et al.* [344] published a 24-round impossible differential attack on LBlock. Due to the nature of impossible differential attacks, trails with probability 1 are used for constructing these. The best differential that we found reaches 15 rounds with a probability of $2^{-61.43}$.

3.5 EXPERIMENTAL VERIFICATION AND THE INFLUENCE OF KEYS

In Section 3.1 we made several assumptions in order to compute $\mathrm{DP}(Q)$ and in this section we compare the theoretical estimates with experiments for reduced-round versions. This serves two purpose: First we want to see how close our estimate for $\mathrm{DP}(\alpha, \beta)$ is and secondly we want to see the distribution over the choice of the keys. Specifically, we are interested in the number of pairs

$$\delta_K(\alpha, \beta) = \#\{x \in \mathbb{F}_2^n \mid E_K(x) \oplus E_K(x \oplus \alpha) = \beta\}. \tag{54}$$

This number of *good* pairs will vary over the choice of key. For a random process we would expect that the number of valid pairs is about $\mathrm{DP} \cdot 2^n$ and follows a Poisson distribution.

Definition 46. Let X be a Poisson distributed random variable representing the number of pairs (a, b) with values in $\mathbb{F}_2^n$ following a differential $D = (\alpha \xrightarrow{f} \beta)$, that means $f(a) \oplus f(a \oplus \alpha) = \beta$, then

$$\Pr(X = l) = (2^n p)^l \frac{e^{-(2^n p)}}{l!} \tag{55}$$

where p is the probability of the differential.

In the following, we experimentally verify differentials for Skinny, Speck and Midori for a large number of random pairs of plaintexts and a random choice of keys to see how good this approximation is.

3.5.1 Skinny

As a first example we look at Skinny-64. We use the 6-round differential

$$D = (\texttt{0x0000010010000041}, \texttt{0x4444004040044044})$$

for Skinny-64. The best trail which is part of D has a probability of 2^{-32} and by collecting all trails (100319) contributing to this differential we estimate $\mathrm{DP}(D) \approx$

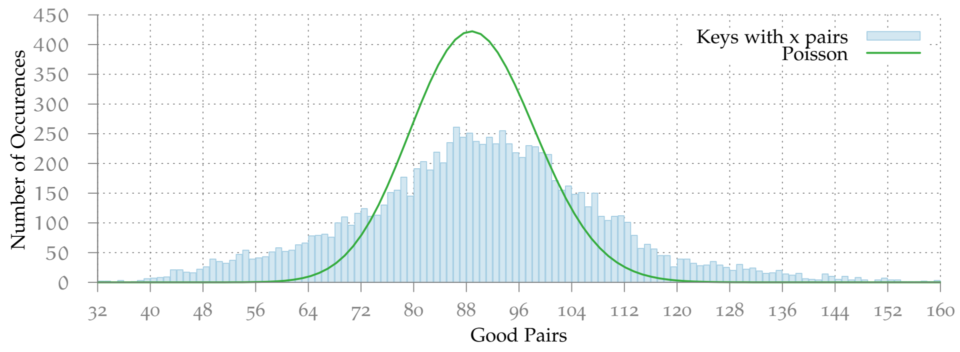


Figure 47: Distribution of $\delta_K(Q)$ over a random choice of K for 6-round SKINNY-64.

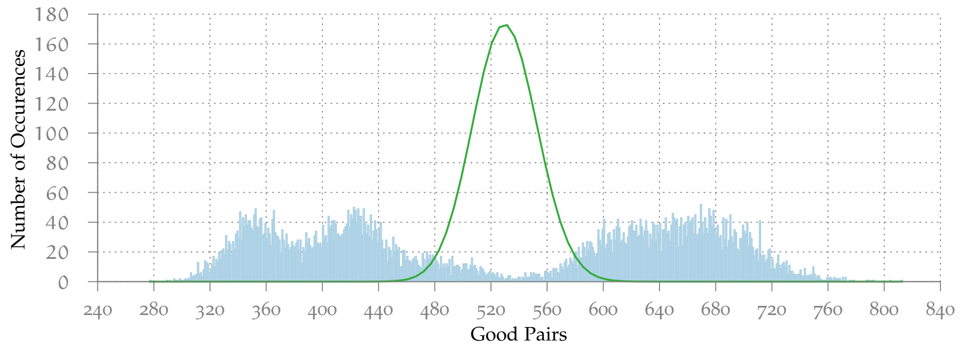


Figure 48: Distribution of $\delta_K(Q)$ over a random choice of K for 7-round SPECK-64.

$2^{-23.52}$. We try out 2^{30} randomly selected pairs for 10000 keys and count the number of pairs following D. From our estimate we would expect that on average we get about 89 pairs for a key.
As one can see from Figure 47 our estimate of DP(D) provides a good approximation for the distribution over the keys, although the distribution has a larger variance than we expected.

3.5.2 SPECK

For SPECK-64 we look at the differential

$$D = ((\texttt{0x40004092}, \texttt{0x10420040}), (\texttt{0x8080A080}, \texttt{0x8481A4A0}))$$

over 7 rounds. The best trail in D has a probability of 2^{-21} and this only slightly improves to about $2^{-20.95}$ using six additional trails. We again run our experiments for 2^{30} randomly selected pairs for 10000 keys and count the number of pairs following D. On average we would expect 530 pairs.
In Figure 48 it can be seen that for 7-round SPECK-64 the distribution is bimodal and we over- and under-estimate the number of valid pairs for most keys.

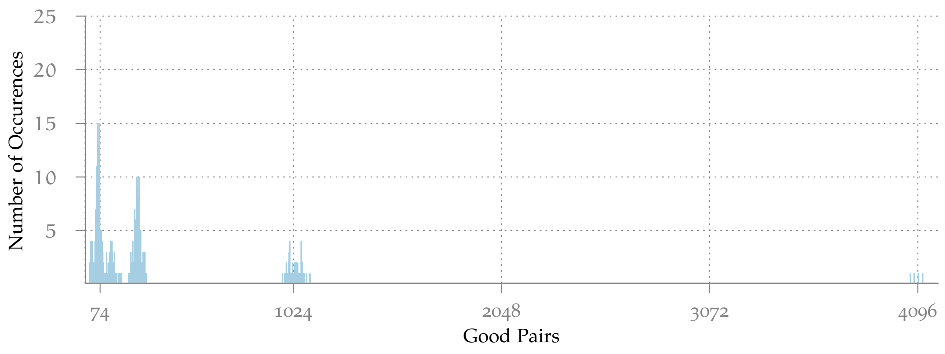


Figure 49: Distribution of $\delta_K(Q)$ over a random choice of K for 4-round MIDORI-64. We omitted the 2545 keys with zero good pairs in this plot.

3.5.3 MIDORI

For MIDORI-64 we look at the differential

$$D = (\texttt{0x0200200000020000}, \texttt{0x0202220020020020})$$

over 4 rounds. The best trail in D has a probability of 2^{-32} and this improves to about $2^{-23.79}$ using 896 additional trails. We again run our experiments for 2^{30} randomly selected pairs for 3200 keys and count the number of pairs following D. On average we would expect about 74 pairs.
In Figure 49 it can be seen that for 4-round MIDORI-64 the distribution is very different from the previous cases. For some keys the probability is significantly higher and for ≈ 80% of the keys we get zero good pairs. This means that for a large fraction of keys we actually found an impossible differential and one should be careful when constructing differential distinguishers for MIDORI. In particular it would be interesting to classify this set of impossible keys and we leave this as an open problem. Moreover, this also implies the existence of a large class of weak keys, that has also been observed in the invariant subspace attacks on MIDORI-64 [166, 220, 325].

3.6 CONCLUSION

In this Chapter, we showed for several lightweight block ciphers that the gap between single differential trails and differentials can be surprisingly large. This leads to significantly higher probability of differentials in several designs and allows us to have differential distinguishers covering more rounds.
We provided a simple framework to automate the process of collecting many differential trails that are contributing to the probability of a differential. We hope this will encourage future designs of cryptographic primitives to apply our methodology in order to provide better bounds on the security against differential cryptanalysis.

Further we verified differentials for a reduced number of rounds experimentally and showed that our improved estimates of the probability of differentials of Skinny closely resembles what happens in experiments. However, we can also observe that some commonly made assumptions on the distribution of good pairs following a differential over the choice of keys has to be made very carefully. For instance, the results for Speck and Midori indicate that one needs to be very careful in presuming that the estimates apply to all key values.
One restriction with our approach is that while our differentials improve a lot upon the single trails, we do not always find the optimal differentials. Identifying the most promising differentials is still an open question and using the best single trail as a starting point might not always be optimal.

4 ANALYSIS OF LOW-ENERGY 4-BIT S-BOXES

CONTENTS

EXECUTIVE SUMMARY. In this chapter, we study 4-bit cryptographic S-boxes with a focus on low-energy designs. Many devices in resource constrained environments, the Internet of Things (IoT) in general, are powered by batteries. Thus, those devices often operate on a tight power/energy budget. Medical implants such as pacemakers, insulin pumps or brain implants are some examples. Security and privacy is crucial in the communication channel of these devices.
Lightweight cryptography is an active field of research and there have been many cipher proposals in the last few years. In conventional cryptographic standards the trade-off between security and performance is optimised for high performance environments. However, in resource constrained environments those cryptographic standards are difficult or impossible to implement. Hence, lightweight ciphers have been optimised for area, power consumption, memory complexity, latency and throughput. Yet, little work has been done on energy efficient ciphers.
We are trying to fill this gap and give a detailed study on energy-efficient design strategies for block ciphers. Moreover, we concentrate our research on Substitution-Boxes (S-boxes) that are an important building block of Substitution Permutation Networks (SPN). We analyse all *optimal* 4×4-bit S-boxes, and classify them in two groups, based on Present-like and Prince-like designs, that we show to be optimal block cipher designs for low-energy consumption. In that context we further analyse all involutory 4-bit permutations, and study the differential and linear branch numbers of all optimal 4-bit affine equivalence classes.

As a result, we give recommendations for optimal low-energy S-boxes, that cipher designers can use in their ciphers, for instance in the upcoming lightweight cryptography standardisation process by the National Institute of Standards and Technology (NIST).

DECLARATION OF AUTHORSHIP. The work described in this chapter is based on the paper: *Fighting Climate Change - Towards Energy-Efficient S-boxes* which is currently in preparation for publishing. The paper is joint work with Erik Boss, Miroslav Knežević, Marco Martinoli, and Tolga Yalçın. All authors contributed equally to the results of the paper. The contributions of the author are the following:

- Implementation of the search for involutory S-boxes,
- Implementation of the search for S-boxes with high differential/linear branching number,
- Cryptographic analysis of all 4×4-bit S-boxes

4.1 INTRODUCTION

Lightweight cryptography emerged from the lack of primitives that are capable to run in highly constraint but interconnected environments (*i.e.,* automotive systems, sensor networks, RFID tags, healthcare and smart grids) the Internet of Things (IoT) in general. Security and privacy is very important in all these areas. Yet, modern cryptographic algorithms are designed to run on desktop/server systems and are not suitable for these restricted environments. Lightweight cryptographic algorithms aim to provide solutions tailored for resource constraint environments without compromising efficiency or security. A significant amount of effort has been done in recent years by the cryptographic community to tackle these limitations; this combines efficient implementations of conventional cryptographic algorithms and the design of new lightweight primitives.
A lot of block ciphers have been designed were the aspects of lightweightness are measured by *e.g.,* area, power consumption, memory complexity, latency and resistance against side channel attacks. These ciphers include HIGHT [174], PRESENT [95], CLEFIA [310], LED [168], PRINCE [101], KATAN/KATANTAN [105], PICCOLO [309], SPARX [135], SKINNY [50] and GIFT [40].
However, there has been little work on energy-efficient designs with just MIDORI [37] that has been optimised specifically for low-energy. Energy consumption is a measurement of a power source (*i.e.,* battery) over time (*i.e.,* the duration of executing an operation). Hence, the measurement of energy is a more relevant parameter than power to measure the efficiency of a cipher. Certainly, optimising a cipher for low-energy has a wide range of applications as a lot of devices in constraint environments are battery powered and they are therefore working with a tight power/energy budget. Especially crucial are several medical implants such as pacemakers, brain implants for epilepsy and seizure patients as well as insulin pumps for diabetes patients.

Recently, the US National Institute of Standards and Technology (NIST) has also published a call for a standardisation process of lightweight ciphers [43]. In the call, they specifically devote one of the main design requirements to be low-energy:

> "The algorithms should be flexible to support various implementation strategies (low energy, low power, low latency)."

We concentrate on some of the underlying components of a block cipher. In particular, we focus on cryptographic Substitution-Boxes (S-boxes) that are optimised for low-energy consumption. In that context, we analyse all 4×4-Bit S-boxes, but target our analysis on cryptographically optimal S-boxes as defined by Leander and Poschmann [219].

We analyse several design aspects of low-energy block ciphers, and recommend two designs in particular, PRESENT [95] and PRINCE [101], as they are optimal candidates for low-energy ciphers. Thus, additionally to low-energy consumption, we categorise the analysed S-boxes in those two groups relevant to their application area.

For PRESENT-like designs, we argument that low-energy S-boxes should also have a high differential/linear branch number to ensure that in an overall block cipher design the cipher reaches full diffusion after a short number of rounds. Therefore, we coincide low-energy S-boxes with S-boxes that have an optimal differential/linear branch number.

PRINCE-like designs build upon an involutory diffusion layer and the α-reflection property of reflection ciphers. This property allows to re-use parts of the cipher, by mirroring the rounds around an involutory middle construction, that then enables that decryption is the same as encryption with a related key. For a more energy-efficient design, we therefore study all involutory S-boxes that have an optimal low-energy consumption.

RELATED WORK. In his Ph.D thesis, De Cannière [128] firstly studied 4×4-Bit S-boxes and provided a simple and efficient algorithm to classify them in 302 affine equivalent classes. Leander and Poschmann [219] classified all *optimal* 4×4-Bit S-boxes, up to affine equivalence, and identified that there are 16 *optimal* classes of S-boxes with respect to differential and linear attacks. Further improvements were made by Saarinen [293], who provided extended properties of the optimal S-box classes, up to linear equivalence. In his analysis, he showed that the 16 optimal affine equivalent classes not only share their differential and linear bounds but also have equivalent algebraic properties and circuit complexities. Moreover, he defines 4 classes of *golden* S-boxes with optimal differential/linear bounds and algebraic properties. In 2018, Sarkar and Syed [297] analysed differential and linear branch number for S-boxes and obtained bounds for the differential and linear branch numbers of permutations of $\mathbb{F}_2^n$. They also showed that the maximal differential branch number for 4×4-Bit S-boxes can only be achieved by affine permutations.

Banik *et al.* [39] firstly explored the energy-efficiency of lightweight block ciphers where they analysed the building blocks of Substitution-Permutation Network (SPN) ciphers and identified the parameters that affect the energy consumption of a cryptographic primitive. Later, Banik *et al.* proposed MIDORI [37], the first block cipher optimised for low-energy. In their design, they tried to optimise each component of the electrical circuit as well as the entire architecture for energy. Recently, Banik *et al.* proposed another cipher, GIFT [40], which is optimised in all domains (faster and smaller). The designers claim that any cheaper choice of S-box would lead to a very weak cipher. Moreover, GIFT is one of the most energy efficient ciphers as of today.

4.2 DESIGN CONSIDERATIONS FOR LOW-ENERGY

The energy consumption for any given block ciphers is dominated by the following factors.

- **Clock Frequency:** The energy consumption of charging and discharging the capacitive load of a gate when an output transition (Note, we will later define it as switching activity) occurs is highly dependent on the clock frequency of the circuit. For high frequencies, energy due to leakage just minimally contributes to the total energy consumption [39].
- **Architecture of Cipher Components:** The architecture of the underlying cipher components contributes highly to the energy consumption. While the *Canright* architecture [106] seems to be the most compact representation in terms of area, the designers of MIDORI [37] argument that the *Decoder-Switch-Encoder* architecture [64] is optimal for energy.
- **Latency:** The energy consumption further depends on the latency of the circuit [199]. Latency in a block cipher can be defined as a measurement of the time needed to encrypt a single message block:

  $$\text{Latency} = N \cdot t_{cp},$$

  where N is the number of cycles to encrypt a single message block, and t_{cp} is the time used for encryption by using the critical path of the circuit.
- **Width of Data-Path:** In hardware implementations, unrolling the round function of a block cipher improves the throughput compared to fully serialised designs, where an implementer aims to compute one round in one clock cycle. Unrolling a block cipher allows to compute several round functions in fewer cycles, at the cost of increased area of the combined circuit. In an unrolled design, the energy consumption is improved, as energy does not need to be spent to store signals at the output of each round [132]. Nevertheless, Banik *et al.* [39] showed that fully serialised designs are better for low-energy by analysing the energy consumption of all components in unrolled and fully serialised designs.

4.2.1 Fundamental Principles of Power/Energy Consumption

Optimising cryptographic building blocks for low-energy consumption requires a designer to improve the area, latency and power consumption, simultaneously. In recent years, the focus of designers has been to reduce the hardware area [38] and

Table 14: Comparison of energy consumption for round-based implementation of various lightweight ciphers synthesized with STM 90nm Standard cell library.[†]

Cipher	Area (GE)	Delay (ns)	Cycles	Power (μW/@10MHz)	Energy (pJ)
MIDORI-64/128	1542	2.06	17	60.6	103.0
PRINCE-64/128	2286	4.06	13	111.3	144.7
RECTANGLE-64/128	1637	1.61	27	76.2	206.0
GIFT-64/128	1345	1.83	29	74.8	216.9
PRESENT-64/128	1560	1.63	33	71.1	234.6
PICCOLO-64/128	1868	2.32	32	79.4	254.1
SKINNY-64/128	1477	1.84	37	80.3	297.0
SIMON-64/128	1458	1.83	45	72.7	327.3
LED-64/128	1831	5.25	50	131.3	656.5
AES-128/128	7215	3.83	11	730.3	803.3

the latency [101] of block ciphers. Power and energy are correlated parameters, as power can be essentially defined as the energy that is consumed per a unit of time. The power consumption in a fully unrolled hardware circuit is defined as:

$$P_{total} = P_{switching} + P_{leakage}$$

where $P_{switching} \gg P_{leakage}$. Power leakage $P_{leakage}$ results from transistors in CMOS gates when the transistor is off. If the clock frequency is above 1MHz the leakage is negligible [39]. The switching activity *sw* of a gate is defined as the average number of output transitions from $0 \to 1$ and $1 \to 0$. Table 15 gives an overview of the switching activity for several CMOS gates. When measuring the area of logic gates, the term *Gate Equivalence (GE)* is often used. Hereby, the logical gates are compared to the NAND-gate, which is the most common gate, and normally most of the other gates are constructed with NAND-gates.
The switching power consumption can be estimated as

$$P_{switching} \approx C_{eff} \cdot V_{DD}^2 \cdot f_{clk} \cdot sw$$

where C_{eff} is the effective capacity of the gate, V_{DD} is the supply voltage, f_{clk} is the clock frequency of the circuit, and *sw* is the switching activity of all gates.
The energy consumption of a hardware circuit is a measurement of the time integral of power.

$$\begin{aligned} E = P \cdot t = P \cdot \frac{N}{f_{clk}} \\ \approx C_{eff} \cdot V_{DD}^2 \cdot N \cdot sw \end{aligned}$$

[†] Data obtained from [37, 39, 40]

Table 15: Comparison of CMOS gates.

Gate	Switching Activity	Gate Equivalent (GE)
`NOT`	100%	0.67
`NAND`	75%	1.0
`NOR`	75%	1.5
`AND`	25%	1.6
`OR`	25%	2
`XOR`	50%	3

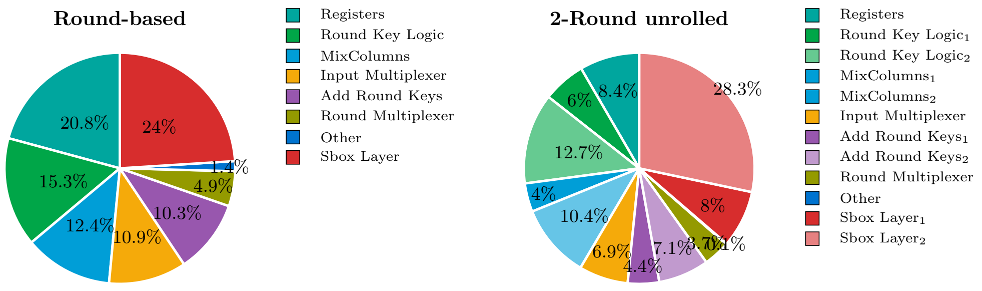


Figure 50: Energy shares for AES-128[‡]. **(Left):** Round-based, **(Right):** 2-Round unrolled.

where P is the power consumption of the hardware circuit, t is the time until one full block is encrypted and, N is the number of cycle to encrypt a single message block.
The energy/power consumption of a hardware circuit can be improved by either, reducing the circuit area (*e.g.*, by serialising the circuit) which reduces the capacity of the circuit C_{eff}, but would increase the number of cycle to encrypt a single message block N. Moreover, one can reduce the switching activity *sw* by choosing different CMOS gates (See Table 15) or clock gating [292]. Moving to a smaller CMOS technology would essentially decrease the capacity of the circuit C_{eff}, and the supply voltage V_{DD}, but the leakage currents $P_{leakage}$ would increase. Another option is to reduce the operating clock frequency f_{clk} or to reduce the latency, which would also decrease the number of cycle to encrypt a single message block N in the hardware circuit.

4.2.2 Block Cipher specific Design Considerations for Low-Energy

The energy consumption of a block cipher depends on several factors as mentioned above. Figure 50 shows the energy consumption for the underlying building blocks of AES-128. In a round-based implementation of AES-128 the separate steps of

[‡] Data obtained from [39](Figure 2)

the round function (S-box Layer, `MixColumns`, `AddRoundKey`) use about 46.7% of the total energy consumption. The key schedule takes about 15.3% of the energy consumption, and the remaining 38% are spent to store the states and general logic (Registers, Input Multiplexer, and Round Multiplexer). The energy consumption in a block cipher can be reduced by considering the following design decisions:

- **Round-based Ciphers:** While it seems counter-intuitive that a fully serialised iterative block cipher consumes less energy, as it obviously requires a longer execution time, Banik *et al.* [39] showed that round-based designs are actually the best in terms of low-energy. In their work they studied AES-128, and showed that a round-based implementation needs about 350.7 pJ, while 2, 3, 4, 5, and 10-round unrolled designs need 593.6, 1043, 1416.5, 1634.4, and 2129.5 pJ, respectively.

- **Substitution Permutation Networks:** Block ciphers can be split in two main design strategies, Feistel networks and Substitution Permutation Networks. In Feistel networks the encryption and decryption function can be designed with low overhead making them easy to implement. Yet, the fact that normally just half of the state is updated by a non-linear function requires more rounds for the same security. In contrary, in Substitution Permutation Networks the round transformation is applied to the entire state requiring less rounds for the same security bounds. Therefore, Substitution Permutation Networks are more suitable in low-energy designs.

- **Few Complex Rounds:** In general, low-energy ciphers with a few more complex rounds require less energy than ciphers with a simple round function, but a vast number of rounds. This can be observed for example by comparing the energy consumption of PRINCE and MIDORI to SIMON and SKINNY. While PRINCE and MIDORI have 11 and 16 more complex rounds, SIMON and SKINNY have 44 and 36 very simple rounds. Nevertheless, MIDORI requires 103.0pJ, PRINCE requires 144.7pJ, while SKINNY requires 297.0pJ and SIMON requires 327.3pJ. A comparison of several lightweight ciphers is given in Table 14.

- **Low Area and Signal Delay:** By reducing the area required for the implementation of a circuit, the capacity is reduced, which leads to less power leakage and a lower energy consumption. Furthermore, a reduced area improves the signal delay in the circuit.

- **Smaller S-boxes:** The designers of MIDORI [37] observed that 4-bit S-boxes consume less energy than 8-bit S-boxes, as the signal delay is lower in a 4-bit S-box. However, designs based on 4-bit S-boxes normally require more rounds to achieve the same security bounds regarding differential/linear cryptanalysis. Nevertheless, as shown by Banik *et al.* [37] 4-bit S-boxes still outperform 8-bit S-boxes by a factor of 2.

- **Involutions:** Cipher designs like NOEKEON and the middle layer of PRINCE that relies on involutions, facilitates that some components can be reused in the implementation of the encryption/decryption algorithms. This effectively reduces the area and improves the energy consumption of a cipher.

4.2.3 S-box specific Design Considerations for Low-Energy

By analysing the energy consumption of AES as illustrated in Figure 50, one can observe that a vast majority of the energy consumed by a block cipher is spent by S-boxes. In a round-based design about 24% of the total energy consumption is spent in the S-box layer. When unrolling the round function, where we unroll two rounds of AES-128, around 8% of the energy consumption is spent for the S-box layer of the first round and around 28.3% are spent for the S-box layer of the second round. Consequently, by reducing the energy used in an S-box one should be able to reduce the energy consumption of the whole block cipher. In the following, we give some S-box specific design considerations for low-energy S-boxes.

- **Architecture:** While the Canright [106] architecture is the most compact in terms of area it is less optimal for low-energy. Implementing the S-box directly as Look-up-table (LUT) requires a large amount of area [302]. The most energy-efficient architecture for S-boxes is the Decoder-Switch-Encoder architecture [64, 354].

- **Low Area and Signal Delay:** Similar as for block cipher designs, reducing the area of an S-box reduces the signal delay and decreases the capacity of the circuit.

- **Algebraic Structure:** The algebraic structure of an S-box has a huge impact on the switching activity in the circuit. While non-linear gates such as `AND`, `OR` in average just switch 25% on average, `XOR` gates switch 50% on average and optimal gates in terms of area, like `NAND`, `NOR` gates switch 75% on average (Note, see Table 15 for details).

- **Mathematical Properties:** Involutory S-boxes do not require to implement an inverse S-box, as the S-box is its own inverse. Therefore, involutory S-boxes are ideal for low-energy ciphers.

4.3 ANALYSIS OF OPTIMAL LOW-ENERGY S-BOXES

4.3.1 Requirements for Optimal Low-Energy S-boxes

An optimal *low-energy* S-box has to fulfil several criteria. In Section 4.2 we already specified the performance considerations for an low-energy S-box. Furthermore, in Chapter 2.5.1 we outlined the necessary criteria for a cryptographically secure S-box. For an *optimal* low-energy S-box we first defined the following criteria:

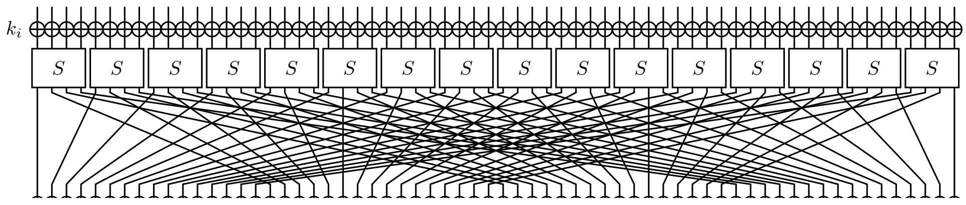


Figure 51: Round function of PRESENT.

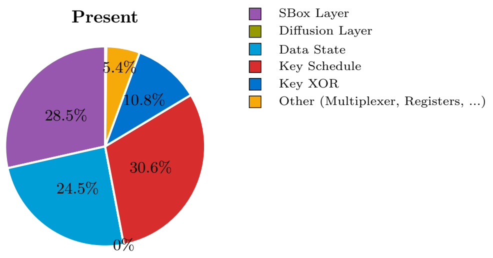


Figure 52: Area distribution of PRESENT synthesized in the Virtual Silicon (VST) 0.18µm standard cell library.

Table 16: Area distribution of PRESENT [¶].

	Area	Distribution
S-box Layer	448 GE	28.5%
1 S-box	$\approx$ 28 GE	1.78%
Diffusion Layer	0 GE	0%
Data State	384 GE	24.5%
Key Schedule	480 GE	30.6%
Key XOR	170 GE	10.8%
Other	86 GE	5.4%
Total	1570 GE	100%

Cryptographic Properties :

- Max. diff. probability = 2^{-2},
- Max. lin. bias = 2^{-2},
- Branch number (diff/lin) = 3,
- Algebraic degree = 3,
- No fixed points,
- Involutory

Performance Properties :

- Low area $\leqslant$ 16 GE
- Low power $\leqslant$ 350 nW/@1MHz
- Low number of gates $\leqslant$ 20 gates
- Low circuit complexity

These requirements were inspired by the design rationales of the designs for MIDORI, GIFT and PRINCE. The designers of MIDORI searched for involutory S-boxes with maximum differential/linear probabilities/bias of 2^{-2}. The designers of GIFT further specified *no fixed points*, which is relevant in bit-based diffusion layer designs, an algebraic degree of 3 for the whole S-box (*i.e.*, the component functions can have a lower degree), and a maximum area of 16 GE for the S-box (*i.e.*, the PRESENT S-box has 21.33 GE, the SKINNY S-box has 13.33 GE). However, the designers relaxed the requirements for the maximum differential probability to $2^{-1.415}$ and ensured that there are no high probability differential trails when designing the diffusion layer. The designers of PRINCE specified maximum differential/linear probability/bias of 2^{-2} and further restricted the search space by requiring that all non-zero component functions of the S-box should have an algebraic degree of 3.

[¶] Data obtained from [95]

4.3.2 Block Cipher Design Strategies for Low-Energy Ciphers

The ambition behind our analysis of low-energy S-boxes is to use the best performing S-boxes in the design of a block cipher. In Section 4.2.2, we already outline the requirements for low-energy block cipher designs. Now we want to further study some lightweight cipher designs to analyse the contribution of the different building blocks in a block cipher, to further motivate the use of energy-efficient S-boxes.
In this work, we focus on the block ciphers PRINCE and PRESENT. In the following, we give details for our choices.

PRESENT [95] is a lightweight block cipher that is optimised for small area. The cipher supports a block size of 64-bits and key sizes of 80 and 128-bits. PRESENT recommends 31 rounds where one round is illustrated in Figure 51. The diffusion layer of PRESENT is a simple bit-permutation, that can be implemented in hardware just by wiring. Compared to the diffusion layer of *i.e.,* AES, that uses a large MDS matrix, a bit-permutation is optimal to reduce area and consequently also the energy consumption. The efficiency characteristics (*e.g.,* energy, power, area, ...) and also the cryptographic strength of PRESENT-like designs therefore, mainly depends on its S-box Layer. PRESENT uses a simple 4-bit S-box $S : \mathbb{F}_2^4 \to \mathbb{F}_2^4$. The differential branch number of PRESENT is three, while the linear branch number just reaches two. The algebraic degree is three. The maximal differential probability is 2^{-2} and the linear probability bias is 2^{-2}. Figure 52 and Table 16 show the area distribution of the components of PRESENT. It is easy to see that the S-box layer has the most impact in PRESENT. Hence, this also follows for the energy consumption of the components.
The energy consumption of PRESENT is 234.6 pJ in the implementation by [39] in the STM 90nm standard cell library. This puts PRESENT in the middle of our energy comparison for lightweight block ciphers as illustrated in Table 14. However, the designers of GIFT [40] have shown that changing the S-boxes to smaller ones, and adapting the diffusion layer to provide strong cryptographic security bounds can significantly reduce the area and the energy consumption.

PRINCE [101] is a lightweight block cipher that is optimised for latency when implemented in hardware. The cipher supports a block size of 64-bits and a key size of 128-bits. PRINCE recommends 12 rounds as illustrated in Figure 53. The diffusion layer uses a combination of an involutory binary matrix and AES `ShiftRows`. PRINCE uses a simple 4-bit S-box $S : \mathbb{F}_2^4 \to \mathbb{F}_2^4$. The differential and linear branch numbers of PRINCE are only two. The algebraic degree is three. The maximal differential probability is 2^{-2} and the linear probability bias is 2^{-2}. Figure 54 and Table 17 show the area distribution of the components of fully unrolled PRINCE for unconstrained (*i.e.,* 1MHz) and constrained (*i.e.,* 65MHz) clock frequencies. Similar as for PRESENT, the S-box layer of PRINCE consumes the most area with around 28.5%. Hence, this also follows for the energy consumption of the components.
The energy consumption of PRINCE is 144.7 pJ in the implementation by Banik *et al.* [39] in the STM 90nm standard cell library. This puts PRINCE in the second place after MIDORI in our energy comparison for lightweight block ciphers as illustrated in Table 14. The design of PRINCE is optimal for low-energy as it requires just 12 rounds compared to *i.e.,* PRESENT or GIFT with 31 and 28 rounds, respectively.

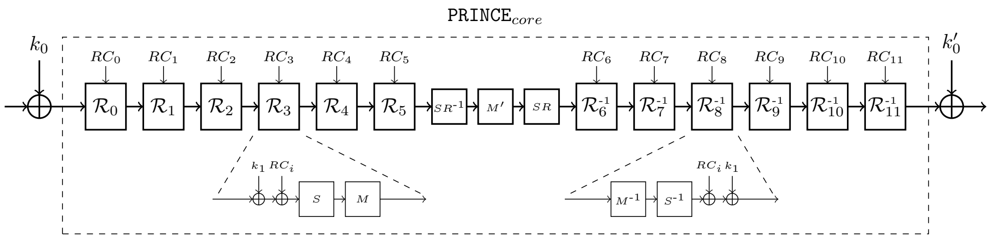

PRINCE$_{core}$
k_0
k'_0
RC_0 RC_1 RC_2 RC_3 RC_4 RC_5 RC_6 RC_7 RC_8 RC_9 RC_{10} RC_{11}
$\mathcal{R}_0$ $\mathcal{R}_1$ $\mathcal{R}_2$ $\mathcal{R}_3$ $\mathcal{R}_4$ $\mathcal{R}_5$
SR^{-1} M' SR
$\mathcal{R}_6^{-1}$ $\mathcal{R}_7^{-1}$ $\mathcal{R}_8^{-1}$ $\mathcal{R}_9^{-1}$ $\mathcal{R}_{10}^{-1}$ $\mathcal{R}_{11}^{-1}$
k_1 RC_i S M
M^{-1} S^{-1} RC_i k_1

Figure 53: Round function of PRINCE.

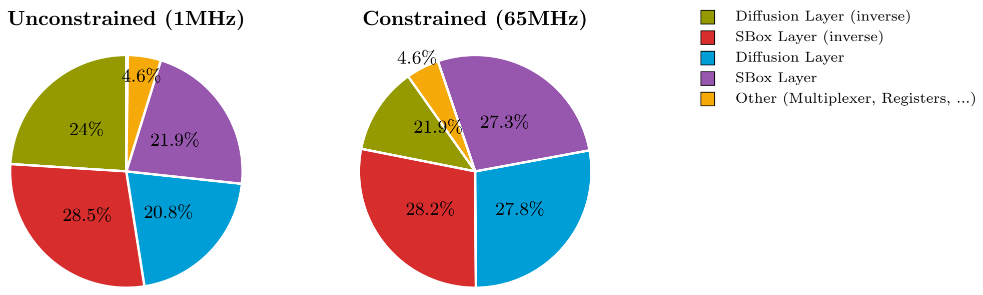


Figure 54: Area distribution of fully unrolled PRINCE. **(Left):** clock frequency is unconstrained (*i.e.*, 1MHz), **(Right):** clock frequency is constrained to 65MHz.

Table 17: Area distribution of fully unrolled PRINCE. We give details for unconstrained (1MHz)/constrained (65MHz) clock frequencies. PRINCE is synthesized with the 40nm CMOS library.

	Unconstrained	Constrained
Fully Unrolled Cipher	7716 GE	47758 GE
Forward Round	623 GE	4065 GE
Diffusion Layer	215 GE	1298 GE
S-box Layer (16 S-boxes)	226 GE	1988 GE
1 S-box	≈ 14 GE	≈ 78-161 GE
Backward Round	696 GE	4415 GE
Diffusion Layer	248 GE	1601 GE
S-box Layer (16 S-boxes)	295 GE	2055 GE
1 S-box	≈ 18 GE	≈ 96-151 GE

PRINCE uses an involutory binary matrix in the diffusion layer and is a reflection cipher that uses the α-reflection property that enables the cipher to reuse parts of the cipher components for both the forward and backwards rounds.

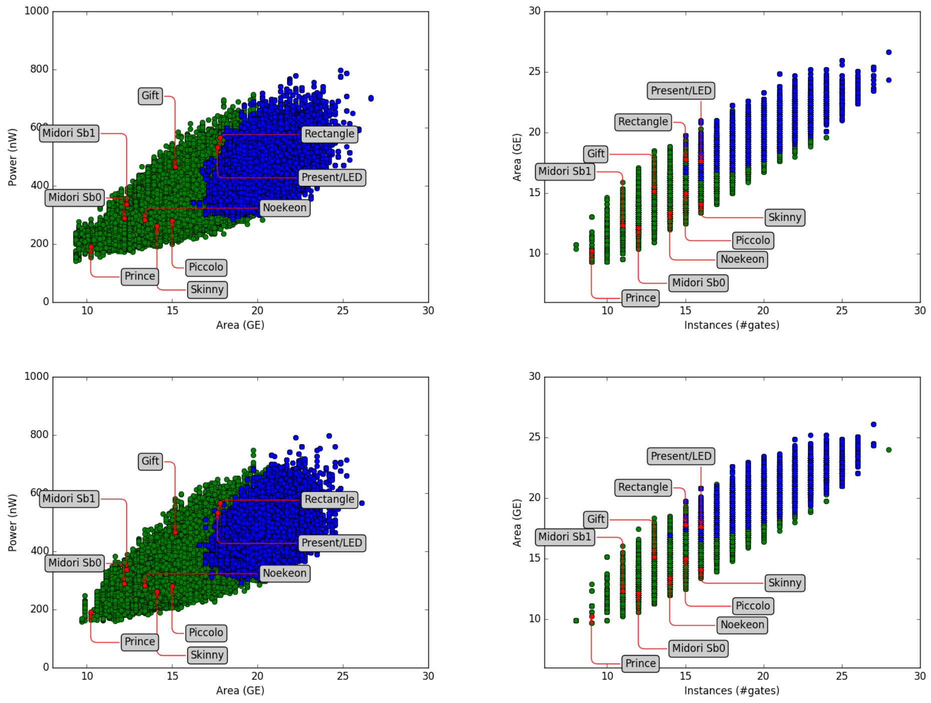


Figure 55: Green circles indicate *involutory* S-boxes and blue circles indicate S-boxes with a high *branching number*. **(Top Left):** Area/power plot of the *golden* class 160. **(Top Right):** Instances/area plot of the *golden* class 160. **(Bottom Left):** Area/power plot of the *golden* class 163. **(Bottom Right):** Instances/area plot of the *golden* class 163.

4.3.3 Analysis of Optimal 4-bit S-boxes for Low-Energy Consumption

In our search for optimal low-energy S-boxes, we focused on the requirements defined in Section 4.3.1. Moreover, we made use of the affine equivalence classes defined by De Cannière [128], and we particularly focused on the set of 16 *optimal classes* as classified by Leander and Poschmann [219] and the *golden classes* classified by Saarinen [293].

Furthermore, we restricted all 4×4-bit S-boxes to permutations, which reduced the search space from 2^{n2^n} to $2^{n!}$. In the case of 4×4-bit S-boxes the search space is reduced from 2^{64} to $2^{44.25}$. As involutory functions are not persistent under affine equivalence, we then searched for all involutions in the search space and split them into the affine equivalence classes. Moreover, the differential and linear branch number also is not persistent under affine equivalence. Therefore, we further studied all $2^{44.25}$ S-boxes regarding their differential and linear branching numbers. Table 20 gives an overview of the distribution of involutions and differential/linear branch numbers for all 302 affine equivalence classes. Moreover, we provide an overview of the most important cryptographic properties that hold under affine equivalence.

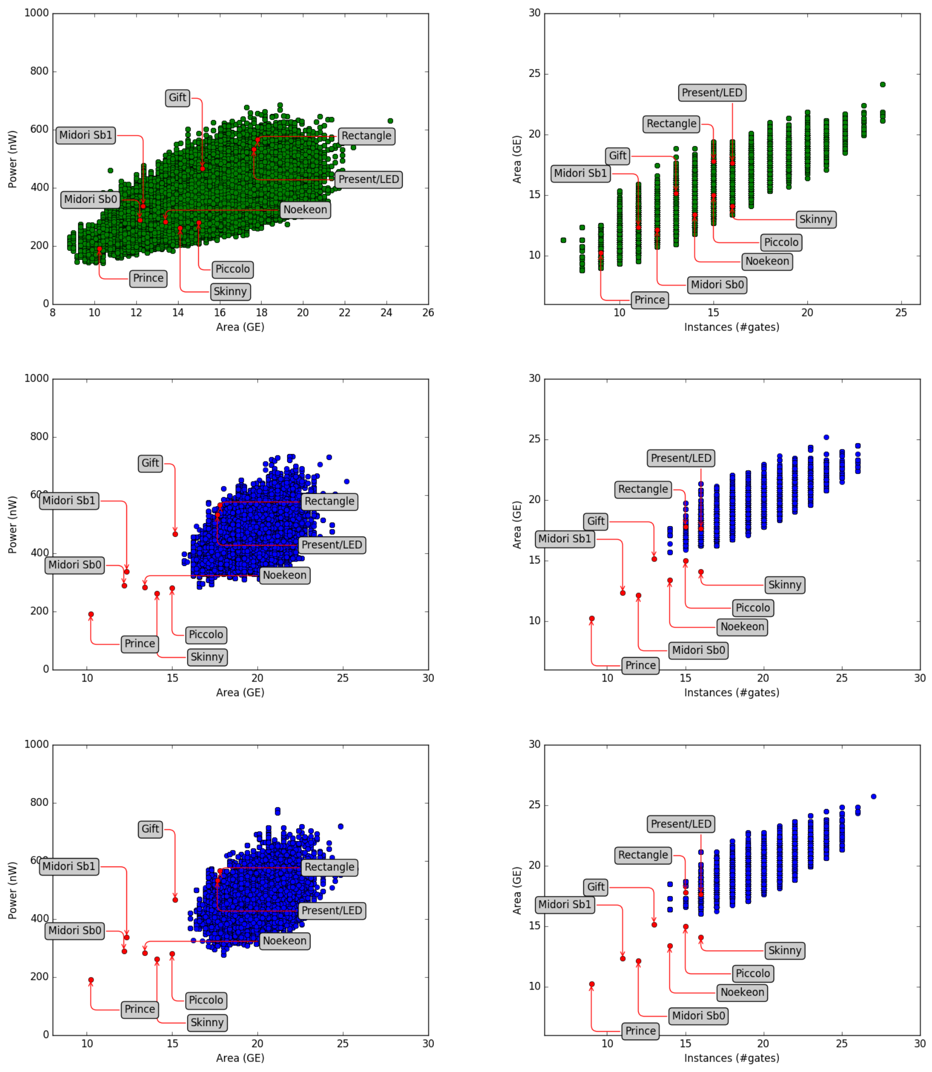


Figure 56: Green circles indicate *involutory* S-boxes and blue circles indicate S-boxes with a high *branching number*. **(Top Left):** Area/power plot of the *optimal* class 266. **(Top Right):** Instances/area plot of the *optimal* class 266. **(Middle Left):** Area/power plot of the *golden* class 209. **(Middle Right):** Instances/area plot of the *golden* class 209. **(Bottom Left):** Area/power plot of the *golden* class 210. **(Bottom Right):** Instances/area plot of the *golden* class 210.

Figure 55 shows area/power and area/instances plots of the *golden* classes 160 and 163. Figure 56 shows area/power and area/instances plots for the *golden* classes 209, 210 and the *optimal* class 266. Note that all *golden* classes are also *optimal* classes. In our Figures, we label *involutions* with green circles, and S-boxes with *high differential/linear branch number* with blue circles. Moreover, we added annotations for several lightweight S-boxes of existing ciphers such as Gift, Midori, Noekeon,

Piccolo, Present, Prince, Rectangle and Skinny. An interesting observation is that the S-box of the low-latency cipher Prince is leading the ranking, followed by the S-boxes of the energy-efficient cipher Midori. Furthermore, the involutory S-box of Noekeon is on 4th position and the simple low-area S-boxes of Skinny and Piccolo are next. Surprisingly, the S-box of Gift is just on the 7th position in the ranking, even though the designers made a huge effort to reduce the area and the number of instances as far as possible. Some further observations show that non of the *optimal* affine-equivalence classes have a linear branch number above two. Moreover for the involutory S-boxes, there are no S-boxes with a differential/linear branch number of three within all the *optimal* affine-equivalence classes. This has also been observed by Leander and Poschmann [219] who compared the *optimal* affine-equivalence classes to Serpent-like [67] S-boxes, and by Sakar and Syed [297] who studied differential/linear branch numbers of permutations. Additionally, there are also no involutions with less then two fixed points in the *optimal* affine-equivalence classes.
Nevertheless, we focused on S-boxes that are optimal in terms of area/power and area/instances. While in general we are interested in the energy consumption of an S-box, let's take briefly a deeper look into how the average power is calculated in our measurements. Let's consider a simple AND gate, then the switching power $P_{switching}$, also called dynamic power in the literature, is calculated as the power that the AND gate consumes when switching (*i.e.,* a $0 \rightarrow 1$ transition or a $1 \rightarrow 0$ transition). In more details the switching power is equal to the product of switching current and the power supply (*i.e.,* voltage).

$$P^{\mathsf{AND}}_{switching} = V_{DD} \cdot I^{\mathsf{AND}}_{switching}$$

Moreover, we can derive the switching energy by multiplying the switching power of a gate with the time it takes for that single gate to switch.

$$E^{\mathsf{AND}}_{switching} = P^{\mathsf{AND}}_{switching} \cdot t^{\mathsf{AND}}_{switching}$$

That is for a single gate but once we start building up, first the S-box and then the whole block cipher, for example, then we need to sum up all the *energy chunks* consumed by every single gate that switched during the whole encryption, basically. The number of times a single gate switches is characterised by the switching activity *sw*, which is basically a probability that the output of that gate will switch from $0 \rightarrow 1$, or vice versa. While the switching activity is normally data dependent, when evaluating a cryptographic S-box we can assume that the input and outputs are chosen uniform at random. This way we can theoretically derive the switching activity for every single gate from the full circuit description (*i.e.,* what is also called a *netlist*). However, in practice, we normally run hundreds of encryptions using random test vectors and, during those runs, these small lumps of energy of every individual gate are added up and the total energy consumed is divided with the time it takes to run all those encryptions. That is how we obtain the figure of average power consumption for our measurements.
Table 18 gives an overview of the best S-boxes we found in the *golden* and a few selected *optimal* affine-equivalence classes. Those S-boxes are optimal in case of their cryptographic properties, as well as they are optimal in case of their performance properties for the use in low-energy ciphers.

Table 18: Recommended list of S-boxes for low-energy consumption, based on their cryptographic and performance properties. The first eight S-boxes are involutory S-boxes optimised for PRINCE-like cipher, and the next seven S-boxes have a high branching number that are optimised for PRESENT-like ciphers.

		Cryptographic properties									Performance properties		
S-box	Affine Eq. Class	Algebraic Degree	Diff. Branch number	Lin. Branch number	Linearity	Linear Pro. Bias	Diff. Uniformity	Diff. Probability	Fixed Points	Involution	Area (GE)	Instances (#gates)	Power (nW/@1Mhz)
`BD2CAF6E98403175`	160	3	2	2	8	2^{-2}	4	2^{-2}	2	✓	13.25	10	148
`DBCF4A6E98512073`	160	3	2	2	8	2^{-2}	4	2^{-2}	2	✓	13.25	10	142
`E623CA1BDF574809`	160	3	2	2	8	2^{-2}	4	2^{-2}	2	✓	13.25	10	157
`EC7F6A42895D1B03`	160	3	2	2	8	2^{-2}	4	2^{-2}	2	✓	13.25	10	159
`E6AC451DBF283709`	160	3	2	2	8	2^{-2}	4	2^{-2}	2	✓	13.25	10	155
`EA6C7F24891D3B05`	160	3	2	2	8	2^{-2}	4	2^{-2}	2	✓	13.25	10	156
`94CE165780BA2F3D`	163	3	2	2	8	2^{-2}	4	2^{-2}	2	✓	13.75	9	158
`6132C90875FB4DEA`	266	3	2	2	8	2^{-2}	4	2^{-2}	4	✓	13.00	9	146
`6B8EC750D43921FA`	209	3	3	2	8	2^{-2}	4	2^{-2}	0	✗	23.50	17	285
`6D8EA730B25941FC`	209	3	3	2	8	2^{-2}	4	2^{-2}	0	✗	23.50	17	285
`5E8D9730B16A42FC`	209	3	3	2	8	2^{-2}	4	2^{-2}	0	✗	23.50	17	285
`3E8B9750D16C24FA`	209	3	3	2	8	2^{-2}	4	2^{-2}	0	✗	23.50	17	285
`A74ECB90D83521F6`	209	3	3	2	8	2^{-2}	4	2^{-2}	0	✗	23.50	17	285
`974DCBA0E83612F5`	209	3	3	2	8	2^{-2}	4	2^{-2}	0	✗	23.50	17	285
`B760C2F9D41E38A5`	210	3	3	2	8	2^{-2}	4	2^{-2}	0	✗	23.75	17	325

4.3.4 Low-energy S-boxes for PRINCE-like Designs

After identifying some low-energy S-boxes the natural question that arises is, if one replaces the S-boxes in an already existing cipher with the low-energy S-boxes, does the energy consumption of the whole cipher improve? As we already have seen in Figure 54, the area distribution of the S-box layer takes about 21.9% when the clock frequency is unconstrained. However, there are also other factors that affect the energy consumption in a block cipher as further outlined in Section 4.2.2.

Therefore, we replaced the original PRINCE S-box with the before identified low-energy S-boxes in a fully-unrolled PRINCE implementation, and studied the energy

Table 19: Comparison of fully-unrolled implementations of PRINCE with our recommended low-energy S-boxes. We highlight in bold font the S-boxes that improve one of the performance properties and in light green S-boxes that improve the whole implementation of PRINCE. The implementation of PRINCE is done using the 40nm CMOS library.

S-box	Area (GE)	Power (nW/@1MHz)	Note
BF32AC916780E5D4	**12876**	58.12	Original PRINCE S-box
BD2CAF6E98403175	**12729**	59.18	Involutory S-boxes
DBCF4A6E98512073	**12729**	59.21	
E623CA1BDF574809	**12678**	59.15	
EC7F6A42895D1B03	**12677**	58.45	
E6AC451DBF283709	**12677**	59.84	
EA6C7F24891D3B05	**12684**	**57.98**	
94CE165780BA2F3D	13028	66.43	
6132C90875FB4DEA	**12706**	62.64	
6B8EC750D43921FA	15004	72.79	S-boxes high Branch Nr.
6D8EA730B25941FC	15002	71.26	
5E8D9730B16A42FC	15004	70.89	
3E8B9750D16C24FA	15004	71.74	
A74ECB90D83521F6	15004	72.64	
974DCBA0E83612F5	15000	71.48	
B760C2F9D41E38A5	14143	79.05	

consumption of the whole cipher. Table 19 outlines our results. We highlight in bold font, the S-boxes that improve upon the PRINCE S-box when implemented in a fully-unrolled PRINCE implementation. While there are many S-boxes that potentially improve upon PRINCE, as it can also be seen in Figure 55 and Figure 56, there is just a short list of S-boxes that actually improve the performance of PRINCE when implemented in the cipher. Moreover, as it can be seen in Table 19, nearly all of our recommended S-boxes improve the total area of PRINCE, however, just one of our recommended S-boxes (highlighted in green) additionally slightly improves the power consumption.

Table 20: Cryptographic properties of all affine equivalence classes. We highlight optimal classes in blue, and golden classes in yellow. The *representative* of a class is the lexicographically smallest S-box in the class. *Deg.*, *MDBN*, *MLBN*, *Lin.*, ϵ, δ, *DP* denotes the algebraic degree, the maximum differential branch number within the class, the maximum linear branch number within the class, the linearity, the linear probability bias, the differential uniformity, and the differential probability of an S-box, respectively. *#Inv.* denotes the number of involutions per class.

	Representative	**Deg.**	**MDBN.**	**MLBN.**	**Lin.**	ϵ	δ	**DP**	**#Inv.**
0	`0123456789ABCDEF`	1	4	4	16	$2^{-1.00}$	16	1	$2^{29.03}$
1	`0123456789ABCDFE`	3	3	2	16	$2^{-1.00}$	16	1	$2^{25.82}$
2	`0123456789ABCEFD`	3	2	2	16	$2^{-1.00}$	12	$2^{-0.42}$	$2^{23.88}$
3	`0123456789ABDEFC`	3	3	2	16	$2^{-1.00}$	16	$2^{0.00}$	$2^{24.76}$
4	`0123456789ABDCFE`	2	3	3	16	$2^{-1.00}$	16	$2^{0.00}$	$2^{26.21}$
5	`0123456789ACDBFE`	3	3	2	16	$2^{-1.00}$	12	$2^{-0.42}$	$2^{22.30}$
6	`0123456789ACBDFE`	3	3	2	16	$2^{-1.00}$	12	$2^{-0.42}$	$2^{23.11}$
7	`0123456789ACBEFD`	3	2	2	16	$2^{-1.00}$	10	$2^{-0.68}$	$2^{21.88}$
8	`0123456789ACDEFB`	3	3	2	16	$2^{-1.00}$	12	$2^{-0.42}$	$2^{20.30}$
9	`0123456789ACDEBF`	3	3	2	16	$2^{-1.00}$	10	$2^{-0.68}$	$2^{22.30}$
10	`0123456789BCAEFD`	3	3	2	16	$2^{-1.00}$	10	$2^{-0.68}$	$2^{21.30}$
11	`0123456789BCEFDA`	3	3	2	16	$2^{-1.00}$	12	$2^{-0.42}$	$2^{21.30}$
12	`0123456789CDEFAB`	2	3	3	16	$2^{-1.00}$	16	$2^{0.00}$	$2^{24.11}$
13	`0123456789CDEFBA`	3	3	2	16	$2^{-1.00}$	16	$2^{0.00}$	$2^{23.88}$
14	`0123456879CDEFBA`	3	3	2	12	$2^{-1.42}$	12	$2^{-0.42}$	$2^{19.88}$
15	`012345687A9CBEFD`	3	3	2	12	$2^{-1.42}$	10	$2^{-0.68}$	$2^{19.30}$
16	`012345687A9CDFBE`	3	3	2	12	$2^{-1.42}$	10	$2^{-0.68}$	$2^{18.30}$
17	`0123456879CDEFAB`	3	3	2	16	$2^{-1.00}$	12	$2^{-0.42}$	$2^{20.30}$
18	`0123456879ACDBFE`	3	3	2	16	$2^{-1.00}$	10	$2^{-0.68}$	$2^{19.30}$
19	`0123456879ACDFBE`	3	3	2	12	$2^{-1.42}$	10	$2^{-0.68}$	$2^{18.30}$
20	`0123456879ACDEBF`	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{18.30}$
21	`0123456879ACBDFE`	3	3	2	16	$2^{-1.00}$	10	$2^{-0.68}$	$2^{20.30}$
22	`0123456879ACFEDB`	3	3	2	16	$2^{-1.00}$	10	$2^{-0.68}$	$2^{20.30}$
23	`0123456879BCEFAD`	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{18.30}$
24	`012345687A9CFBDE`	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{20.62}$
25	`0123456879ABCEFD`	3	3	2	16	$2^{-1.00}$	10	$2^{-0.68}$	$2^{20.88}$
26	`0123456879BCDEFA`	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{19.88}$
27	`012345687ABCDEF9`	3	3	2	12	$2^{-1.42}$	10	$2^{-0.68}$	$2^{18.30}$
28	`0123456879BCEAFD`	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{18.30}$
29	`012345687ABCEFD9`	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
30	`012345687ABCE9FD`	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0

continued...

	Representative	Deg.	MDBN.	MLBN.	Lin.	ϵ	δ	DP	#Inv.
31	0123456879ACBEFD	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{18.30}$
32	0123456879ACFBDE	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{18.30}$
33	0123456879BCEFDA	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
34	0123456879BCFEAD	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
35	0123456879CEAFDB	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{18.30}$
36	0123456879CEAFBD	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{18.30}$
37	0123456879ACDEFB	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{18.30}$
38	0123456879ABDEFC	3	3	2	16	$2^{-1.00}$	12	$2^{-0.42}$	$2^{21.30}$
39	012345768A9CBEFD	3	3	2	16	$2^{-1.00}$	8	$2^{-1.00}$	0
40	012345768A9CBFDE	3	3	2	16	$2^{-1.00}$	8	$2^{-1.00}$	0
41	012345768A9CBFED	3	2	2	16	$2^{-1.00}$	8	$2^{-1.00}$	$2^{21.30}$
42	012345786ACBED9F	3	2	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
43	012345786ABCF9DE	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
44	012345786AC9BFED	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
45	012345786A9CFBDE	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{20.88}$
46	012345786ABCDEF9	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{18.30}$
47	012345786AC9DEBF	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
48	012345786AC9EDFB	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
49	012345786A9CDEBF	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
50	012345786A9CFDBE	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
51	012345786ABCDE9F	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
52	012345786ACBDE9F	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
53	012345786ACBDFE9	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
54	012345786A9BCEFD	3	3	2	16	$2^{-1.00}$	8	$2^{-1.00}$	$2^{19.30}$
55	012345786AB9CFDE	3	3	2	16	$2^{-1.00}$	6	$2^{-1.42}$	$2^{19.30}$
56	012345786AC9BFDE	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
57	012345786A9CBEFD	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{18.30}$
58	012345786ACFDE9B	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
59	012345786ACEDFB9	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
60	012345786ACFB9DE	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
61	012345786ACFDEB9	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
62	012345786A9CBFED	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
63	012345786AC9DEFB	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
64	012345786ABCED9F	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
65	012345786A9CFDEB	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{18.30}$
66	012345786ACB9EFD	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0

continued...

	Representative	Deg.	MDBN.	MLBN.	Lin.	ϵ	δ	DP	#Inv.
67	012345786ACF9DBE	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
68	0123457869ACDFEB	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
69	0123457869ACDEBF	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
70	012345786ACBF9ED	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
71	012345786ACEBD9F	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
72	012345786ACDF9EB	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{18.30}$
73	012345786ACDF9BE	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
74	012345786ACDE9FB	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
75	012345786AC9FBED	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
76	012345786ACEBFD9	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
77	012345786A9CEFDB	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{20.30}$
78	0123457869ACBEDF	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{18.30}$
79	0123457869ACBFDE	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
80	0123457869ACBEFD	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
81	0123457869ACEFDB	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{19.30}$
82	0123457869ACEBDF	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
83	0123457869ACEBFD	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{18.30}$
84	012345786ACF9EBD	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
85	012345786A9CEBDF	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
86	012345786A9CFBED	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
87	012345786ACD9EFB	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
88	012345786ACD9FBE	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
89	012345786ACD9EBF	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{18.30}$
90	012345786ABCF9ED	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
91	012345786ACFBD9E	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
92	012345786ABC9EDF	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
93	012345786ABC9EFD	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
94	012345786ACED9FB	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
95	012345786A9CDFEB	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
96	012345786A9CEDFB	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
97	0123458A6BCEDF97	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
98	0123458A6BCF97ED	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
99	0123458A6BC97FDE	3	2	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
100	0123458A6B9CF7ED	3	2	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{19.30}$
101	0123458A6BCFED79	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{19.30}$
102	012345786A9CDBEF	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{19.30}$

continued...

	Representative	Deg.	MDBN.	MLBN.	Lin.	ϵ	δ	DP	#Inv.
103	0123458A69C7DFEB	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
104	0123458A69C7FDBE	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
105	0123458A697CBEFD	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
106	0123458A697CBFDE	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
107	0123458A69CE7FDB	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
108	0123458A6C9FEB7D	3	2	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
109	0123458A6CB9F7ED	3	2	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
110	0123458A69CFD7BE	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
111	0123458A69BC7FDE	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
112	0123458A6C7EBFD9	3	2	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
113	0123458A6C7FBE9D	3	2	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
114	012345786ACFBDE9	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
115	012345786ACBE9DF	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
116	0123458A6C9D7FBE	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
117	0123458A6C9D7EFB	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
118	0123458A6C9FDB7E	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
119	012345786ACB9FED	3	2	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
120	0123458A6C7EBDF9	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
121	0123458A6C7FBD9E	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
122	0123458A6BCE79FD	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
123	0123458A69BCE7DF	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
124	0123458A69CEBDF7	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
125	0123458A69CB7EFD	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
126	012345786AC9EDBF	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
127	012345786ABC9FED	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
128	0123458A6B9CDE7F	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
129	0123458A6BC7F9ED	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
130	0123458A6CBDE79F	3	2	2	8	$2^{-2.00}$	6	$2^{-1.42}$	0
131	0123458A6CE9BDF7	3	2	2	8	$2^{-2.00}$	6	$2^{-1.42}$	0
132	0123458A6CBD7E9F	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
133	0123458A6C9FBD7E	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
134	0123458A69C7DEBF	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
135	0123458A69CDE7FB	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{18.30}$
136	0123458A69C7FBED	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{18.30}$
137	0123458967CEAFBD	3	3	2	12	$2^{-1.42}$	10	$2^{-0.68}$	$2^{20.30}$
138	0123458967CEAFDB	3	3	2	12	$2^{-1.42}$	10	$2^{-0.68}$	$2^{19.30}$

continued...

	Representative	Deg.	MDBN.	MLBN.	Lin.	ϵ	δ	DP	#Inv.
139	`0123456879BCAEFD`	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{18.30}$
140	`012345687ABC9FDE`	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{18.30}$
141	`0123458967CEBFDA`	3	2	2	12	$2^{-1.42}$	10	$2^{-0.68}$	$2^{19.30}$
142	`012345786ACD9FEB`	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{18.30}$
143	`0123458A69CFB7DE`	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
144	`0123458A69CFDEB7`	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
145	`0123458A69BCF7ED`	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
146	`0123458A69CB7FDE`	3	2	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
147	`012345786ABCFDE9`	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
148	`012345786ABCE9FD`	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
149	`012345786ABCFD9E`	3	3	2	12	$2^{-1.42}$	10	$2^{-0.68}$	$2^{19.30}$
150	`0123458A6BCFDE97`	3	3	2	8	$2^{-2.00}$	6	$2^{-1.42}$	0
151	`0123458A6BCF97DE`	3	3	2	8	$2^{-2.00}$	6	$2^{-1.42}$	0
152	`0123458A6BCF7E9D`	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
153	`0123458A6B9CEDF7`	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
154	`0123467859CFBEAD`	3	3	2	12	$2^{-1.42}$	4	$2^{-2.00}$	0
155	`0123467859CFEBDA`	3	3	2	12	$2^{-1.42}$	4	$2^{-2.00}$	0
156	`0123458A69CFE7BD`	3	2	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
157	`0123458A69CEFB7D`	3	2	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
158	`0123458A6BCF7D9E`	3	2	2	8	$2^{-2.00}$	6	$2^{-1.42}$	0
159	`0123458A6BCED79F`	3	2	2	8	$2^{-2.00}$	6	$2^{-1.42}$	0
160	`0123468B59CED7AF`	3	3	2	8	$2^{-2.00}$	4	$2^{-2.00}$	$2^{18.30}$
161	`0123458A6B7CEDF9`	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
162	`0123458A6B7CDFE9`	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
163	`0123468C59BDE7AF`	3	3	2	8	$2^{-2.00}$	4	$2^{-2.00}$	$2^{18.30}$
164	`0123458A6B7C9FDE`	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
165	`0123458A6B7C9EFD`	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
166	`012345896ABCE7DF`	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
167	`0123458A67BC9EFD`	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
168	`0123458A6CBFE7D9`	3	3	2	8	$2^{-2.00}$	6	$2^{-1.42}$	$2^{19.30}$
169	`012345786ACFB9ED`	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
170	`012345786ACEB9DF`	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
171	`0123458A6CBF7E9D`	3	3	2	8	$2^{-2.00}$	6	$2^{-1.42}$	0
172	`0123458A6C9DBF7E`	3	3	2	8	$2^{-2.00}$	6	$2^{-1.42}$	0
173	`012345786A9CBDFE`	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{19.30}$
174	`0123458A69CF7EBD`	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$

continued...

	Representative	Deg.	MDBN.	MLBN.	Lin.	ϵ	δ	DP	#Inv.
175	012345786ACDE9BF	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{18.30}$
176	0123457869ACFEBD	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{19.30}$
177	0123457869BCEAFD	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{18.30}$
178	0123458A6C7DBFE9	3	2	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
179	012345786A9CEDBF	3	2	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{19.30}$
180	0123458A6C9D7FEB	3	2	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
181	012345896ABC7FDE	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
182	0123458A67BC9FDE	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
183	012345896ACF7BED	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
184	0123458A67CF9BED	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
185	012345896ACE7BFD	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
186	0123458A67CF9BDE	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
187	012345786ACEFB9D	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
188	012345786ACFEB9D	3	2	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
189	0123457869CEFBDA	3	2	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{19.88}$
190	0123458A6C7DBEF9	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
191	0123458A6C7FB9DE	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
192	0123458A6C7FBED9	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{19.30}$
193	0123458A6C7FDB9E	3	2	2	8	$2^{-2.00}$	6	$2^{-1.42}$	$2^{20.30}$
194	012345786ACFED9B	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
195	0123458A6BC7DE9F	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
196	0123468C59BDEA7F	3	2	2	8	$2^{-2.00}$	4	$2^{-2.00}$	$2^{20.62}$
197	0123458A6CBDE97F	3	2	2	8	$2^{-2.00}$	6	$2^{-1.42}$	$2^{18.30}$
198	0123458A69C7BEFD	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
199	0123458A6BCFD9E7	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
200	0123458A6BCFD79E	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
201	012345786ACB9FDE	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
202	012345786ACE9DFB	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
203	012345786ACF9BDE	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
204	012345786ACE9BFD	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
205	012345786ACDB9EF	3	3	2	12	$2^{-1.42}$	10	$2^{-0.68}$	$2^{19.30}$
206	012345896ABCEDF7	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
207	0123458A67BCEDF9	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
208	0123458A69C7BFDE	3	2	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
209	0123468B59CF7DAE	3	3	2	8	$2^{-2.00}$	4	$2^{-2.00}$	0
210	0123468A5BCF7D9E	3	3	2	8	$2^{-2.00}$	4	$2^{-2.00}$	0

continued...

	Representative	Deg.	MDBN.	MLBN.	Lin.	ϵ	δ	DP	#Inv.
211	0123458A69CED7FB	3	2	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
212	0123458A69BC7EFD	3	2	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
213	012345896ABC7EFD	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{19.30}$
214	0123458A67CEB9FD	3	3	2	8	$2^{-2.00}$	8	$2^{-1.00}$	0
215	012345896ACEB7FD	3	3	2	8	$2^{-2.00}$	8	$2^{-1.00}$	0
216	0123457869CDEFBA	3	3	2	12	$2^{-1.42}$	12	$2^{-0.42}$	$2^{20.30}$
217	012345687ABC9EFD	3	3	2	12	$2^{-1.42}$	10	$2^{-0.68}$	$2^{18.30}$
218	0123457869BCDEFA	3	3	2	12	$2^{-1.42}$	10	$2^{-0.68}$	$2^{19.88}$
219	012345786ACF9BED	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
220	0123468A59CFDE7B	3	3	2	12	$2^{-1.42}$	4	$2^{-2.00}$	$2^{19.30}$
221	0123457869CEAFDB	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
222	0123467859CFEADB	3	2	2	12	$2^{-1.42}$	4	$2^{-2.00}$	$2^{19.30}$
223	0123468A5BCFDE79	3	2	2	8	$2^{-2.00}$	4	$2^{-2.00}$	$2^{19.30}$
224	0123457869CEBFDA	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
225	0123456879CEBFDA	3	2	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{18.30}$
226	012345786ABC9FDE	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
227	012345786ACFD9BE	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
228	0123458A69BCEDF7	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{18.30}$
229	0123458A6C9DBFE7	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
230	0123458A6CEB7FD9	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	0
231	0123468B59CEDA7F	3	2	2	8	$2^{-2.00}$	4	$2^{-2.00}$	$2^{18.30}$
232	0123458A6C9FDBE7	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{19.30}$
233	0123458A67B9CFDE	3	3	2	16	$2^{-1.00}$	8	$2^{-1.00}$	0
234	012345896AB7CFDE	3	3	2	16	$2^{-1.00}$	8	$2^{-1.00}$	0
235	0123458A69B7CEFD	3	2	2	16	$2^{-1.00}$	6	$2^{-1.42}$	$2^{19.30}$
236	0123458A6B97CFDE	3	3	2	16	$2^{-1.00}$	8	$2^{-1.00}$	$2^{20.88}$
237	0123458A69B7CFDE	3	3	2	16	$2^{-1.00}$	8	$2^{-1.00}$	$2^{20.30}$
238	0123457689CEAFBD	3	2	2	16	$2^{-1.00}$	10	$2^{-0.68}$	$2^{21.30}$
239	0123457689CEAFDB	3	3	2	16	$2^{-1.00}$	10	$2^{-0.68}$	$2^{21.30}$
240	012345768A9CDEFB	3	2	2	16	$2^{-1.00}$	8	$2^{-1.00}$	$2^{20.30}$
241	012345768A9CDEBF	3	3	2	16	$2^{-1.00}$	8	$2^{-1.00}$	0
242	012345768A9CDFEB	3	3	2	16	$2^{-1.00}$	8	$2^{-1.00}$	0
243	012345768ACF9BDE	3	3	2	16	$2^{-1.00}$	8	$2^{-1.00}$	0
244	012345768ACE9BFD	3	3	2	16	$2^{-1.00}$	8	$2^{-1.00}$	0
245	012345768ACF9BED	3	2	2	16	$2^{-1.00}$	8	$2^{-1.00}$	$2^{21.30}$
246	0123456879BAEFDC	3	3	2	16	$2^{-1.00}$	12	$2^{-0.42}$	$2^{20.30}$

continued...

	Representative	Deg.	MDBN.	MLBN.	Lin.	ϵ	δ	DP	#Inv.
247	012345687AB9DEFC	3	2	2	16	$2^{-1.00}$	10	$2^{-0.68}$	$2^{20.88}$
248	0123456879CEFBDA	3	3	2	16	$2^{-1.00}$	8	$2^{-1.00}$	$2^{19.88}$
249	0123458A69CFEB7D	3	2	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
250	0123458A69CD7FEB	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{19.30}$
251	0123458A69CEF7DB	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
252	0123458A69CEFBD7	3	2	2	16	$2^{-1.00}$	6	$2^{-1.42}$	$2^{21.11}$
253	0123458A69CE7FBD	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
254	0123458A69BCFD7E	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
255	012345786ABCEDF9	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
256	012345896ACF7BDE	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
257	012345896ABCFD7E	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	0
258	012345896ACE7BDF	3	3	2	16	$2^{-1.00}$	8	$2^{-1.00}$	$2^{20.30}$
259	012345896ACEFDB7	3	3	2	16	$2^{-1.00}$	8	$2^{-1.00}$	0
260	012345896AB7CEFD	3	3	2	16	$2^{-1.00}$	8	$2^{-1.00}$	0
261	0123458A69CEB7FD	3	2	2	8	$2^{-2.00}$	6	$2^{-1.42}$	$2^{18.30}$
262	0123458A6C7DB9FE	3	2	2	8	$2^{-2.00}$	8	$2^{-1.00}$	$2^{20.30}$
263	0123458A6BC7EDF9	3	2	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{19.30}$
264	0123458A6C7DFEB9	3	3	2	16	$2^{-1.00}$	8	$2^{-1.00}$	$2^{20.88}$
265	0123458A6BCDE9F7	3	3	2	12	$2^{-1.42}$	8	$2^{-1.00}$	$2^{20.30}$
266	0123468A5BCFED97	3	3	2	8	$2^{-2.00}$	4	$2^{-2.00}$	$2^{19.30}$
267	012345786ABCE9DF	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
268	0123458A69CFBED7	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
269	0123458A69CEBFD7	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
270	0123468B5C9DEA7F	3	2	2	8	$2^{-2.00}$	4	$2^{-2.00}$	$2^{20.30}$
271	0123468B5C9DAFE7	3	2	2	12	$2^{-1.42}$	4	$2^{-2.00}$	$2^{21.11}$
272	0123468B5CD79FAE	3	2	2	8	$2^{-2.00}$	4	$2^{-2.00}$	$2^{19.30}$
273	0123458A6C7FEB9D	3	2	2	8	$2^{-2.00}$	6	$2^{-1.42}$	$2^{19.30}$
274	0123458A6BCED97F	3	2	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
275	0123458A6CF7BE9D	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{19.30}$
276	0123458A6CF7BD9E	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{19.30}$
277	0123458A6BC9DE7F	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
278	0123468B5CD7AF9E	3	2	2	8	$2^{-2.00}$	4	$2^{-2.00}$	$2^{21.30}$
279	0123458A6BC7DFE9	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{20.30}$
280	0123457869ACEDBF	3	2	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{21.88}$
281	0123457869ACFBDE	3	3	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{19.30}$
282	0123468B5CD7F9EA	3	2	2	8	$2^{-2.00}$	4	$2^{-2.00}$	$2^{22.47}$

continued...

	Representative	Deg.	MDBN.	MLBN.	Lin.	ϵ	δ	DP	#Inv.
283	0123468B5C9DE7AF	3	2	2	8	$2^{-2.00}$	4	$2^{-2.00}$	$2^{20.62}$
284	0123458A6BCF9D7E	3	3	2	8	$2^{-2.00}$	6	$2^{-1.42}$	$2^{18.30}$
285	0123457869CEAFBD	3	2	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
286	0123458967CEFBDA	3	2	2	16	$2^{-1.00}$	10	$2^{-0.68}$	$2^{20.30}$
287	012345768A9CDFBE	3	3	2	16	$2^{-1.00}$	8	$2^{-1.00}$	$2^{20.88}$
288	0123456789CEFBDA	3	2	2	16	$2^{-1.00}$	10	$2^{-0.68}$	$2^{23.76}$
289	0123456789CEBFDA	3	2	2	16	$2^{-1.00}$	10	$2^{-0.68}$	$2^{22.30}$
290	0123456789BCEAFD	3	3	2	16	$2^{-1.00}$	10	$2^{-0.68}$	$2^{22.62}$
291	012345768A9BCFED	3	2	2	16	$2^{-1.00}$	8	$2^{-1.00}$	$2^{23.30}$
292	012345768A9BCEFD	3	3	2	16	$2^{-1.00}$	8	$2^{-1.00}$	$2^{21.30}$
293	0123457689CDEFBA	2	3	3	16	$2^{-1.00}$	16	$2^{0.00}$	$2^{24.11}$
294	0123456789BAEFDC	2	3	3	16	$2^{-1.00}$	16	$2^{0.00}$	$2^{24.30}$
295	0123468C59DFA7BE	3	2	2	8	$2^{-2.00}$	4	$2^{-2.00}$	$2^{18.30}$
296	0123468A5BCF7E9D	3	3	2	8	$2^{-2.00}$	4	$2^{-2.00}$	0
297	0123468A5BCF79DE	3	3	2	8	$2^{-2.00}$	4	$2^{-2.00}$	0
298	012345687ACEB9FD	3	2	2	12	$2^{-1.42}$	6	$2^{-1.42}$	$2^{18.30}$
299	012345678ACEB9FD	2	2	3	16	$2^{-1.00}$	8	$2^{-1.00}$	$2^{21.30}$
300	0123458967CDEFAB	2	3	3	16	$2^{-1.00}$	16	$2^{0.00}$	$2^{24.11}$
301	0123458967CDEFBA	3	3	2	12	$2^{-1.42}$	16	$2^{0.00}$	$2^{24.11}$

We provide all involutory S-boxes and S-boxes with a differential/linear branching number of > 3 in a git repository to encourage further research in this topic. The S-boxes can be obtained at: `https://github.com/TheBananaMan/low-energy-sboxes`.

4.4 CONCLUSION

In this chapter, we study cryptographic S-boxes with a focus on low-energy consumption. We give detailed design considerations for low-energy block ciphers in general, and further provide recommendations for S-boxes with low-energy consumption. After identifying Present and Prince as two optimal block cipher design strategies, for low-energy consumption, we define requirements for optimal low-energy S-boxes. In a Present-like design, the bit-based diffusion layer does not require a lot of overhead and therefore, the main contribution to the energy consumption is based on the S-box layer. We show however, that for a secure cipher, the differential/linear branch number of the S-box has to be high for this type of block cipher. On the other hand, in a Prince-like design, one can make use of involutory S-boxes. We show that there exist some S-boxes that are better then the currently

used S-boxes, and show that the overall energy-consumption decreases when we replace the original S-boxes with the ones we recommend.
However, we further noted that the S-box layer in a cipher just contributes partly to the overall energy consumption and we want to highlight that in an optimal low-energy block cipher design also the diffusion layer should be optimised for low-energy. While bit-based diffusion layers are optimal as they can be implemented in hardware just by wiring, they add several additional requirements to the S-box layer. As a possible future work, one might look at other ways to construct efficient diffusion layers, such as binary diffusion matrices (*i.e.,* as used in MIDORI [37], SKINNY [50] and QARMA [34]) or one even consider non-linear diffusion layers as suggested by Liu *et al.* [231].

Part II

Cryptanalysis

5 ZERO-CORRELATION ATTACKS ON TWEAKABLE BLOCK CIPHERS WITH A LINEAR TWEAK SCHEDULE

CONTENTS

executive summary. In this chapter, we present zero-correlation attacks on tweakable block ciphers with a linear tweak schedule, based on [23].
The design and analysis of dedicated tweakable block ciphers is a quite recent and very active research field that provides an ongoing stream of new insights. For instance, results of Kranz, Leander, and Wiemer from FSE'17 show that the addition of a tweak using a linear tweak schedule does not introduce new linear characteristics. We consider for the first time—to the best of our knowledge— the effect of the tweak on zero-correlation linear cryptanalysis. It turns out that the tweak can be used to get zero-correlation linear hulls covering more rounds, which also implies the existence of integral distinguishers on the same number of rounds. The so obtained integral distinguishers cover more rounds compared to existing ones that have been found using the division property, for the tweakable block ciphers Qarma, Mantis, and Skinny. In particular, this leads to the best attack (with respect to number of rounds) on a round-reduced variant of Qarma.

declaration of authorship. The work described in this chapter is based on the paper [23]: *Zero-Correlation Attacks on Tweakable Block Ciphers* and is currently under submission at *The 26th Fast Software Encryption conference (FSE'19)* in Paris, France. The paper is joint work with Christoph Dobraunig, Jian Guo, Eran Lambooij, Gregor Leander and Yosuke Todo. The work was initiated during a research visit of the author to Gregor Leander at Ruhr-Universität Bochum, Germany. All authors contributed equally to the results of the paper. The contributions of the author are the following:

- Discovering that the tweak schedule in tweakable block ciphers can be used to add additional constraints for zero-correlation attacks.
- Application of the attack to Mantis.
- Application of the attack to Skinny.

5.1 INTRODUCTION

Tweakable block ciphers are constructions, which have—compared to traditional block ciphers—an additional input called *tweak*. Ideally, each different choice of the tweak produces a different instance of a block cipher. This concept has first been introduced by Schroeppel in the Hasty pudding cipher [304] and was formally treated by Liskov, Rivest and Wagner [227, 228]. The concept of tweakable block ciphers allows for very clean modes of operations for authenticated encryption like: ΘCB3 [211], or Counter in Tweak [277]. When using such a mode, one faces two choices, either use a construction that takes an ordinary block cipher as building block to build a tweakable block cipher [211, 216, 245, 342],or use a dedicated tweakable block cipher [34, 50, 188].
One can expect that designing a tweakable block cipher from scratch results in more efficient designs than reusing a block cipher to create a tweakable block cipher. However, when designing dedicated tweakable block ciphers, it has to be kept in mind that the tweak is an additional publicly known input, which can potentially be influenced by an attacker. This leads to a new challenge in the analysis of such schemes, since in the chosen plaintext model, the extra input provides additional freedom for the attacker. This freedom can be exploited in attacks. The most self-evident attack vector that is influenced by the tweak is differential cryptanalysis [73]. By introducing differences in the tweak, the attacker is able to introduce differences in-between rounds, which typically leads to longer differential characteristics. Naturally, this increases the number of rounds the attacker can attack in a key-recovery attack.
Besides this, there is a constant evaluation of known attack vectors on tweakable block ciphers that exploit using the tweak. There are for example: Boomerang attacks [113, 141], meet-in-the-middle attacks [327], impossible differential attacks [141, 298] and integral attacks [139]. A positive result with respect to the security of tweakable block ciphers is that the addition of a tweak, using a linear tweak schedule, does not require additional considerations with respect to linear cryptanalysis [208].
Attacks on dedicated tweakable block ciphers exploit the additional freedom introduced by the tweak to extend a distinguisher in the data-path of a cipher. In this work, we follow this general idea to derive distinguishers on the data-path plus tweak schedule which can be turned into better attacks. In particular, we exploit zero-correlation linear hulls [97, 99] on the data-path plus tweak. The fact that a lot

of state-of-the-art tweakable block cipher constructions not only use a tweak schedule that is linear, but also have very limited diffusion in the tweak bits comes in handy. This allows us to search for zero-correlation linear hulls with the help of a miss-in-the-middle approach. In our attacks the miss (*i.e.,* contradiction) occurs within the tweak schedule.
These zero-correlation linear hulls typically cover more rounds than zero-correlation linear hulls that only consider the data-path. Next to that, the relation between zero-correlation and integral distinguishers [96, 315] allows us to get an integral property (zero sum) in the data path. This property can then be exploited in key-recovery attacks.
We first examine the effects of zero-correlation linear cryptanalysis on tweakable block ciphers having a linear key schedule. We focus on the implications on tweakable block ciphers following the *Superposition* Tweakey (STK) constructions. After that we give examples for zero-correlation linear hulls for three dedicated tweakable block ciphers Qarma [34], Mantis [50] and Skinny [50]. As shown in Table 21, the acquired distinguishers cover more rounds compared to existing results utilising the division property, zero-correlation, or conventional approaches to search integral distinguisher. In the case of round-reduced Qarma [34], these new distinguishers allow for attacks covering more rounds than previous ones.
Note that some of the attacks shown in Table 21 require more than 2^n data for an n-bit block size. In contrast to standard block ciphers where 2^n is the natural limit per key (i.e. the full-codebook is reached), tweakable block ciphers allow to gather the amount of 2^n data per tweak and hence, a total of 2^{n+t} data can be collected considering a t-bit tweak. Our attacks on Skinny require data above 2^n, but we do not collect the full-codebook under one fixed tweakey. Hence, we can recover unknown tweakey-information that has not been queried in our key-recovery attacks.
Apart from the dedicated attacks, this new way of searching for integral distinguishers provides further insights in the design of tweakable block ciphers. One of the new insights is a better intuition on how the number of positions and the locations of the tweak addition influences the security of a tweakable block cipher. For instance, consider the case of a tweakable block cipher where the addition of the tweak is just performed for a few rounds at the beginning and the end of the cipher, while for the rounds in the middle just the round keys are added. Such a construction can lead to the unfortunate situation, that the zero-correlation linear hulls are independent of the number of computed keyed middle-rounds.

RELATED WORK. The conversion [315] of zero-correlation linear hulls to what is commonly referred to as integral distinguishers is not the only method to find such distinguishers. A common approach to find integral distinguishers is to exploit knowledge about upper bounds on the algebraic degree of a function as shown by higher-order differential cryptanalysis [213]. Later on, methods that exploit the structure of a cipher in a more direct manner have been introduced in an attack on the block cipher Square [118] which became known under the name integral cryptanalysis [205]. Moreover, the division property [322] and bit-based division

Table 21: Overview on previous and proposed key-recovery attacks on variants of Qarma-64, Mantis, Skinny-64/128 , and Skinny-64/192. MITM/ID/ZC/Inv. = Meet-in-the-Middle/Impossible Differentials/Zero-Correlation/Invariants

Cipher	Rounds	Attack type	Time	Data	Memory	Ref.
Qarma-64	4/4*	MITM	2^{90}	2^{16}	2^{90}	[225]
Qarma-64	4/5*	MITM	2^{89}	2^{16}	2^{89}	[225]
Qarma-64	4/6*	MITM	$2^{70.1}$	2^{53}	2^{116}	[363]
Qarma-64	3/8*	ID	$2^{64.4}$	2^{61}	-	[364]
Qarma-64	4/7*	ID	$2^{120.4}$	2^{61}	2^{116}	[355]
Qarma-64	4/8*	ZC/Integral	$2^{66.2}$	$2^{48.4}$	$2^{53.70}$	This Work
Mantis	5/5*	Inv.	2^{56}	$2^{9.3}$	-	[65]
Mantis	6/6*	Diff.	2^{38}	2^{28}	-	[137]
Mantis	7/7*	Diff.	$2^{53.94}$	$2^{53.94}$	-	[151]
Mantis	4/8*	ZC/Integral	$2^{66.2}$	$2^{48.4}$	$2^{53.70}$	This Work
Skinny-64/128	18	ZC	2^{126}	$2^{62.68}$	2^{64}	[295]
Skinny-64/128	19	ID	$2^{119.8}$	2^{62}	2^{110}	[356]
Skinny-64/128	20	ID	$2^{121.08}$	$2^{47.69}$	$2^{47.69}$	[328]
Skinny-64/128	20	ZC/Integral	$2^{97.5}$	$2^{68.4\dagger}$	2^{82}	This Work
Skinny-64/128	23	ID	2^{124}	$2^{62.47}$	$2^{77.47}$	[295]
Skinny-64/128	23	ID	$2^{125.9}$	$2^{62.5}$	$2^{124.0}$	[230]
Skinny-64/128	23	ID	2^{79}	$2^{71.4\dagger}$	$2^{64.0}$	[20]
Skinny-64/192	21	ID	$2^{180.5}$	2^{62}	2^{170}	[356]
Skinny-64/192	22	ID	$2^{183.97}$	$2^{47.84}$	$2^{74.84}$	[328]
Skinny-64/192	23	ZC/Integral	$2^{155.6}$	$2^{73.2\dagger}$	2^{138}	This Work
Skinny-64/192	27	Rectangle	$2^{165.5}$	$2^{63.5}$	2^{80}	[230]

property [326] provide a powerful improvement in the search for integral distinguishers that leads to attacks on full Misty-1 [321, 323].

It is worth mentioning that Table 21 just shows key-recovery attacks and thus, does not represent a complete list of results that provide insight into the security of Qarma, Mantis and Skinny. For instance Leander, Tezcan, and Wiemer [221] provide results regarding the length of subspace trails for various ciphers including Qarma and Skinny. Furthermore, Cid et al. [114] use their new tool called *Boomerang Connectivity Table* to re-evaluate existing related-tweakey boomerang character-

* We state the number of S-box layers in the inbound/outbound phase of the cipher.

† The attack requires more than 2^n data, where n is the block size. The full-codebook in a tweakable block cipher is exceeded by using more than 2^{n+t} data, considering a t-bit tweak.

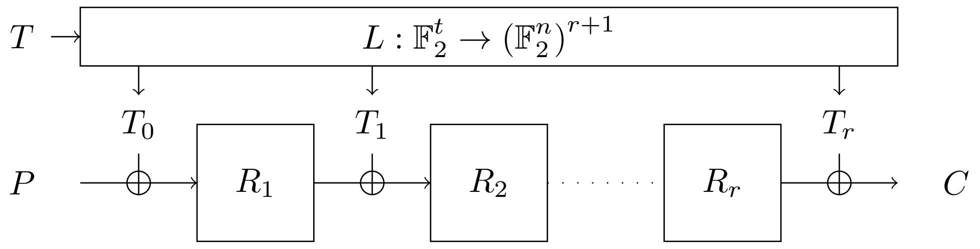


Figure 57: Key-alternating tweakable block cipher with linear tweak schedule.

istics of Skinny. Further works give more insight into the security of Skinny against differential cryptanalysis [25] and impossible differential cryptanalysis [299] and the security of Skinny and Mantis against invariant attacks [48]. Eskandari et al. [152] search for integral distinguishers based on the division property for Qarma-64, Mantis, and Skinny-64. Furthermore, Zhang and Rijmen [362] give integral distinguishers for 10 rounds of Skinny-64 based on the division property.
The property of linear hulls under the related-key setting was also discussed by Bogdanov et al. in [92]. They showed that there exist linear hulls such that their bias are invariant under key difference. More concretely, when some bits in the secret-key must be inactive of a given linear hull, then there exists another linear hull with the same correlation, where the key difference is induced into the inactive bits. In comparison to our work, we review this property from zero-correlation linear hulls. By considering zero-correlation linear hulls, we can construct non-trivial distinguishers even if all bits in the secret-key/tweak are active. Therefore, our attacks are less restricted and improve over the results of Bogdanov et al. [92].

5.2 PRELIMINARIES

5.2.1 Tweakable Block Cipher and the Tweakey Framework

Let the block and key lengths be n and κ bits, respectively. Then a conventional block cipher is defined as a function from $\mathbb{F}_2^n \times \mathbb{F}_2^\kappa \to \mathbb{F}_2^n$. A tweakable block cipher can accept an additional input called the *tweak* and it is defined as a function from $\mathbb{F}_2^n \times \mathbb{F}_2^\kappa \times \mathbb{F}_2^t \to \mathbb{F}_2^n$ when the tweak length is t bits. A more detailed explanation of tweakable block ciphers can be found in Section 2.3.1.
Throughout this chapter, we consider the case of a tweakable round based block cipher with a linear tweak-scheduling $L : \mathbb{F}_2^t \to (\mathbb{F}_2^n)^{r+1}$ mapping the (master)-tweak to the sub-tweaks, as outlined in Figure 57. Those sub-tweaks are then combined with the current state of the cipher using the XOR operation.
The Tweakey framework [188], as illustrated in Figure 58, is often used to design dedicated tweakable block ciphers, where the key and tweak are basically treated as one object called *tweakey*. Moreover, each sub-tweakey is generated by applying the same permutation recursively. Based on this framework, there are several dedicated tweakable block ciphers such as Kiasu-BC [187], Deoxys [189], Joltik [186] and Skinny [50]. A class of tweakable block cipher denoted by TK-p is introduced

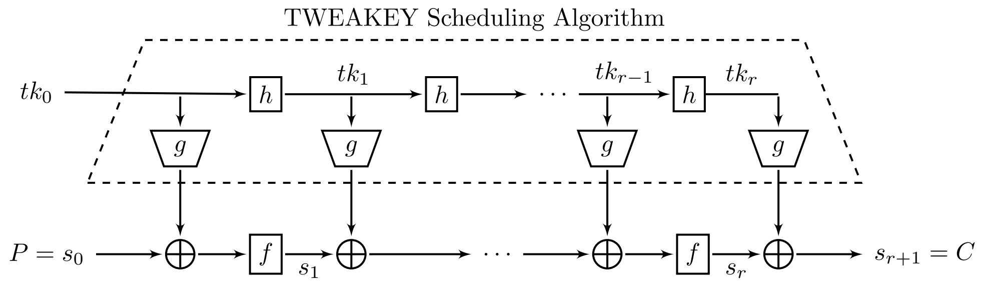


Figure 58: The Tweakey framework.

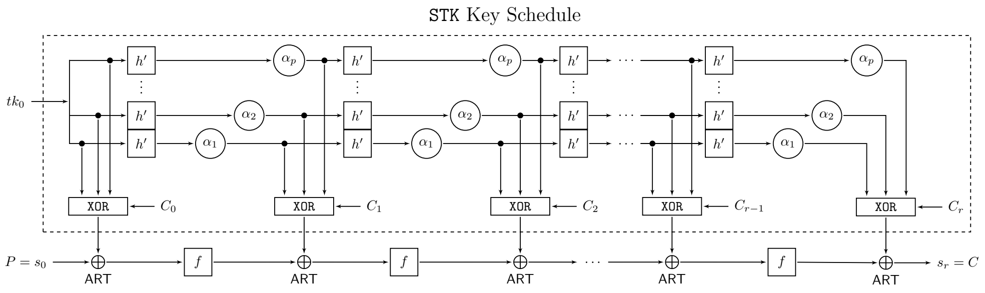


Figure 59: STK construction: Example with TK-p.

when the size of the tweakey is $(p \times n)$ bits. Then, TK-1 is suited to the simple single-key block cipher with an n-bit key, and TK-2 is suited to the tweakable block cipher with an n-bit key and an n-bit tweak. Jean *et al.* [188] gave practical subclass of the Tweakey framework named Superposition Tweakey (STK), and Fig. 59 shows the construction with TK-p. In the STK construction, the internal state and tweakey state are partitioned into n/c and pn/c c-bit nibbles, respectively. A more detailed description of the Tweakey framework can be found in Section 2.3.1.
We can notice that the tweakey scheduling algorithm of the STK construction is fully linear. Since the tweak is usually publicly accessible, attackers can naturally execute related-tweak attacks. Therefore, we need to discuss the security against related-tweakey attacks more carefully.

5.2.2 Differential Propagation through Tweakable Block Ciphers

The XOR operation is used to mix the sub-tweakey and internal state in the tweakable block ciphers discussed in this chapter. Then, a difference of an internal state can be cancelled by XORing the same difference of a sub-tweakey, and one round function is passed for free (*i.e.,* see Fig. 60). Considering related-tweak setting, we can control differences of certain internal states and acquire the characteristic with higher probability. The probability of the related-tweak/(twea)key differential characteristic is generally higher than that of the single-key characteristic. Therefore, such attacks have been well discussed in the context of both related-key attacks on the block cipher and related-tweakey attacks on the tweakable block ciphers.

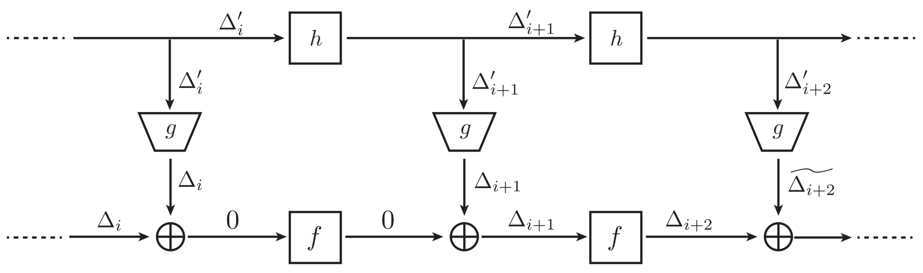


Figure 60: Propagation of differences in a tweakable block cipher.

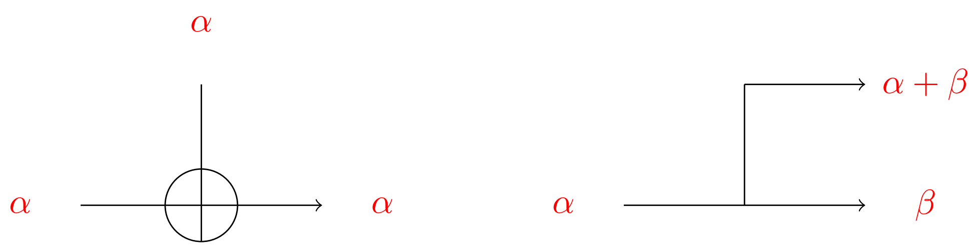


Figure 61: Left: Propagation of linear masks through XOR. **Right:** Propagation of linear masks through a branching point.

5.2.3 Linear Propagation through Tweakable Block Ciphers

Linear cryptanalysis makes use of correlations between linear combinations between input and output bits of a block cipher. More specifically, given a function

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

an input mask $\alpha \in \mathbb{F}_2^n$, and an output mask $\beta \in \mathbb{F}_2^m$ we consider

$$\mathrm{cor}_F(\alpha, \beta) := 2 \cdot \mathrm{Prob}\left(\langle \alpha, x \rangle + \langle \beta, F(x) \rangle = 0\right) - 1,$$

where the probability is taken over uniformly distributed inputs x. Traditionally, a high correlation is used as a distinguisher and then extended to a key-recovery attack [238]. Moreover, we like to mention that for the understanding of our attacks, it might be helpful to have two special cases for the propagation of linear masks in mind. These are how linear masks propagate through an *XOR-operation* and a *branching-point* as illustrated in Figure 61. In the formula, for the *XOR-operation*

$$\begin{array}{rcl} \mathbb{F}_2^n \times \mathbb{F}_2^n & \to & \mathbb{F}_2^n \\ X(x, y) & = & x + y \end{array}$$

it holds that

$$\mathrm{cor}_X\left(((\alpha_1, \alpha_2), \beta)\right) \neq 0 \text{ iff } \alpha_1 = \alpha_2 = \beta,$$

and for the *branching point*

$$\begin{array}{rcl} \mathbb{F}_2^n \times \mathbb{F}_2^n & \to & \mathbb{F}_2^n \\ B(x) & = & (x, x) \end{array}$$

it holds that

$$\mathrm{cor}_B\left(\alpha, ((\beta_1, \beta_2))\right) \neq 0 \text{ iff } \alpha + \beta_1 + \beta_2 = 0.$$

Zero-Correlation Linear Cryptanalysis

Zero-correlation linear cryptanalysis was introduced by Bogdanov and Rijmen [97] in 2014. Let α and β be the linear mask for a plaintext and ciphertext, respectively, it exploits the pair (α, β) with correlation exactly zero. One clear drawback of the basic zero-correlation linear cryptanalysis is its huge data complexity. In order to detect that the correlation is exactly zero, it is a priori necessary to encrypt (almost) every possible message. Later, the data complexity was reduced by exploiting multiple or multidimensional zero-correlation linear approximations [96, 99]. When there are ℓ zero-correlation linear approximations for an n-bit block cipher, the required data complexity is roughly estimated as $\mathcal{O}(2^n/\sqrt{\ell})$. Another option to reduce the data complexity is to exploit a link between different cryptanalysis methods and map the zero-correlation distinguisher to another (different) type of cryptanalysis with a lower requirement of data-complexity.
Several mathematical links among different types of cryptanalysis have been discussed, and here we focus on the link between zero-correlation linear cryptanalysis and integral cryptanalysis [315].

Theorem 1 ([315])**.** *Let $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be a function, and A be a subspace of $\mathbb{F}_2^n$ and $\beta \in \mathbb{F}_2^n \setminus \{0\}$. Suppose that (α, β) is a zero correlation linear approximation for any $\alpha \in A$, then for any $\lambda \in \mathbb{F}_2^n$, $\langle \beta, F(x + \lambda) \rangle$ is balanced on $A^\perp = \{x \in \mathbb{F}_2^n | \langle \alpha, x \rangle = 0, \alpha \in A\}$.*

In other words, when there are zero-correlation linear hulls, it implies an integral distinguisher. The required number of texts is 2^{n-m}, where m denotes the dimension of the subspace A. The key-recovery of the integral attack is often more efficient than the key-recovery of the zero-correlation linear cryptanalysis. Therefore, when the key-recovery is taken into consideration, we convert the zero-correlation linear hulls into integral distinguisher.
Recall the zero-correlation linear hull on 4-round AES (*i.e.,* see Figure 62). The distinguisher can be converted into the integral distinguisher with 2^{32} texts, which is the exactly same as the well-known integral distinguisher of the 4-round AES [118, 205].

5.3 ZERO-CORRELATION LINEAR CRYPTANALYSIS FOR TWEAKABLE BLOCK CIPHERS

In the case of a tweakable block cipher

$$E_k : \mathbb{F}_2^n \times \mathbb{F}_2^t \to \mathbb{F}_2^n,$$

we consider the tweak to be an additional input from which we can include the tweak bits into the linear combination of input bits, when considering linear ap-

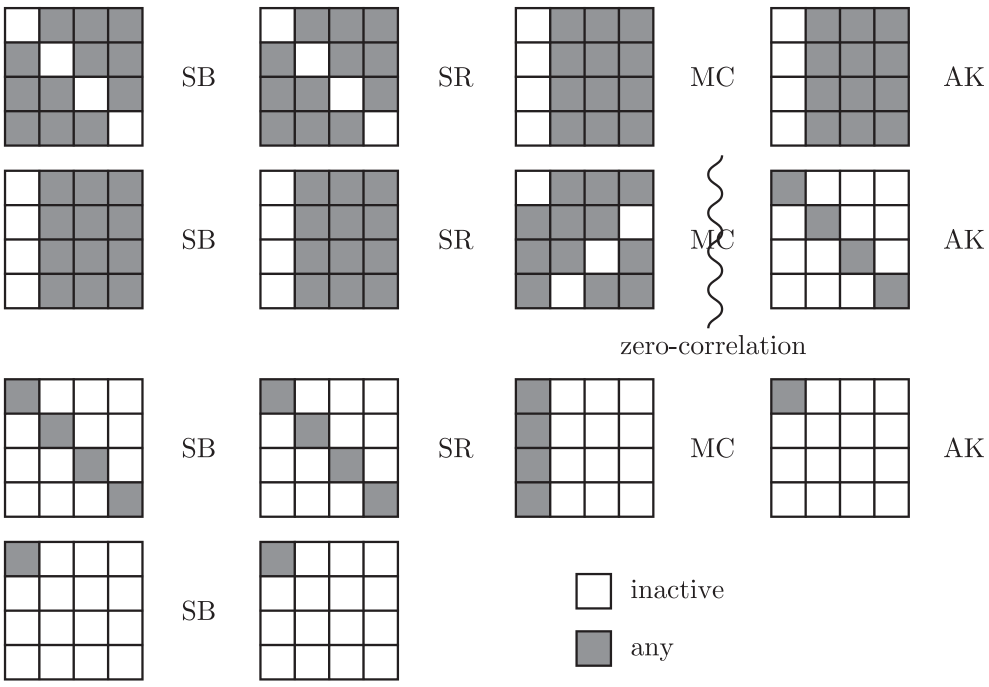


Figure 62: Zero-correlation linear hull on 4-round AES.

proximations. More precisely, the input mask α now consists of two parts, $\alpha_1 \in \mathbb{F}_2^n$ and $\alpha_2 \in \mathbb{F}_2^t$ and we have to consider

$$\mathrm{cor}_{E_k}\left((\alpha_1, \alpha_2), \beta\right) := 2 \cdot \mathrm{Prob}\left(\langle \alpha_1, P\rangle + \langle \alpha_2, T\rangle + \langle \beta, E_k(P,T)\rangle = 0\right) - 1$$

where now the probability is taken over uniformly distributed inputs P and T. Let $L : \mathbb{F}_2^t \to (\mathbb{F}_2^n)^{r+1}$ be a linear tweak-schedule, as was shown in [208]. The corresponding linear hull for this setting becomes

$$\mathrm{cor}_F\left((\alpha_1, \alpha_2), \beta\right) = \sum_{\substack{\Gamma \in (\mathbb{F}_2^n)^{r-1}, \Gamma_0 = \alpha_1, \Gamma_r = \beta \\ L^T(\Gamma) = \alpha_2}} C_\Gamma \tag{56}$$

where L^T is the adjoint linear layer of L, *i.e.*, the unique linear mapping such that

$$\langle x, L(y)\rangle = \langle L^T(x), y\rangle$$

for all x, y. If we represent L as a matrix multiplication, then L^T is the transposed matrix. This was used in [208] to argue that, in contrast to differential cryptanalysis, no new linear trails are introduced by the tweak. Thus, in order to protect against linear cryptanalysis, no fundamental new tools have to be developed. However, given the additional restriction on linear trails in the hull for tweakable ciphers, the formula actually already hints that zero-correlation might be more effective in this case.

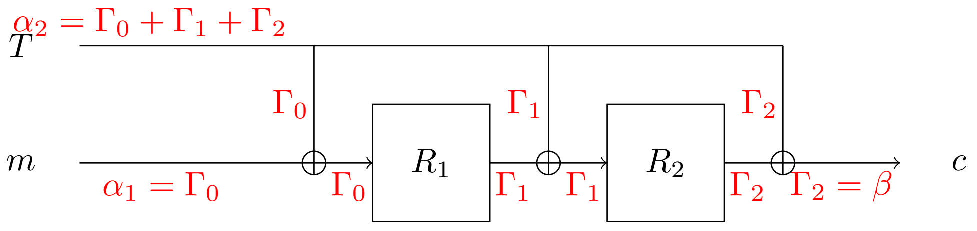


Figure 63: Propagation of masks in a simple two round tweakable block cipher.

As a first example consider the simple case of a two-round tweakable cipher, where the tweak is just XORed to the state as illustrated in Figure 63.

$$\begin{aligned} E_k : \mathbb{F}_2^n \times \mathbb{F}_2^n &\rightarrow \mathbb{F}_2^n \\ E_k(x,t) &= R_2(R_1(x+t+k)+t+k)+t+k \end{aligned}$$

Here, the tweak-scheduling is clearly linear and the mapping is simply

$$\begin{aligned} L : \mathbb{F}_2^n &\rightarrow \mathbb{F}_2^n \times \mathbb{F}_2^n \times \mathbb{F}_2^n \\ L(t) &= (t,t,t). \end{aligned}$$

The adjoint linear layer, is the mapping

$$\begin{aligned} L^T : \mathbb{F}_2^n \times \mathbb{F}_2^n \times \mathbb{F}_2^n &\rightarrow \mathbb{F}_2^n \\ L^T(t_1,t_2,t_3) &= t_1+t_2+t_3. \end{aligned}$$

Now, consider the linear hull for E_k with input mask (α_1,α_2) and output mask β. Note, that the input and output masks are independent [92]. Here α_1 is the mask for the data input and α_2 is the input mask for the tweak. According to Equation (56), the correlation of E_k becomes

$$\mathrm{cor}_{E_k}(((\alpha_1,\alpha_2),\beta)) = \sum_{\substack{\Gamma\in(\mathbb{F}_2^n)^3,\Gamma_0=\alpha,\Gamma_2=\beta \\ L^T(\Gamma)=\beta}} C_\Gamma.$$

Now as

$$L^T(\Gamma) = L^T(\Gamma_0,\Gamma_1,\Gamma_2) = \Gamma_0+\Gamma_1+\Gamma_2$$

and $\Gamma_0 = \alpha$ as well as $\Gamma_2 = \beta$, we see that $\Gamma_1 = \alpha_1+\alpha_2+\beta$ and the linear hull reduces to a *single trail*, namely

$$\mathrm{cor}_{E_k}(((\alpha_1,\alpha_2),\beta)) = \mathrm{cor}_{R_1}(\alpha_1,\alpha_1+\alpha_2+\beta)\,\mathrm{cor}_{R_2}(\alpha_1+\alpha_2+\beta,\beta)$$

Thus, for a given α_1 and β by choosing α_2 such that either $\mathrm{cor}_{R_1}(\alpha_1,\alpha_1+\alpha_2+\beta)$ or $\mathrm{cor}_{R_2}(\alpha_1+\alpha_2+\beta,\beta)$ equals zero, we derived a zero-correlation linear approximation. Thus, as long as there exist a zero-correlation linear approximation for R_1 (resp. R_2) the corresponding tweakable block cipher has a zero-correlation for *any choice of* R_2 (resp. R_1). This is the basic observation we are going to use throughout the paper for our attacks.

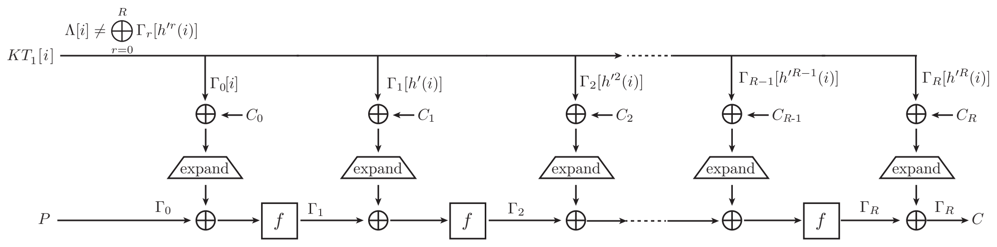


Figure 64: Related-tweakey zero-correlation linear hull on the STK with TK-1.

In the general case, we are going to use forward and backward propagation to get a superset $S \subset (\mathbb{F}_2^n)^{r+1}$ of all characteristics with non-zero correlation. Next, we check if $L(S) \subset \mathbb{F}_2^t$ is not the full space. If so, we get a zero correlation by picking the mask for the tweak in $\mathbb{F}_2^r \setminus S$. Note that, this becomes easier when the tweak-scheduling actually operates on single nibbles, as is the case for the tweakey setting of the STK construction, as we will explain next.

5.3.1 Zero-Correlation Linear Hull on STK with TK-1

For simplicity, we start our discussion from the STK construction with TK-1, and we explain the general case (TK-p) in the following.

Figure 64 shows the zero-correlation linear hull on the STK construction with TK-1. The tweakey schedule of the STK construction with TK-1 consists of the h' function, where the nibble position is simply substituted. Therefore, different nibbles are never mixed in the tweakey scheduling algorithm, and we can focus on the i^{th} nibble in KT_1. Then, given a pair of input and output linear masks (Γ_0, Γ_R), we enumerate all possible linear characteristics $(\Gamma_0, \Gamma_1, \ldots, \Gamma_R)$ and evaluate a set S such that

$$S = \left\{ \Lambda[i] = \bigoplus_{r=0}^{R} \Gamma_r[h'^r(i)] \mid \forall\, (\Gamma_0[i], \Gamma_1[h'(i)], \ldots, \Gamma_R[h'^R(i)]) \right\},$$

where $\Gamma_j[i]$ denotes the linear mask of the i^{th} nibble in Γ_j. If the complement $\mathbb{F}_2^c \setminus S$ is not empty, it causes a contradiction when $\Lambda[i] \in \mathbb{F}_2^c \setminus S$. Note that the tweakey except for the i^{th} nibble is independent of this linear hull, and it can be fixed to (secret) constant. Then, the domain expansion is only $n + c$ and not $n + t$.

Practically, we can use a miss-in-the-middle like algorithm to find such a distinguisher.

Definition 47 (Γ sequence)**.** The forward and backward propagations with probability one are evaluated from the given input linear mask Γ_0 and output linear mask Γ_r, respectively. Then, for any i, the Γ sequence is defined by the $(R+1)$ sequence, where whether $\Gamma_r[h'^r(i)]$ is *active*, *inactive*, or *any* is stored in the r^{th} element.

When the Γ sequence is inactive for any i, it causes a contradiction when $\Lambda[i]$ is an *active* mask. Moreover, when there is one active value in the Γ sequence, it causes contradiction when $\Lambda[i]$ is the zero mask. To demonstrate the Γ sequence and to show how the zero-correlation linear hull is calculated, we introduce the following toy cipher.

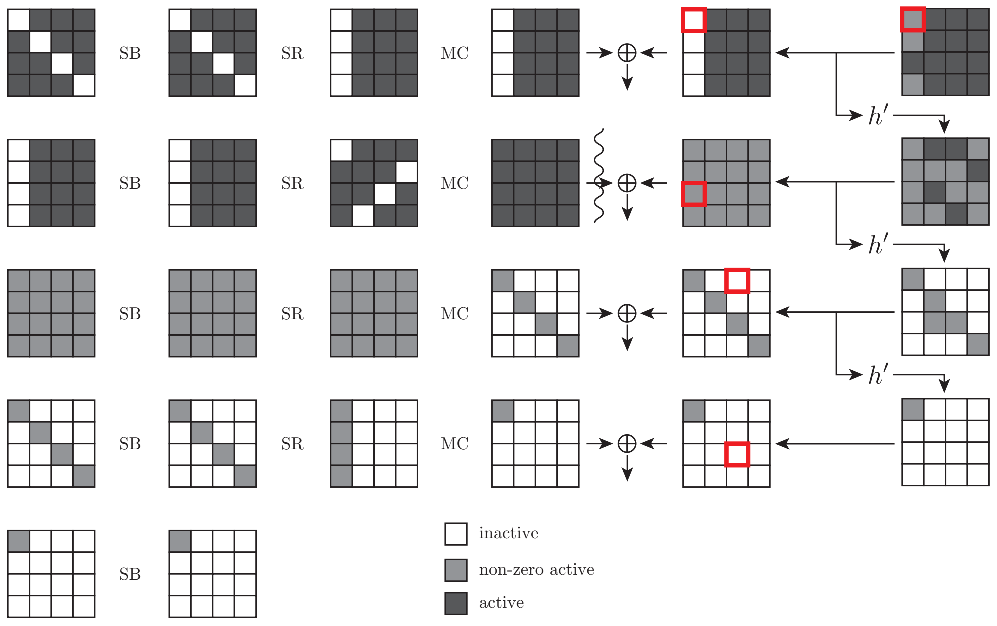


Figure 65: Related-tweakey zero-correlation linear on the Toy cipher.

Example 1 (ToyCipher)**.** The round function is exactly the same as the AES round function. A simple tweakey scheduling algorithm is adopted instead of the AES key scheduling algorithm. The full tweak state is XORed when `AddRoundKey` is originally applied, and it uses $h' = [9, 15, 8, 13, 10, 14, 12, 11, 0, 1, 2, 3, 4, 5, 6, 7]$, which is the same as the P_T of Skinny.

Figure 65 shows the 5-round related-tweakey linear hull, where we focus on the first byte in KT_1. Then, the Γ sequence is $(0, 1, 0, 0)$, where 0 and 1 denote *inactive* and *active*, respectively. Therefore, if the zero linear mask is applied to the first byte of KT_1, it derives the zero-correlation linear hull. Moreover, we convert the found zero-correlation linear hull to the corresponding integral distinguisher, as shown in [315]. *Zero* linear masks are applied to 32 state bits and 8 tweak bits, and *any* linear mask can be applied to the remaining 96 bits. Therefore, the required texts of the corresponding integral distinguisher is 2^{40}, and the distinguisher can cover 5 rounds.
An interesting observation is that the second round function is independent of the zero-correlation linear hull. In other words, this distinguisher holds even if the second round function is replaced with any permutation.

5.3.2 Zero-Correlation Linear Hull on TK-p

The STK construction with TK-p has p lines in the tweakey scheduling algorithm, and the same nibble position substitution function h' is applied to each line. On the other hand, a different coefficient α_j is multiplied with each c-bit nibble over $GF(2^c)$ in every line.

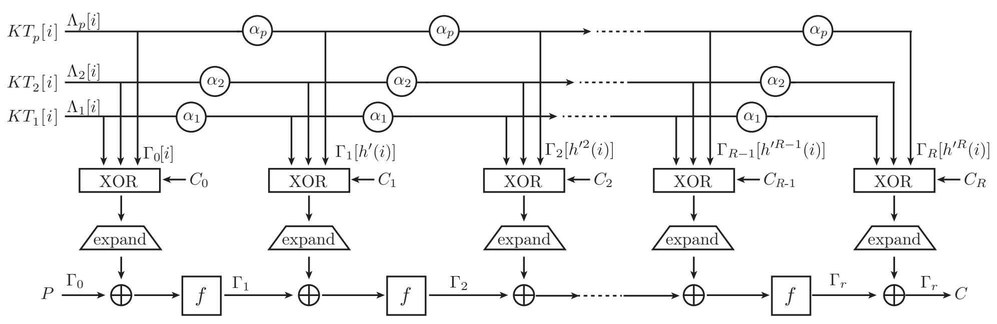


Figure 66: Related-tweakey zero-correlation linear hull on the STK with TK-p.

Similarly to the case of the zero-correlation linear hull on the STK with TK-1, we can focus on the i^th nibble in $KT_1, KT_2, \ldots, KT_p$. Sub-tweakeys are generated by the XOR of p lines and all branches connected by XOR must have the same linear mask. Therefore, we can compute the matrix $\Lambda_j[i]$ for $0 < j \leqslant p$, that contains the tweak masks, by multiplying the block matrix containing the state masks with the Γ-sequence as follows

$$\begin{pmatrix} \Lambda_1[i] \\ \Lambda_2[i] \\ \vdots \\ \Lambda_p[i] \end{pmatrix} = \begin{pmatrix} I & \alpha_1^T & (\alpha_1^T)^2 & \cdots & (\alpha_1^T)^R \\ I & \alpha_2^T & (\alpha_2^T)^2 & \cdots & (\alpha_2^T)^R \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ I & \alpha_p^T & (\alpha_p^T)^2 & \cdots & (\alpha_p^T)^R \end{pmatrix} \times \begin{pmatrix} \Gamma_0[i] \\ \Gamma_1[h'(i)] \\ \Gamma_2[h'^2(i)] \\ \vdots \\ \Gamma_R[h'^R(i)] \end{pmatrix}$$

where $\alpha_j^T : \mathbb{F}_2^c \to \mathbb{F}_2^c$ denotes the adjoint linear mapping of α_j, *i.e.*, the mapping such that

$$\langle x, \alpha_j(y) \rangle = \langle \alpha_j^T(x), y \rangle, \ \forall\, x, y \in \mathbb{F}_2^c.$$

We finally enumerate all possible linear characteristics $(\Gamma_0, \Gamma_1, \ldots, \Gamma_R)$ from a given pair of input and output linear masks (Γ_0, Γ_R). If the complement of the set of all possible $(\Lambda_1[i] \| \Lambda_2[i] \| \cdots \| \Lambda_p[p])$ is not empty, there exists a zero-correlation linear hull.

Practically, we can use the same method as in the case for TK-1. This is, chosen Γ_0 and Γ_r, we evaluate the Γ sequence for any i. Then, we can show the following proposition.

Proposition 2. *If there is a pair of linear masks* (Γ_0, Γ_r) *and the nibble position* i *such that the* Γ *sequence has at most* p *linearly active values, the tweakable block cipher has a non-trivial related-tweakey zero-correlation linear hull.*

Proof. We consider two cases: the Γ sequence is either inactive or active. The first case is trivial, and active linear mask $\Lambda_j[i]$ causes a contradiction. In the second case, we exploit the structure of the $p \times (R+1)$ block matrix. In the STK construction, α_j is chosen such that the block matrix

$$\begin{pmatrix} I & \alpha_1 & (\alpha_1)^2 & \cdots & (\alpha_1)^R \\ I & \alpha_2 & (\alpha_2)^2 & \cdots & (\alpha_2)^R \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ I & \alpha_p & (\alpha_p)^2 & \cdots & (\alpha_p)^R \end{pmatrix}$$

becomes MDS. Then, the block matrix

$$\begin{pmatrix} I & \alpha_1^T & (\alpha_1^T)^2 & \cdots & (\alpha_1^T)^R \\ I & \alpha_2^T & (\alpha_2^T)^2 & \cdots & (\alpha_2^T)^R \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ I & \alpha_p^T & (\alpha_p^T)^2 & \cdots & (\alpha_p^T)^R \end{pmatrix}$$

also becomes MDS, as—using a suitable choice for the inner product—the adjoint linear mapping α_j^T is identical to α_j, and thus the block matrix is unchanged. Therefore, in order to satisfy $\Lambda_j[i] = 0$ for all $j \in \{1, 2, \dots, p\}$, the Γ sequence must have at least $p+1$ linearly active nibbles. In other words, if there are at most p linearly active nibbles in the Γ sequence, and this causes a contradiction when all $\Lambda_j[i] = 0$. □

Proposition 2 implies that TK-$(p+1)$ always has $(r+1)$-round zero correlation linear hull when TK-p has an r-round zero-correlation linear hull.

5.4 APPLICATION TO QARMA

We apply our technique to the QARMA family of lightweight tweakable block ciphers [34]. QARMA has a block size of 64 or 128 bits, a key length of 128 or 256 bits, and a tweak length of 64 or 128 bits, respectively. We can successfully attack QARMA-64 whose numbers of forward and backward rounds are reduced to 4 and 8, respectively, under the related-tweak and chosen plaintext setting. More accurately, only 1 out of 16 cells of the tweak is active, while the other 15 cells take a known constant value. Our attack is currently the best known attack with respect to the number of total rounds.

5.4.1 Description of QARMA

An encryption of QARMA consists of forward round functions, a central construction, and backward round functions. In the specifications, the designer defines QARMA$_r$ as QARMA whose numbers of forward and backward rounds are $r+1$. In this paper however, for simplicity, we use a different notation denoted by QARMA$_{r_1,r_2}$, where the numbers of forward and backward rounds are r_1 and r_2, respectively. Thus, QARMA$_r$ corresponds to QARMA$_{r+1,r+1}$.

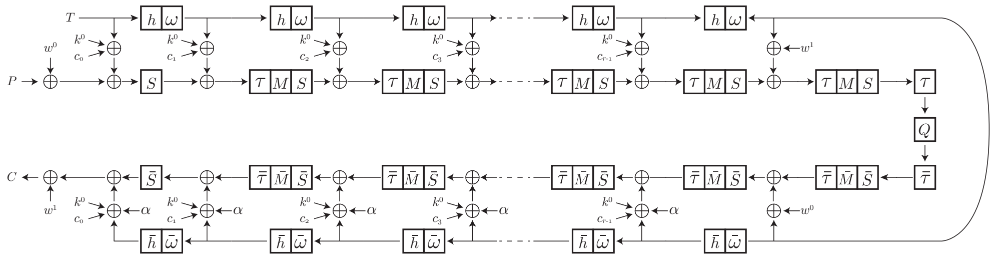


Figure 67: Structure of QARMA.

The state of QARMA is represented as a 4×4 matrix, where each index is defined as

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

Every cell takes a 4 or 8-bit value in QARMA-64 and QARMA-128, respectively. In the state denoted by X, let $X[i_1, i_2, \ldots, i_m]$ be $(s_{i_1}, s_{i_2}, \ldots, s_{i_m})$ of X.
One round of QARMA consists of the following round operations (illustrated in Figure 67):

- `AddRoundTweakey` (ART): the round tweakey is XORed with the internal state.
- `ShuffleCells` (τ): substitutes the position of cells, where the MIDORI [37] cell permutation,

  $$\tau = [0, 11, 6, 13, 10, 1, 12, 7, 5, 14, 3, 8, 15, 4, 9, 2]$$

  is used.
- `MixColumns` (M:) mixes each column in the internal state by multiplying it with the matrix M defined as

  $$M = \mathrm{circ}(0, \rho^a, \rho^b, \rho^c) = \begin{pmatrix} 0 & \rho^a & \rho^b & \rho^c \\ \rho^c & 0 & \rho^a & \rho^b \\ \rho^b & \rho^c & 0 & \rho^a \\ \rho^a & \rho^b & \rho^c & 0 \end{pmatrix},$$

  where ρ^a denotes the a-bit left rotation. For QARMA-64 and QARMA-128, $M = \mathrm{circ}(0, \rho, \rho^2, \rho)$ and $M = \mathrm{circ}(0, \rho, \rho^4, \rho^5)$ are used, respectively.
- `SubCells` (S): applies a simple 4-bit S-box to every nibble:

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
S(x)	0	e	2	a	9	f	8	b	6	4	3	7	d	c	1	5

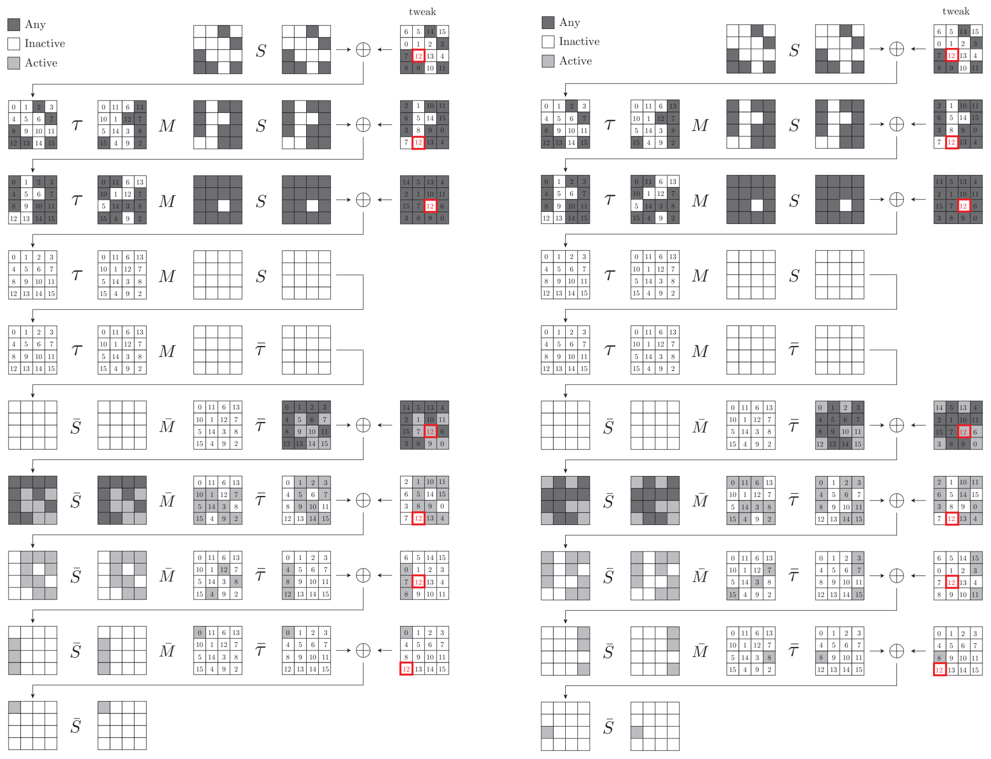


Figure 68: Two zero-correlation linear hulls on $\textsc{Qarma}_{4,5}$

The tweak schedule consists of two functions, h and ω, where the h function is defined as simple permutation $h = [6, 5, 14, 15, 0, 1, 2, 3, 7, 12, 13, 4, 8, 9, 10, 11]$. Moreover, the ω function is a bit-based LFSR. The LFSR is however irrelevant for our attack, as we only consider cell-based linear masks that are either *inactive*, *active* or *any*. Moreover, we focus on Qarma-64.

5.4.2 Zero-Correlation Linear Hull on $\text{QARMA}_{4,5}$

Figure 68 shows two zero-correlation linear hulls on $\textsc{Qarma}_{4,5}$. Any linear masks are applied to 6 cells of the state, i.e., $(s_2, s_7, s_8, s_{12}, s_{13}, s_{15})$. Moreover, active linear masks are applied to s_0 and s_8 of the output, as shown in Figure 68. Then, we focus on the tweak cell labelled 12. As illustrated in cells highlighted by red frames, the Γ sequence has just one active cell. Therefore, applying an inactive mask to the tweak cell labelled 12 causes a contradiction due to Proposition 2. Note that we do not need to activate any of the other 15 cells in the tweak and they can take any fixed value. Thus, the domain space of the zero-correlation linear hull becomes at most 17 $(= 16 + 1)$ cells.

The attack assumption of the naive algorithm using zero-correlation linear hull is the known-plaintext and tweak setting, but it usually requires a huge data complexity. If we assume a chosen-plaintext and related-tweak setting, the required data complexity can be reduced by linking to integral distinguishers. Any linear masks

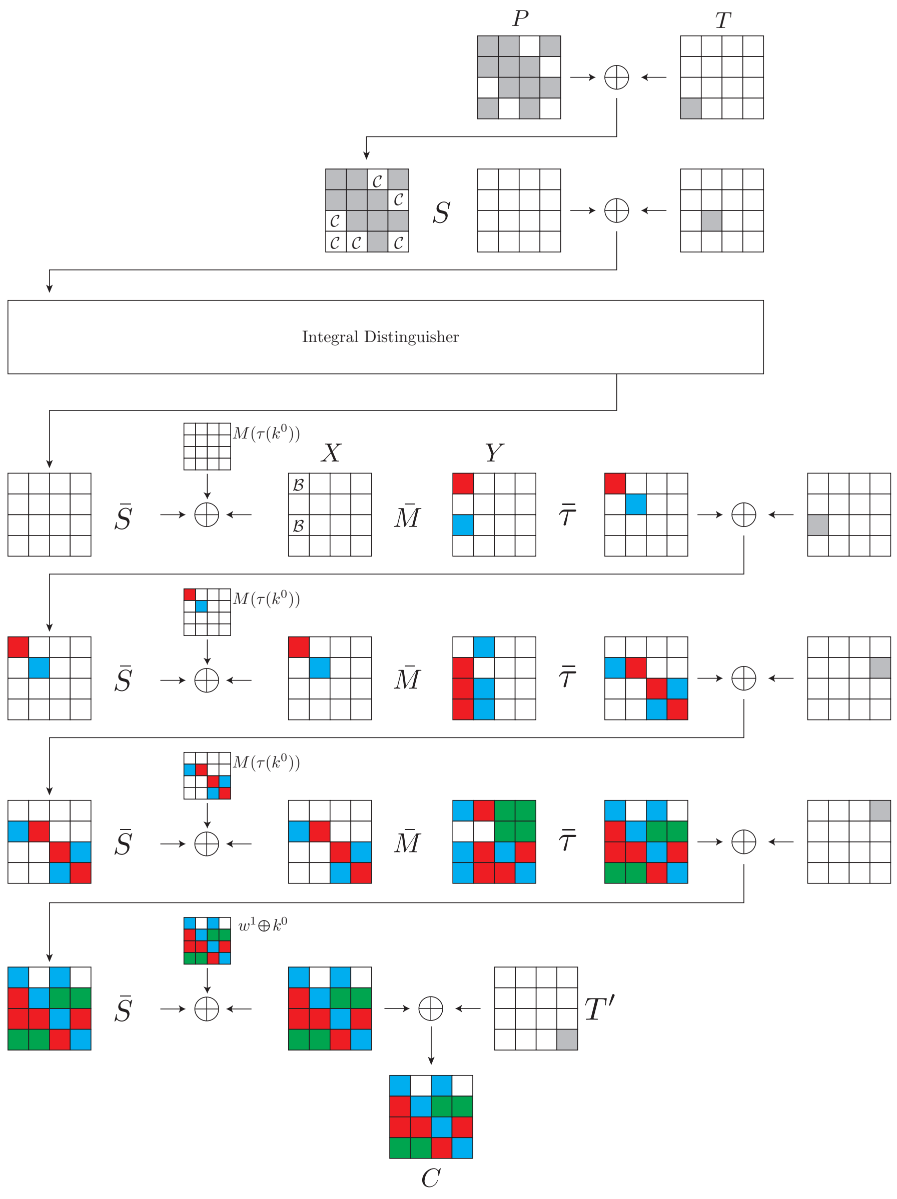


Figure 69: Key-Recovery Attacks on $\text{Qarma}_{4,8}$

are applied to six cells in the two zero-correlation linear hulls, and inactive linear masks are applied to the other 11 (= 10 + 1) cells. Therefore, the corresponding related-tweak integral distinguisher requires $2^{10\times4} = 2^{40}$ chosen plaintexts over 2^4 related tweaks, and the total data complexity is $2^{40+4} = 2^{44}$. Here, the relation of the tweak is defined in such a way that the 4-bit cell labelled 12 takes all values. Both zero-correlation linear hulls outlined in Figure 68 share the same input linear mask, and the output in position s_0 and s_8 is balanced at the same time[‡].

[‡] s_{10} is also balanced at the same time.

5.4.3 Key-Recovery Attacks on QARMA$_{4,8}$

In the key recovery, we first add pre-whitening before the integral distinguisher. Let P and T denote the states of plaintext and tweak, respectively. We first prepare a set of plaintexts and tweaks, where 10 cells at position $P[0,1,3,4,5,6,9,10,11,14]$ and 1 cell at position $T[12]$ are active. Moreover, we choose the input such that $P[12] = T[12]$, and then, $(P \oplus T)[12]$ becomes constant, and further coincides to the input of the integral distinguisher.
We can append three rounds after the integral distinguisher. Figure 69 shows the key-recovery, and let X and Y be the states defined in Fig. 69. Due to the integral distinguisher, both s_0 and s_8 in X are balanced at the same time. Then

$$\begin{pmatrix} X_0 \\ X_4 \\ X_8 \\ X_{12} \end{pmatrix} = \begin{pmatrix} 0 & \rho & \rho^2 & \rho \\ \rho & 0 & \rho & \rho^2 \\ \rho^2 & \rho & 0 & \rho \\ \rho & \rho^2 & \rho & 0 \end{pmatrix} \begin{pmatrix} Y_0 \\ Y_4 \\ Y_8 \\ Y_{12} \end{pmatrix} = \begin{pmatrix} \rho Y_4 + \rho^2 Y_8 + \rho Y_{12} \\ \rho Y_0 + \rho Y_8 + \rho^2 Y_{12} \\ \rho^2 Y_0 + \rho Y_4 + \rho Y_{12} \\ \rho Y_0 + \rho^2 Y_4 + \rho Y_8 \end{pmatrix},$$

and

$$\begin{aligned} \bigoplus_{i=0}^{n} (X_0 + X_8) &= \bigoplus_{i=0}^{n} (\rho Y_4 + \rho^2 Y_8 + \rho Y_{12}) \oplus (\rho^2 Y_0 + \rho Y_4 + \rho Y_{12}) \\ &= \bigoplus_{i=0}^{n} (\rho^2 Y_0 + \rho^2 Y_8) = 0. \end{aligned}$$

Moreover ρ^2 is the rotation function, $\bigoplus_{i=0}^{n}(Y_0 + Y_8) = 0$. We use the meet-in-the-middle technique for the integral attacks [301], where $\bigoplus_{i=0}^{n} Y_0$ and $\bigoplus_{i=0}^{n} Y_8$ are evaluated independently, and round keys satisfying $\bigoplus_{i=0}^{n} Y_0 = \bigoplus_{i=0}^{n} Y_8$ are recovered. The size of the involved secret-key is $56 (= 14 \times 4)$ bits in $w^1 \oplus k^0$ and $28 (= (1+6) \times 4)$ bits in $M(\tau(k^0))$, which gives a total of 84 key-bits that can be recovered. Since one structure removes incorrect secret-key bits by a factor of 2^{-4}, we need $84/4 = 21$ structures to uniquely determine the secret-key.
To compute $\bigoplus_{i=0}^{n} Y_0$, cells labelled red and green are involved, where 36-bit of ciphertexts, 36-bit of $w^1 \oplus k^0$, and 16-bit of $M(\tau(k^0))$ are involved. We first store the frequency of 36-bit $(C[4,6,7,8,9,11,12,13,14])$ into memory. Then, we use the FFT key-recovery technique [324], and the time complexity can be evaluated as

$$21 \times 4 \times (2^{16} \times (3 \times 36 \times 2^{36})) \approx 2^{65.2}.$$

As a result, we generate a list whose size is $2^{36+16} = 2^{52}$ and each value takes $((21 \times 4) + 16 + 4) = 104$ bits, where (21×4)-bit are the concatenation of $\bigoplus_{i=0}^{n} X_0$ in 21 structures, 16-bit are $(w^1 \oplus k^0)[6,7,12,13]$, and 4-bit are $M(\tau(k^0))[5]$.
Similarly, cells labelled blue and green are involved to compute $\bigoplus_{i=0}^{n} X_8$, where 36-bit of ciphertexts, 36-bit of $w^1 \oplus k^0$, and 16-bit of $M(\tau(k^0))$ are involved. Then, we store the frequency of 36-bit $(C[0,2,5,6,7,10,12,13] \| C[15] \oplus T'[15])$ into memory. Again, the FFT key-recovery technique enables us to compute the sum with $2^{65.2}$ computations, and we generate a similar list as in the case of $\bigoplus_{i=0}^{n} Y_0$. Finally, we compare these lists to find a match, and the time complexity is 2^{52}.

Thus, the total time complexity is $2 \times 2^{65.2} + 2^{52} \approx 2^{66.2}$, and the required data complexity is $21 \times 2^{44} \approx 2^{48.4}$. The memory complexity is determined by storing our two lists and is $2 \times 104/64 \times 2^{52} = 2^{53.70}$ 64-bit blocks. We already recover 56-bit $w^1 \oplus k^0$ and 28-bit $M(\tau(k^0))$, and there are still 44 bits of the secret-key, remaining. Finally, we exhaustively guess these bits, but the complexity, i.e., 2^{44}, is negligible compared with $2^{66.2}$. The security of QARMA-64 is claimed as $2^{128-d-2}$ where 2^d chosen or known {plaintext, ciphertext, tweak} triples, i.e., $2^{128-48.46-2} = 2^{77.54}$ in our case. Therefore, our attack against $\text{QARMA}_{4,8}$ is valid.

5.5 APPLICATION TO MANTIS

In this section, we apply the attack to a reduced-round version of MANTIS_8, where the number of forward and backward rounds are reduced to 4 and 8, respectively. Our attack assumption is the same as the case of QARMA, where only 1 cell in the tweak is activated and the other 15 cells can take any known constant.

5.5.1 Description of MANTIS

MANTIS is a family of lightweight tweakable block ciphers proposed by Beierle *et al.* [50] together with SKINNY. MANTIS has a block size of 64 bits, a key length of 128 bits, and a tweak length of 64 bits, respectively. The structure of MANTIS follows the design of PRINCE [101] and is aimed to achieve low-latency. While it is rather easy to turn a cipher into a tweakable cipher by using the TWEAKEY framework, the designers reused components of MIDORI [37] to achieve low-latency. One round of MANTIS consists of the following round operations (illustrated in Figure 70):

- `SubCells` (SC): substitutes each nibble x by the involuntary MIDORI S-box $Sb_0(x)$ that is given below:

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S(x)$	c	a	d	3	e	b	f	7	8	9	1	5	0	2	4	6

- `AddConstant` (AC): adds a round constant RC_i to the state. The constants are similarly generated as in PRINCE.
- `AddRoundTweakey` (ART): adds the (full) round tweakey to the internal state.
- `PermuteCells` (PC): applies the cell permutation of MIDORI as given below:

$$P = [0, 11, 6, 13, 10, 1, 12, 7, 5, 14, 3, 8, 15, 4, 9, 2].$$

- `MixColumns` (MC): multiplies each column of the state by the binary matrix from MIDORI M as shown below:

$$M = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

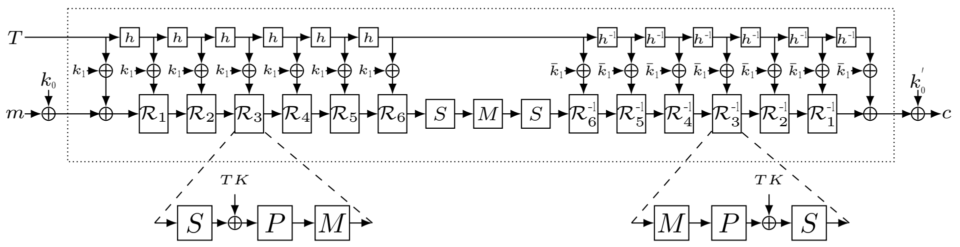


Figure 70: Illustration of the tweakable block cipher Mantis.

The state is represented as a 4×4 matrix, where each index is defined as

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

In the state denoted by X, let $X[i_1, i_2, \dots, i_m]$ be $(s_{i_1}, s_{i_2}, \dots, s_{i_m})$ of X. The encryption of Mantis consists of a forward round function, a central construction, and backward round function, similar as in Qarma. The designers of Mantis defines Mantis_r as Mantis whose numbers of forward and backward rounds are r. For simplicity, we use different notation denoted by Mantis_{r_1,r_2}, where the numbers of forward and backward rounds are r_1 and r_2, respectively.

5.5.2 Zero-Correlation Linear Hull on $\text{MANTIS}_{4,5}$

The related-tweak zero-correlation linear distinguishers and the consequential integral distinguishers for $\text{Mantis}_{4,5}$ are identical to the distinguishers on $\text{Qarma}_{4,5}$. This is because, *wlog.* we can re-arrange the components of the round function in Mantis so that the overall structure of Mantis is the same as for Qarma. We can define $\text{Mantis}_r \sim \text{Qarma}_r$ by changing the order of the round components from

$$\texttt{MixColumns} \circ \texttt{PermuteCells} \circ \texttt{AddTweakey}_{tk} \circ \texttt{AddConstanct}_i \circ \texttt{SubCells}$$

to

$$\texttt{SubCells} \circ \texttt{MixColumns} \circ \texttt{PermuteCells} \circ \texttt{AddTweakey}_{tk} \circ \texttt{AddConstanct}_i$$

that is equivalent to the round structure of Qarma. Moreover, as the first and last round of Qarma are partial rounds (omitting `ShuffleCells` and `MixColumns`) this works for the beginning of the forward/backward rounds. Furthermore, as Qarma employs one forward and one backward round in the central construction, `ShuffleCells` and `MixColumns` can be added from Mantis to complete the last round of the forward/backward rounds. The remaining application of one S-box in Qarma is then equivalent to the application of an S-box in the middle construction of Mantis.

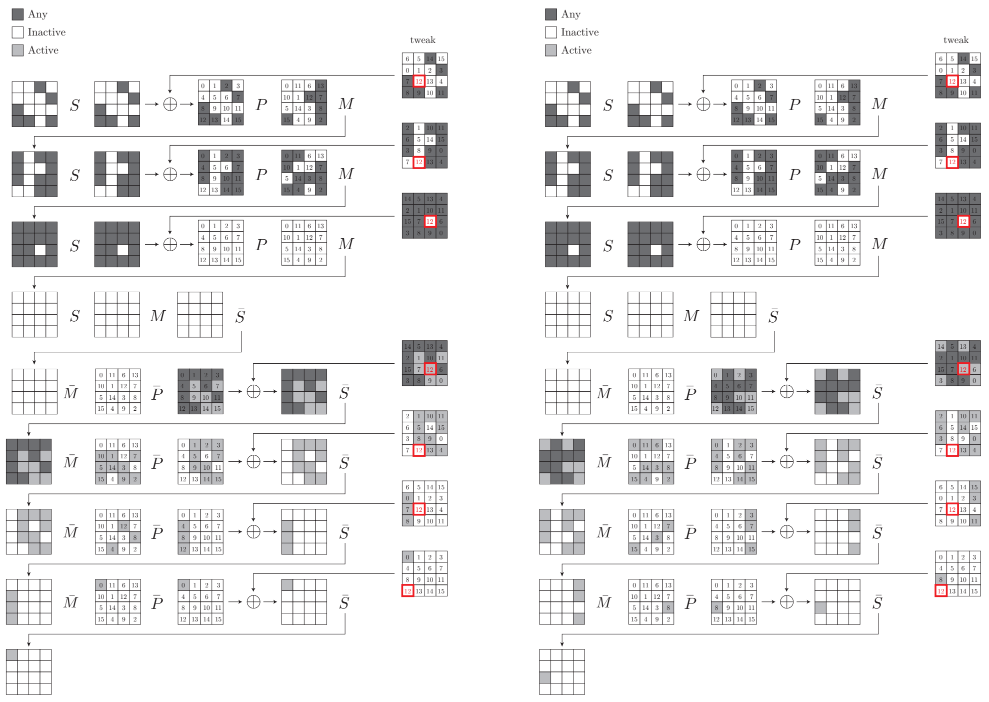


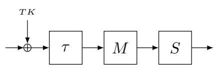
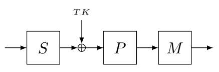
Figure 71: Two zero-correlation linear hulls on $\text{Mantis}_{4,5}$.

Since our attack is of a general nature, and the components of Mantis and Qarma are very similar (see the differences in Table 22) the distinguishers of Qarma can be re-used. All operations of both Mantis and Qarma are in a nibble-by-nibble fashion, and the alignment of the state words is the same. Moreover, in the search for the related-tweak zero-correlation distinguishers, we consider the m-bit S-box as an arbitrary S-box and do not consider the structure of a particular S-box. Similar the linear layer of Mantis and Qarma just differ by some entries of the `MixColumns` matrix M, but again as we consider nibble-by-nibble operations and the matrices have the same structure (with an all zero-diagonal) there are no differences in the distinguisher. Finally, the additional application of an LFSR ω in the tweak-update function of Qarma also does not change the distinguisher, with a similar argument as for the differences in the `MixColumns` matrix M.
Figure 71 explicitly shows the distinguishers for $\text{Mantis}_{4,5}$, where cells s_0 and s_8 after `MixColumns` are linearly *active*, respectively.

5.5.3 Key-Recovery Attacks on $\text{Mantis}_{4,8}$

Since our attacks are general against the Tweakey framework, and we can reuse the distinguishers of Qarma on Mantis, we can further reuse the key-recovery for Mantis.
Qarma uses a 128-bit master key K that is initially partitioned as $w^0 \| k^0$, where w^i are the whitening keys and k^i are the core keys, respectively, for $i \in \{0, 1\}$. For encryption, $k^0 = k^1$ and $w^1 = (w^0 \ggg 1) \oplus (w^0 \gg (64 - 1))$.

Table 22: Comparison between Mantis and Qarma.

	Qarma	Mantis
Round function		
S-box	$\sigma_1 = [a, d, e, 6, f, 7, 3, 5, 9, 8, 0, c, b, 1, 2, 4]$	$Sb_0 = [c, a, d, 3, e, b, f, 7, 8, 9, 1, 5, 0, 2, 4, 6]$
	$\tau = [0, b, 6, d, a, 1, c, 7, 5, e, 3, 8, f, 4, 9, 2]$	$P = [0, b, 6, d, a, 1, c, 7, 5, e, 3, 8, f, 4, 9, 2]$
Linear Layer	$M = \begin{pmatrix} 0 & \rho & \rho^2 & \rho \\ \rho & 0 & \rho & \rho^2 \\ \rho^2 & \rho & 0 & \rho \\ \rho & \rho^2 & \rho & 0 \end{pmatrix}$	$M = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$

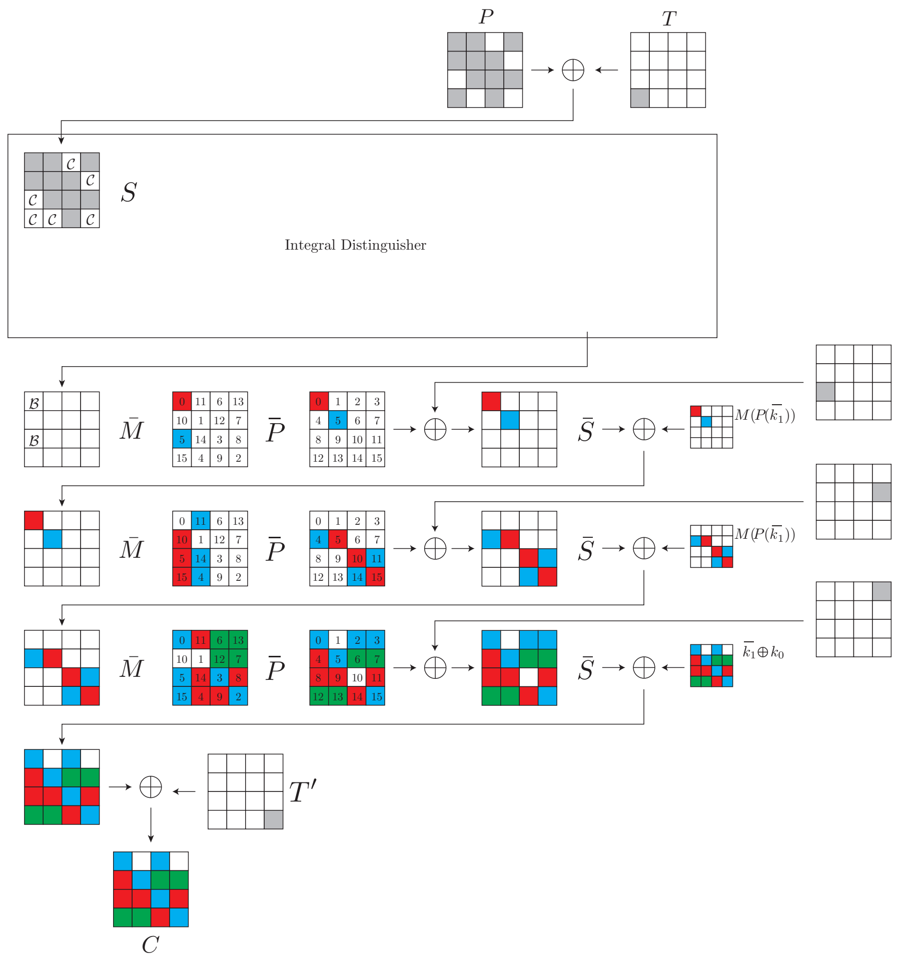


Figure 72: Key-recovery attack on $\text{Mantis}_{4,8}$.

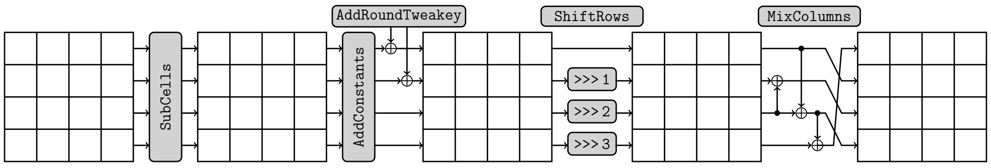


Figure 73: Round function of the tweakable block cipher SKINNY.

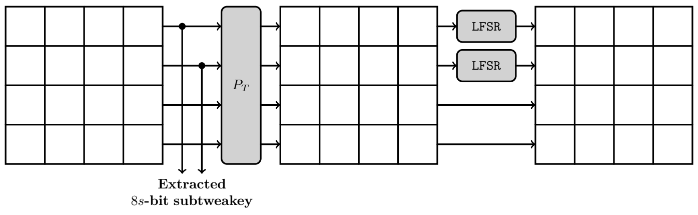


Figure 74: TWEAKEY schedule of SKINNY.

MANTIS uses a 128-bit master key K that is split into $k_0||k_1$ that is then further extended to the 192-bit key

$$(k_0||k_0'||k_1) = (k_0||(k_0 \ggg 1) \oplus (k_0 \gg 63)||k_1)$$

where k_0, k_0' are the whitening keys and k_1 is the round key for all rounds in MANTIS.
In the key-recovery of QARMA we recover 56-bit of $w^1 \oplus k^0$ and 28-bit of $M(\tau(k^0))$. Since $w^1_{\text{QARMA}} = k'_{0_{\text{MANTIS}}}$ and $k^0_{\text{QARMA}} = \bar{k}_{1_{\text{MANTIS}}}$ we can recover the same key information as in QARMA (i.e., we can recover 56-bit of $k_0' \oplus \bar{k}^1$ and 28-bit of $M(P(\bar{k}^1))$). Equally, as in the attack on QARMA$_{4,8}$ the complexities to attack MANTIS$_{4,8}$ are $2^{66.2}$ for time complexity, $2^{48.4}$ for data complexity, and $2^{53.64}$ 64-bit blocks for the memory complexity. Figure 72 explicitly shows the key recovery for MANTIS$_{4,8}$.

5.6 APPLICATION TO SKINNY

In this section, we apply the attack to reduced-round versions of SKINNY-64/128 and SKINNY-64/192. SKINNY [50] is designed according to the STK construction with TK-p, where $p \in \{1, 2, 3\}$. We show attacks on 20 rounds of SKINNY-64/128 and 23 rounds of SKINNY-64/192.

5.6.1 Description of SKINNY

SKINNY is a family of lightweight tweakable block ciphers proposed by Beierle *et al.* [50]. SKINNY has a block size n of 64 or 128 bits, and a tweakey size of $n/2n/3n$, where the tweakey can be both tweak and key. The aim of SKINNY is to achieve the performance of the NSA ciphers SIMON and SPECK [46], while still offering strong security bounds against differential/linear cryptanalysis. One round of SKINNY consists of the following round operations (illustrated in Figure 73):

- `SubCells` (SC): substitutes each nibble x by the S-box $S(x)$ which is given below:

x	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$S(x)$	c	6	9	0	1	a	2	b	3	8	5	d	4	e	7	f

- `AddRoundConstants` (AC): adds LFSR-based round constants to cells $0, 4$, and 8 of the state.
- `AddRoundTweakey` (ART): adds the round tweakey to the first two rows of the state.
- `ShiftRows` (SR): rotates the i^{th} row, for $i = 0 \leqslant i \leqslant 3$, by i positions to the right.
- `MixColumns` (MC): multiplies each column of the state by the binary matrix M:

$$M = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

The state is represented as a 4×4 matrix, where each index is defined as

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

In the state denoted by X, let $X[i_1, i_2, \dots, i_m]$ be $(s_{i_1}, s_{i_2}, \dots, s_{i_m})$ of X. In our work, we just consider Skinny-64, however, the attack should easily be applicable to Skinny-128. For two tweakey words (*i.e.,* TK-2) the designers of Skinny recommend 36 rounds, for three tweakey words (*i.e.,* TK-3) the designers recommend 40 rounds.

5.6.2 Zero-Correlation Linear Hull on Skinny-64/128

We searched the zero-correlation linear hull by using the miss-in-the-middle like algorithm. As a result, we found a 13-round zero-correlation linear hull for Skinny-64/128. Here, active linear masks are applied to two cells (s_0, s_3) at the input, and active linear masks are applied to cells s_7 and s_{11} in the state before `MixColumns` at the output, as shown in Figure 75 and Figure 76. Then, we focus on the tweak cell labelled 9, where the Γ sequence is depicted by using a red frame. Since the Γ sequence has just two active cells and Skinny-64/128 is based on TK-2, applying an inactive mask to the before mentioned tweak cell causes a contradiction due to Proposition 2. Note that the remaining 15×2 cells in the tweakey can take any constant, and the domain of our zero-correlation linear hull is $64 + 8 = 72$ bits.

We can link the zero-correlation linear hull to a related-tweakey integral distinguisher. We apply any linear mask to the two cells (s_0, s_3) in the zero-correlation linear hulls as illustrated in Figure 75 and Figure 76 and apply inactive linear masks to the remaining 14 cells. Moreover, we apply inactive linear masks to the $2 \times 4 = 8$-bit tweak cell labelled 9. Therefore, the corresponding related-tweakey integral distinguisher requires $2^{14\times 4} = 2^{56}$ chosen plaintexts over 2^8 related tweakeys, and the total data complexity is $2^{56+8} = 2^{64}$. Here, the relation of the tweakey is defined in such a way that the $2 \times 4 = 8$-bit cell labelled 9 takes all values. The integral distinguishers share the same input linear masks Γ_0, and the output in cell s_{11} in the state after `MixColumns` is balanced.

5.6.3 Key-Recovery for Skinny-64/128

Our attack model is a related-tweakey attack, where 2^8 related tweakeys are exploited. Then, there exist generic key-recovery attack with the time complexity of $2^{128-8} = 2^{120}$ [45]. Therefore, the time complexity of a non-trivial key-recovery attack must be at most 2^{120}.
In the key-recovery, we can prepend 1 round and append 6 rounds to the integral distinguisher. In total the attack reaches 20 rounds. Figure 77 shows the key-recovery, and let X_i, Y_i, and Z_i be the states defined in Figure 77. Let P and T be the states of plaintext and tweak, respectively. We first prepare a set of chosen Z_1, where 14 cells are active, i.e., $Z_1[1,2,4,5,6,7,8,9,10,11,12,13,14,15]$. Moreover, we need the tweak cell $T[1]$ in the two lines to be active, as it will propagate to cell $T[9]$ after 1 round, which coincides with the beginning of both integral distinguishers as shown in Figure 75 and Figure 76. Note that the consistent set of chosen plaintexts and tweaks is computed from Z_1 and $T[1]$ without guessing any bits, since Skinny does not have a whitening-key addition at the beginning.
Due to the integral distinguisher, both s_7 and s_{11} in Y_{14} are balanced. Then

$$\begin{pmatrix} Z_{14}[3] \\ Z_{14}[7] \\ Z_{14}[11] \\ Z_{14}[15] \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} Y_{14}[3] \\ Y_{14}[7] \\ Y_{14}[11] \\ Y_{14}[15] \end{pmatrix} = \begin{pmatrix} Y_{14}[3] + Y_{14}[11] + Y_{14}[15] \\ Y_{14}[3] \\ Y_{14}[7] + Y_{14}[11] \\ Y_{14}[3] + Y_{14}[11] \end{pmatrix},$$

and

$$\bigoplus_{i=0}^{n} Z_{14}[11] = \bigoplus_{i=0}^{n} (Y_{14}[7] \oplus Y_{14}[11]) = 0$$

In Skinny, the full tweakey is not XORed with the internal state (i.e., just the top two rows of the tweakey are XORed to the state), and then, the FFT key-recovery technique is less efficient [324]. Therefore, we estimate the time complexity to recover round keys satisfying $\bigoplus_{i=0}^{n} Y_{11} = 0$ in detail by using the partial-sum technique [155]. The size of the involved secret key is $(7+6+4+2+1+0) \times 4 = 80$ bits, and one structure filters incorrect secret-key guesses by a factor of 2^{-4}. Therefore, we need about $80/4 = 20$ structures to uniquely determine the secret key.

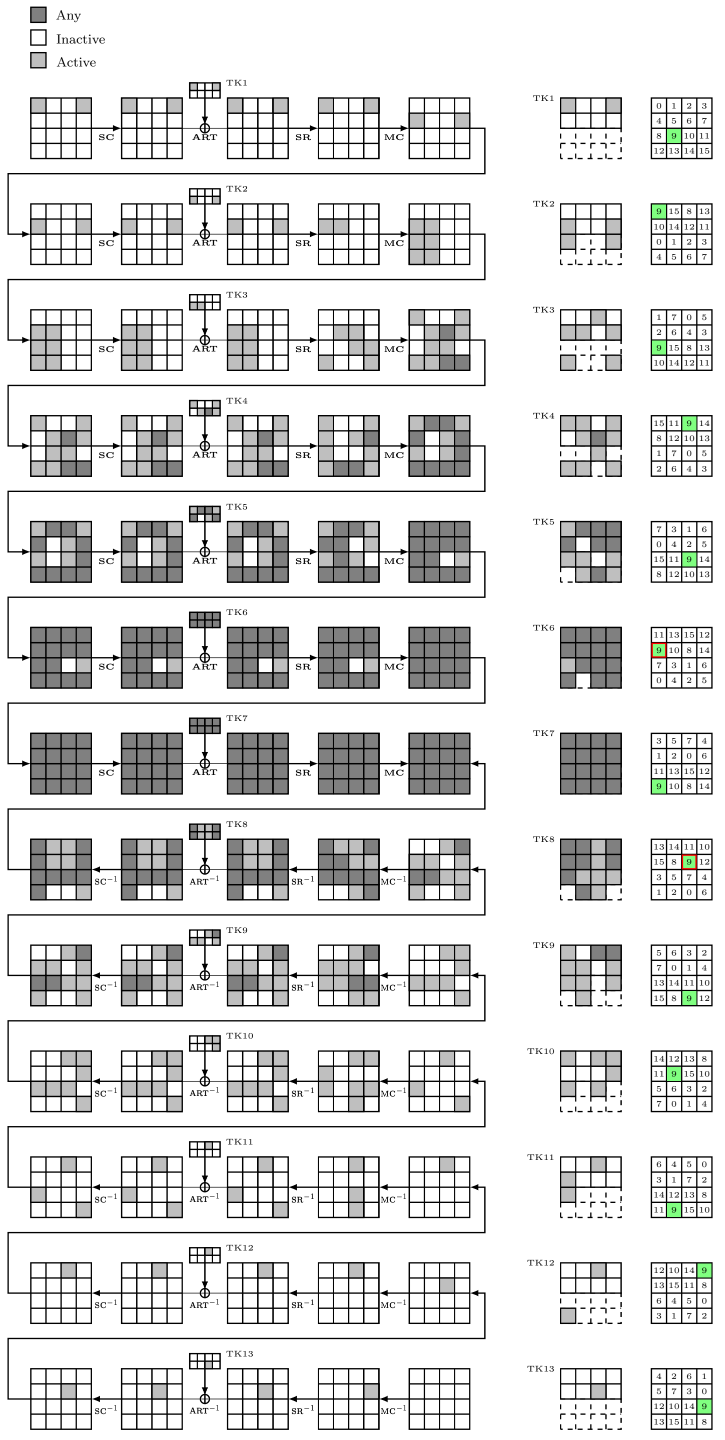


Figure 75: Zero-Correlation Linear Distinguisher for Skinny-64/128.

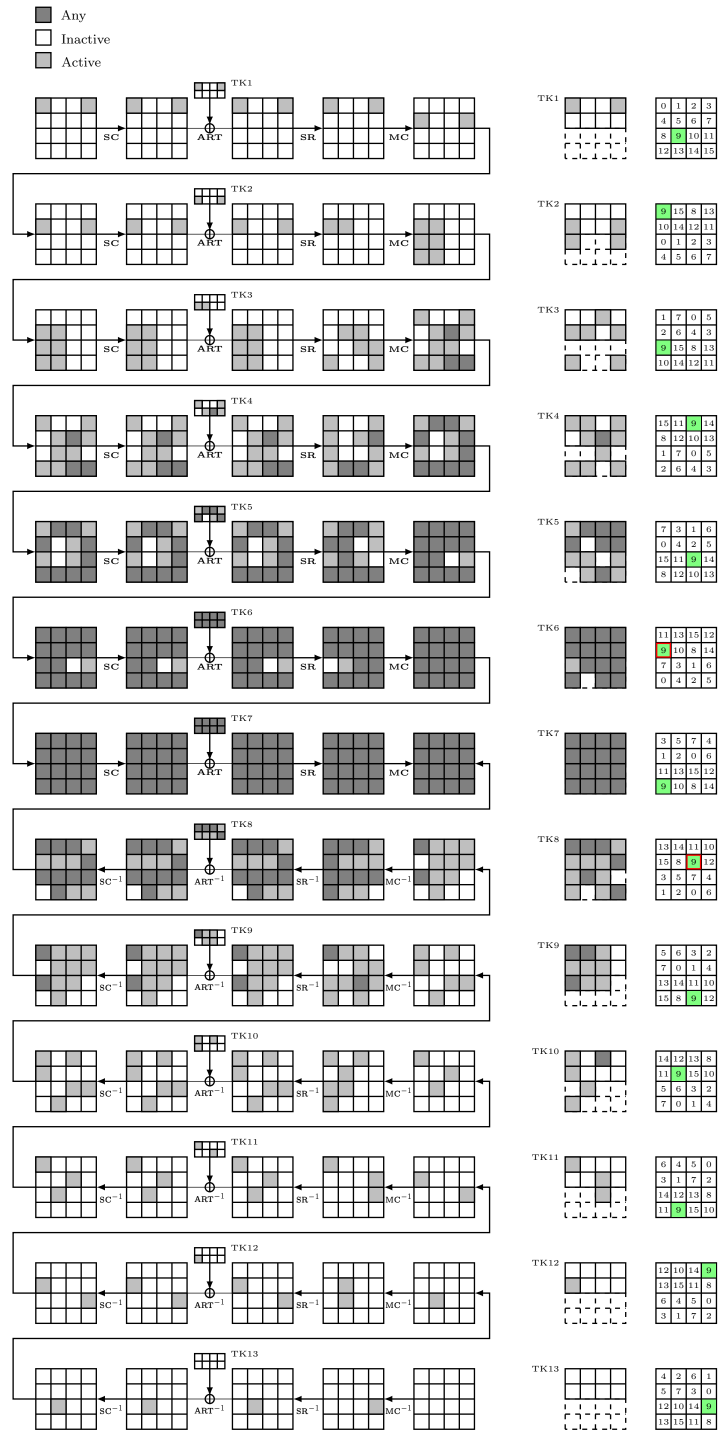


Figure 76: Zero-Correlation Linear Distinguisher for Skinny-64/128.

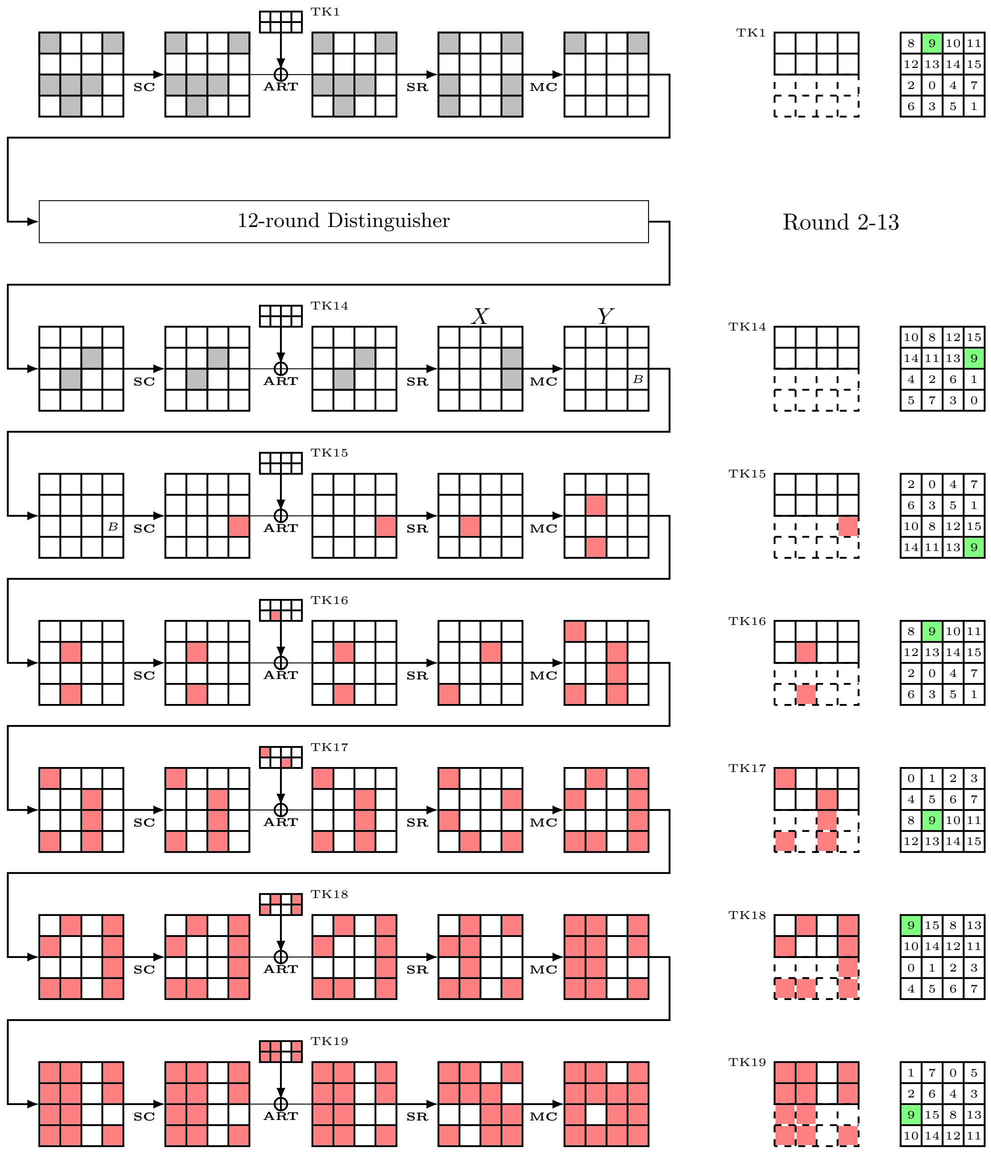


Figure 77: Key Recovery Attacks on 19 rounds of SKINNY-64/128.

Table 23 summarises the procedure of the partial-sum technique, where the time complexity can be computed as

$$3 \times 2^{64} + 2^{72} + 2^{76} + 2 \times 2^{80} + 2^{84} + 2 \times 2^{92} + 4 \times 2^{88} \approx 2^{93.2}$$

We need to repeat this procedure 20 times to recover the secret-key. Thus, the total time complexity is $2^{97.5}$, the data complexity is $20 \times 2^{64} \approx 2^{68.4}$, and the memory complexity is $1/64 \cdot 2^{88} = 2^{82}$ 64-bit blocks. Note that our attack requires a data complexity above 2^{64}, however, we do not need to collect the full-codebook under a fixed tweakey.

Table 23: Procedure for the key-recovery on Skinny-64/128

Step	Guessed Key	Data	Stored Texts	Memory (bits)	Complexity
0		2^{56}	X_{20}[0,1,3,4,5,6,7,8,10,11,12,13,14,15]	2^{56}	2^{64}
1	TK_{20}[1,5]	2^{52}	X_{20}[0,3,4,6,7,8,10,11,12,14,15],Y_{19}[1,5]	$2^{52+8} = 2^{60}$	$2^{56+8} = 2^{64}$
2	TK_{20}[6]	2^{52}	X_{20}[0,3,4,7,8,11,12,15],Y_{19}[1,5,6,10,14]	$2^{52+12} = 2^{64}$	$2^{52+12} = 2^{64}$
3	TK_{20}[0,4]	2^{48}	X_{20}[3,7,11,15],Y_{19}[0,1,4,5,6,10,12,14]	$2^{48+20} = 2^{68}$	$2^{52+20} = 2^{72}$
4	TK_{20}[3,7]	2^{44}	Y_{19}[0,1,3,4,5,6,10,11,12,14,15]	$2^{44+28} = 2^{72}$	$2^{48+28} = 2^{76}$
-		2^{44}	X_{19}[0,1,3,4,5,7,8,9,12,13,15]		
5	TK_{19}[0,4]	2^{36}	X_{19}[1,3,5,7,9,13,15],Y_{18}[4,12]	$2^{36+36} = 2^{72}$	$2^{44+36} = 2^{80}$
6	TK_{19}[3,7]	2^{32}	X_{19}[1,5,9,13],Y_{18}[3,4,12,15]	$2^{32+44} = 2^{76}$	$2^{36+44} = 2^{80}$
7	TK_{19}[1,5]	2^{32}	Y_{18}[1,3,4,5,9,12,13,15]	$2^{32+52} = 2^{84}$	$2^{32+52} = 2^{84}$
-		2^{32}	X_{18}[1,3,4,7,11,12,13,15]		
8	TK_{18}[3,7]	2^{24}	X_{18}[1,4,12,13],Y_{17}[7,15]	$2^{24+60} = 2^{84}$	$2^{32+60} = 2^{92}$
9	TK_{18}[1]	2^{20}	X_{18}[4,12],Y_{17}[7,13,15]	$2^{20+64} = 2^{84}$	$2^{24+64} = 2^{88}$
10	TK_{18}[4]	2^{20}	Y_{17}[0,7,8,13,15]	$2^{20+68} = 2^{88}$	$2^{20+68} = 2^{88}$
-		2^{20}	X_{17}[0,6,10,12,14]		
11	TK_{17}[6]	2^{12}	X_{17}[0,12],Y_{16}[6]	$2^{12+72} = 2^{84}$	$2^{20+72} = 2^{92}$
12	TK_{17}[0]	2^{8}	Y_{16}[6,12]	$2^{8+76} = 2^{84}$	$2^{12+76} = 2^{88}$
12	TK_{16}[5]	2^{4}	Z_{14}[11]	$2^{4+80} = 2^{84}$	$2^{8+80} = 2^{88}$

5.6.4 Related-Tweak Zero-Correlation Linear Distinguisher on Skinny-64/192

We can reuse parts of the zero-correlation linear hull for Skinny-64/128 in the TK-2 setting for that of Skinny-64/192 in the TK-3 setting. Therefore, we apply any linear mask to cells (s_0, s_3) in the input mask Γ_0. In contrast to the case for Skinny-64/128, we now apply active linear masks to only cell s_9 in the state before `MixColumns`, as shown in Figure 78. Then, we focus on the tweakey cell labelled 7, and the Γ sequence has now three active cells. Again, by using Proposition 2 and applying an inactive mask to the before mentioned tweakey cell, this causes a contradiction. Note that the remaining 15×3 cells in the tweakey can take any constant, and the domain of our zero-correlation linear hull is $64 + 12 = 76$ bits.

Again, we link the zero-correlation linear hull to a related-tweakey integral distinguisher. We apply any linear mask to the two cells (s_0, s_3) in the zero-correlation linear hulls as illustrated in Figure 78 and apply inactive linear masks to the remaining 14 cells. Moreover, we apply inactive linear masks to the $3 \times 4 = 12$-bit tweak cell labelled 7. Therefore, the corresponding related-tweakey integral distinguisher requires $2^{14\times4} = 2^{56}$ chosen plaintexts over 2^{12} related tweakeys, and the total data complexity is $2^{56+12} = 2^{68}$. Here, the relation of the tweakey is defined in such a way that the $3 \times 4 = 12$-bit tweak cell labelled 7 takes all values. The cell s_9 before `MixColumns` is balanced because any linear mask is applied to the cell.

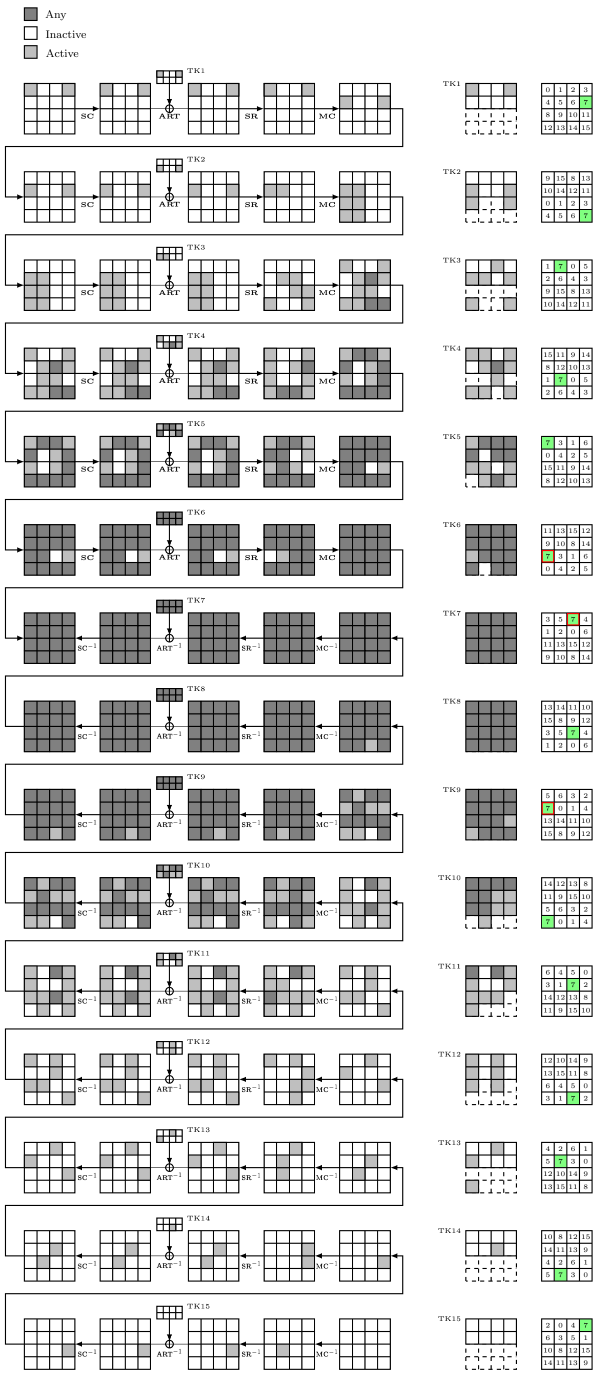


Figure 78: Zero-Correlation Linear Distinguisher for SKINNY-64/192.

5.6.5 Key-Recovery for SKINNY-64/192

Our integral distinguisher uses 2^{12} related tweakeys, and then, there exist generic key-recovery attack with a time complexity of $2^{192-12} = 2^{180}$ [45]. Therefore, the time complexity of a non-trivial key-recovery attack must be at most 2^{180}.

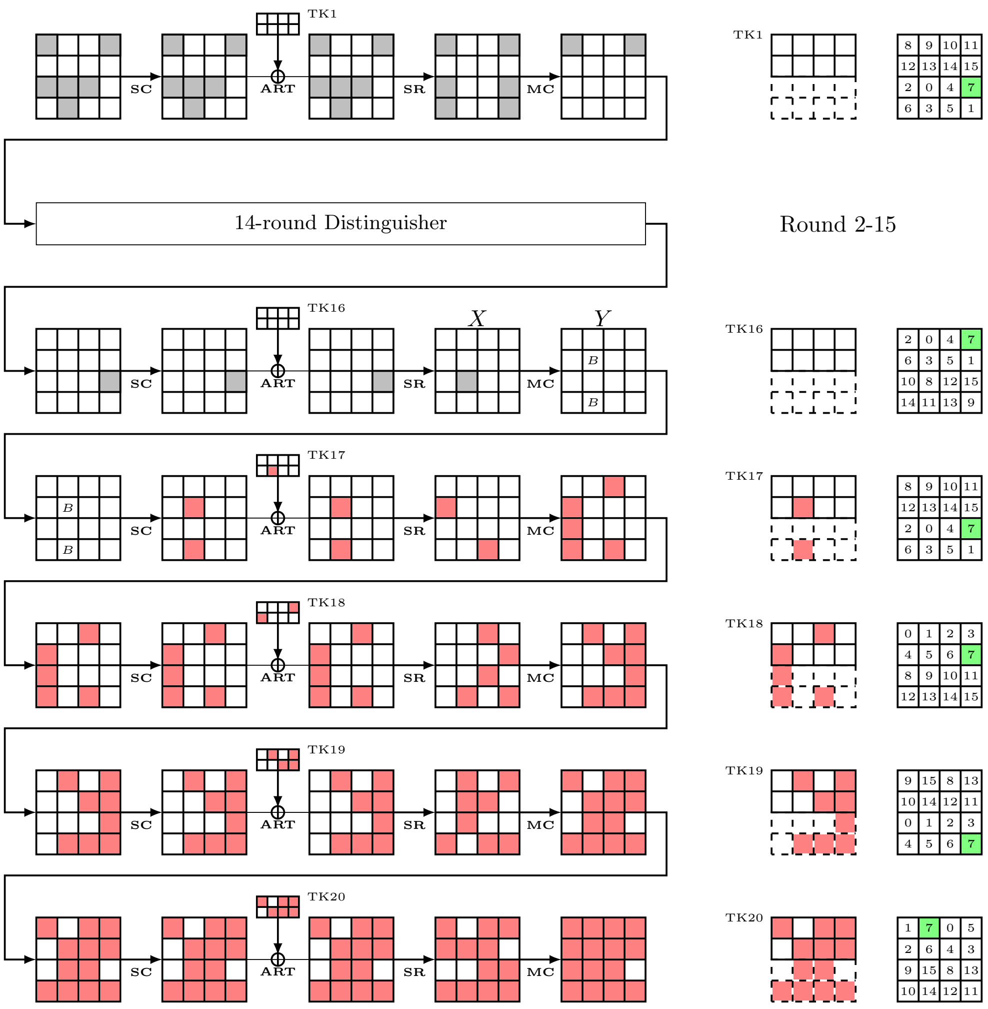


Figure 79: Key-Recovery Attacks on 20-rounds of Skinny-64/192.

In the key-recovery, we can prepend 1 round and append 8 rounds to the integral distinguisher. In total the attack reaches 23 rounds. Figure 79 shows the key-recovery, and let X_i, Y_i, and Z_i be the states as defined in Figure 77. Let P and T be the states of plaintext and tweak, respectively. We first prepare a set of chosen Z_1, where 14 cells are active, i.e., $Z_1[1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]$. Moreover, we need the tweak cell T[11] in all three lines of the tweak-schedule to be active, as it will propagate to cell T[7] after one round, which coincides with the beginning of both integral distinguishers as shown in Figure 78. Note that the consistent set of chosen plaintexts is computed from Z_1 without guessing any bits.

Table 24: Procedure for the key-recovery of $\bigoplus_{i=0}^{n} Z_{16}[5]$ on Skinny-64/192.

Step	Guessed Key	Data	Stored Texts	Memory (bits)	Complexity
0		2^{68}	$X_{23}[0,1,2,\dots,15], \Delta TK_{22}[0]$	2^{68}	2^{68}
1	$TK_{23}[0,\dots,7]$	2^{68}	$X_{22}[0,1,2,\dots,15], \Delta TK_{22}[0]$	$2^{68+32} = 2^{100}$	$2^{68+32} = 2^{100}$
2	$TK_{22}[0,\dots,7]$	2^{64}	$X_{21}[0,1,2,\dots,15]$	$2^{64+64} = 2^{128}$	$2^{68+64} = 2^{132}$
3	$TK_{21}[0,4]$	2^{56}	$X_{21}[1,2,3,5,6,7,9,10,11,13,14,15], Y_{20}[0,12]$	$2^{56+72} = 2^{128}$	$2^{64+72} = 2^{136}$
4	$TK_{21}[1,5]$	2^{48}	$X_{21}[2,3,6,7,10,11,14,15], Y_{20}[0,5,12,13]$	$2^{48+80} = 2^{128}$	$2^{56+80} = 2^{136}$
5	$TK_{21}[2,6]$	2^{48}	$X_{21}[3,7,11,15], Y_{20}[0,2,5,6,10,12,13,14]$	$2^{48+88} = 2^{136}$	$2^{48+88} = 2^{136}$
6	$TK_{21}[3,7]$	2^{44}	$Y_{20}[0,2,3,5,6,10,11,12,13,14,15]$	$2^{44+96} = 2^{140}$	$2^{48+96} = 2^{144}$
-		2^{44}	$X_{20}[0,2,3,4,5,8,9,12,13,14,15]$		
7	$TK_{20}[0,4]$	2^{36}	$X_{20}[2,3,5,9,13,14,15], Y_{19}[4,12]$	$2^{36+104} = 2^{140}$	$2^{44+104} = 2^{148}$
8	$TK_{20}[5]$	2^{36}	$X_{20}[2,3,14,15], Y_{19}[1,4,5,9,12]$	$2^{36+108} = 2^{144}$	$2^{36+108} = 2^{144}$
9	$TK_{20}[2]$	2^{32}	$X_{20}[3,15], Y_{19}[1,4,5,9,12,14]$	$2^{32+112} = 2^{140}$	$2^{36+112} = 2^{144}$
10	$TK_{20}[3]$	2^{28}	$Y_{19}[1,4,5,9,12,14,15]$	$2^{28+116} = 2^{144}$	$2^{32+116} = 2^{148}$
-		2^{28}	$X_{19}[1,4,7,11,12,13,15]$		
11	$TK_{19}[4]$	2^{24}	$X_{19}[1,7,11,13,15], Y_{18}[8]$	$2^{24+120} = 2^{144}$	$2^{28+120} = 2^{148}$
12	$TK_{19}[1]$	2^{20}	$X_{19}[7,11,15], Y_{18}[8,13]$	$2^{20+124} = 2^{144}$	$2^{24+124} = 2^{148}$
13	$TK_{19}[7]$	2^{12}	$Y_{18}[7,8,13]$	$2^{12+128} = 2^{140}$	$2^{20+128} = 2^{148}$
-		2^{12}	$X_{18}[6,10,14]$		
14	$TK_{18}[6]$	2^{4}	$Y_{17}[6]$	$2^{4+132} = 2^{140}$	$2^{12+132} = 2^{144}$
-		2^{4}	$X_{17}[5]$		
15	$TK_{17}[5]$	2^{4}	$Z_{16}[5]$	$2^{4+136} = 2^{140}$	$2^{4+136} = 2^{140}$

Due to the integral distinguisher s_9 in Y_{16} is balanced. Then

$$\begin{pmatrix} Z_{16}[1] \\ Z_{16}[5] \\ Z_{16}[9] \\ Z_{16}[13] \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} Y_{16}[1] \\ Y_{16}[5] \\ Y_{16}[9] \\ Y_{16}[13] \end{pmatrix} = \begin{pmatrix} Y_{16}[1] + Y_{16}[9] + Y_{16}[13] \\ Y_{16}[1] \\ Y_{16}[5] + Y_{16}[9] \\ Y_{16}[1] + Y_{16}[9] \end{pmatrix},$$

and

$$\bigoplus_{i=0}^{n} Z_{16}[5] + Z_{16}[13] = \bigoplus_{i=0}^{n} Y_{16}[9] = 0$$

Similarly to the attack against Qarma, we use the meet-in-the-middle technique for the integral attack [301], where $\bigoplus_{i=0}^{n} Z_{16}[5]$ and $\bigoplus_{i=0}^{n} Z_{16}[13]$ are independently evaluated, and round-tweakeys satisfying $\bigoplus_{i=0}^{n} Z_{16}[5] = \bigoplus_{i=0}^{n} Z_{16}[13]$ are recovered. The size of involved secret-tweakey is $148 (= 37 \times 4)$ bits. Since one structure removes incorrect secret-key guesses by a factor of 2^{-4}, we need $148/4 = 37$ structures to uniquely determine the secret-key.
To compute $\bigoplus_{i=0}^{n} Z_{16}[5]$, all cells labelled red and green are involved. We use the partial-sum technique to recover $\bigoplus_{i=0}^{n} Z_{16}[5]$, and the procedure is summarised in Table 24. Here, $\Delta TK_{22}[0]$ is computed from the relation of tweakeys.

Table 25: Procedure for the key-recovery of $\bigoplus_{i=0}^{n} Z_{16}[13]$ on SKINNY-64/192.

Step	Guessed Key	Data	Stored Texts	Memory (bits)	Complexity
0		2^{56}	$X_{23}[0,1,2,4,5,6,7,10,11,12,13,14,15], \Delta TK_{22}[0]$	2^{56}	2^{68}
1	$TK_{23}[0,1,2,4,5,6,7]$	2^{56}	$X_{22}[0,1,2,4,5,6,7,8,9,12,13,14,15], \Delta TK_{22}[0]$	$2^{56+28}=2^{84}$	$2^{56+28}=2^{84}$
2	$TK_{22}[0,1,2,4,5,6,7]$	2^{44}	$X_{21}[0,1,4,6,7,8,10,12,13,14,15]$	$2^{44+56}=2^{100}$	$2^{56+56}=2^{112}$
3	$TK_{21}[0,4]$	2^{36}	$X_{21}[1,6,7,10,13,14,15], Y_{20}[4,12]$	$2^{36+64}=2^{100}$	$2^{44+64}=2^{108}$
4	$TK_{21}[1]$	2^{32}	$X_{21}[6,7,10,14,15], Y_{20}[4,12,13]$	$2^{32+68}=2^{100}$	$2^{36+68}=2^{104}$
5	$TK_{21}[6]$	2^{28}	$X_{21}[7,15], Y_{20}[2,4,6,12,13]$	$2^{28+72}=2^{100}$	$2^{32+72}=2^{104}$
6	$TK_{21}[7]$	2^{24}	$Y_{20}[2,4,6,11,12,13]$	$2^{24+76}=2^{100}$	$2^{28+76}=2^{104}$
-		2^{24}	$X_{20}[2,5,7,9,13,14]$		
7	$TK_{20}[5]$	2^{16}	$X_{20}[2,7,14], Y_{19}[5]$	$2^{16+80}=2^{96}$	$2^{24+80}=2^{104}$
8	$TK_{20}[2]$	2^{12}	$X_{20}[7], Y_{19}[5,14]$	$2^{12+84}=2^{96}$	$2^{16+84}=2^{100}$
9	$TK_{20}[7]$	2^{12}	$Y_{19}[3,5,14]$	$2^{12+88}=2^{100}$	$2^{12+88}=2^{100}$
-		2^{12}	$X_{19}[3,4,15]$		
10	$TK_{19}[4]$	2^{12}	$X_{19}[3,15], Y_{18}[0]$	$2^{12+92}=2^{104}$	$2^{12+92}=2^{104}$
11	$TK_{19}[3]$	2^{8}	$Y_{18}[0,15]$	$2^{8+96}=2^{104}$	$2^{12+96}=2^{108}$
-		2^{8}	$X_{18}[0,12]$		
12	$TK_{18}[0]$	2^{4}	$Y_{17}[12]=X_{17}[13]=Z_{16}[13]$	$2^{4+100}=2^{104}$	$2^{8+100}=2^{108}$

Then, the time complexity can be computed by

$$2^{68} + 2^{100} + 2^{132} + 3 \times 2^{136} + 2^{140} + 4 \times 2^{144} + 5 \times 2^{148} \approx 2^{150.4}$$

We also need to compute $\bigoplus_{i=0}^{n} Z_{16}[13]$, but the time complexity is clearly negligible compared with that for $\bigoplus_{i=0}^{n} Z_{16}[5]$. Nevertheless, the procedure to recover $\bigoplus_{i=0}^{n} Z_{16}[13]$ is shown in Table 25, where we consider all cells labelled blue and green. Again, we use the partial-sum technique to recover $\bigoplus_{i=0}^{n} Z_{16}[13]$, and the time complexity can be computed by

$$2^{68} + 2^{84} + 2^{112} + 3 \times 2^{108} + 5 \times 2^{104} + 2 \times 2^{100} \approx 2^{112.3}$$

We need to repeat these two procedures for 37 times to recover the secret-key. Thus, the total time complexity can be computed by

$$37 \times (2^{150.4} + 2^{112.3}) \approx 2^{155.6}$$

the data complexity is $37 \times 2^{68} \approx 2^{73.2}$, and the memory complexity is $1/64 \cdot 2^{144} = 2^{138}$ 64-bit blocks. Finally, we compare these lists and find a match. Similarly to the attack against SKINNY-64/128, our attack requires a data complexity above 2^{64}, however, we do not need to collect the full-codebook under a fixed tweakey.

5.7 CONCLUSION

In this chapter, we study zero-correlation attacks on tweakable block ciphers and consider for the first time the effect of the tweak. Kranz, Leander and Wiemer

showed that the addition of the tweak, that is updated by a linear key schedule, does not introduce new linear characteristics, which is quite different to the differential model. However, given additional restrictions from the linear tweak schedule, allow us to efficiently calculate the linear hull for tweakable block ciphers. As zero-correlation cryptanalysis requires huge amounts of data, we turn the distinguishers into integral distinguisher and then mount key-recovery attacks based on them. Distinguishers found this way cover more rounds compared to existing results utilising the division property, zero-correlation or other conventional approaches to find integral distinguishers.
We show new attacks on Qarma, Mantis and Skinny, where the attack on Qarma is currently the best attack (with respect to number of rounds) on a round-reduced variant of Qarma. This new way of searching for distinguishers on tweakable block ciphers does not only allow attackers to find longer distinguishers, but also provides designers of tweakable block ciphers with new insights. For example in tweakable reflection ciphers like Mantis or Qarma, where the tweak is added just in the forward and backward rounds, while in the middle rounds just round-keys are added, the additional middle rounds do not provide extra security with respect to our attacks. This is because the zero-correlation linear hulls over the tweaks are independent of the number of keyed middle rounds.

6 CRYPTANALYSIS OF THE TWEAKABLE BLOCK CIPHER SKINNY-64/128

CONTENTS

EXECUTIVE SUMMARY. In this chapter, we present a related-tweakey impossible-differential attack on a round-reduced version of the tweakable block cipher Skinny based on [20]. Furthermore, we present some unpublished results on integral distinguisher of Skinny.
At CRYPTO'16, Beierle *et al.* presented Skinny, a family of lightweight tweakable block ciphers intended to offer an alternative to the NSA designs Simon and Speck. Skinny can be implemented efficiently in both software and hardware and supports block sizes of 64 and 128 bits as well as tweakey sizes of 64, 128, 192 and 128, 256, 384 bits respectively. In this chapter, we present a related-tweakey impossible-differential attack on up to 23 (out of 36) rounds of Skinny-64/128 for different tweak sizes. All our attacks can be trivially extended to Skinny-128/128.

DECLARATION OF AUTHORSHIP. The work described in this chapter is based on the paper [20]: *Related-Key Impossible-Differential Attack on Reduced-Round* Skinny and was presented at *The* 15th *International Conference on Applied Cryptography and Network Security (ACNS'17)* in Kanazawa, Japan. The paper is joint work with Subhadeep Banik, Avik Chakraborti, Eik List, Florian Mendel, Siang Meng Sim, and Gaoli Wang. The work was initiated during the group sessions of the 6th *Asian Workshop on Symmetric Cryptography (ASK'16)* in Nagoya, Japan. All authors contributed equally to the results of the paper. The contributions of the author are the following:

- Searching for integral and impossible differential distinguishers of Skinny.
- Improving the key-recovery attack by suggesting to insert the differences in the tweak.
- Further improvements of the key-recovery attacks to cover more rounds and reduce the complexities.

6.1 INTRODUCTION

Skinny is a family of lightweight tweakable block ciphers proposed at CRYPTO'16 by Beierle *et al.* [50]. The goal of the designers was to design a cipher that could be implemented highly efficient on both software and hardware platforms, with performance comparable or better than the Simon and Speck [46] families of block ciphers. Like the NSA designs Simon and Speck, Skinny supports a wide range of block sizes and tweak/key sizes – however, in contrast to the And-RX and Add-RX based NSA proposals, Skinny is based on the better understood Substitution-Permutation-Network approach.
Skinny offers a large security margin within the number of rounds for each member of the Skinny family. The designers showed in their initial analysis, which they published alongside with the specification of Skinny, that their attacks reach close to half of the number of rounds of the cipher. To motivate third-party cryptanalysis, the designers of Skinny announced a cryptanalysis competition [49] for Skinny-64/128 and Skinny-128/128 with the obvious challenge of attacking more rounds than the preliminary analysis, concerning both the single- and related-key models.

RELATED WORK. Independent of our analysis Liu *et al.* [230] analysed Skinny in the related-tweakey model, showing impossible-differential and rectangle attacks on 19, 23, and 27 rounds of Skinny-n/n, Skinny-n/2n and Skinny-n/3n, respectively. In [328], Tolba *et al.* showed impossible-differential attacks for 18, 20, 22 rounds of Skinny-n/n, Skinny-n/2n and Skinny-n/3n, respectively. Additionally, Sadeghi *et al.* [295] studied related-tweakey impossible- differential and zero-correlation linear attacks. In comparison to the other attacks, our 23-round related-tweakey impossible-differential attack on Skinny-64/128 has the lowest time complexity so far. Table 26 summarises our attacks and compares them to existing attacks on Skinny.

6.2 DESCRIPTION OF SKINNY

A detailed description of the tweakable block cipher Skinny can be found in Chapter 5.6. In the following, we will give a short description of the tweakey schedule of Skinny, as the attack described in Section 6.3.2 exploits parts of the tweakey schedule.

Table 26: Summary of our attacks and comparison to existing cryptanalysis of Skinny.

Instance	Rounds	Attack Type	Time	Data	Memory	Ref
Skinny-64/*	14	Integral	2^{53}	2^{48}	2^{32}	[50]
Skinny-64/64	18	Impossible	$2^{57.1}$	$2^{47.5}$	$2^{58.5}$	[328]
Skinny-64/64	19	Impossible	$2^{63.0}$	$2^{61.4}$	2^{56}	[230]
Skinny-64/128	20	Impossible	$2^{121.1}$	$2^{47.7}$	$2^{74.7}$	[328]
Skinny-64/128	20	ZC/Integral	$2^{97.5}$	$2^{68.4}$	2^{82}	Chap. 5.6
Skinny-64/128	21	Impossible	$2^{71.4}$	$2^{71.4*}$	$2^{68.0}$	Sect. 6.3.2
Skinny-64/128	22	Impossible†	$2^{71.6}$	$2^{71.4*}$	$2^{64.0}$	Sect. 6.3.3
Skinny-64/128	23	Impossible	$2^{125.9}$	$2^{62.5}$	$2^{124.0}$	[230]
Skinny-64/192	22	Impossible	$2^{184.0}$	$2^{47.8}$	$2^{74.8}$	[328]
Skinny-64/192	23	ZC/Integral	$2^{155.6}$	$2^{73.2}$	2^{138}	Chap. 5.6
Skinny-64/128	23	Impossible†	2^{79}	$2^{71.4*}$	$2^{64.0}$	Sect. 6.3.4
Skinny-64/192	27	Rectangle	$2^{165.5}$	$2^{63.5}$	$2^{80.0}$	[230]
Skinny-128/128	18	Impossible	$2^{116.9}$	$2^{92.4}$	$2^{115.4}$	[328]
Skinny-128/128	19	Impossible	$2^{124.6}$	$2^{122.5}$	$2^{112.0}$	[230]
Skinny-128/256	20	Impossible	$2^{245.7}$	$2^{92.1}$	$2^{147.1}$	[328]
Skinny-128/256	23	Impossible	$2^{251.5}$	$2^{124.5}$	$2^{248.0}$	[230]
Skinny-128/384	22	Impossible	$2^{373.5}$	$2^{92.2}$	$2^{147.2}$	[328]
Skinny-128/384	27	Rectangle	$2^{331.0}$	$2^{112.0}$	$2^{144.0}$	[230]

TWEAKEY SCHEDULE. The tweakey schedule of Skinny, as illustrated in Figure 80, follows the Tweakey framework [188]. In contrast to the previous Tweakey designs Deoxys-BC [189] and Joltik-BC [186], Skinny employs a significantly more lightweight strategy. In each round, only the two topmost rows of each tweakey word are extracted and XORed to the state. An additional round-dependent constant is also XORed to the state to prevent against slide attacks.
The 128-bit tweakey is arranged in two 64-bit tweakey words, represented by TK_1 and TK_2. In each round, the tweakey words are updated by a cell permutation P_T that ensures that the two bottom rows of a tweakey word in a certain round are

* The data complexity of our 21-round attack is beyond codebook. Our attack is more efficient than a full codebook attack in the case where Skinny is used in a tweak-updating mode (*i.e.*, where the tweak changes every time, but the key stays the same). This does not effect the 22/23 round attack as 48 bits of the tweakey are public (i.e. data complexity for full codebook would be $2^{64} \times 2^{48} = 2^{112}$, where 2^{64} comes from the state and 2^{48} comes from the tweak).

† Our attack on 22/23 rounds uses the tweak against the recommendation of the Skinny designers but still conform to the specification in [50].

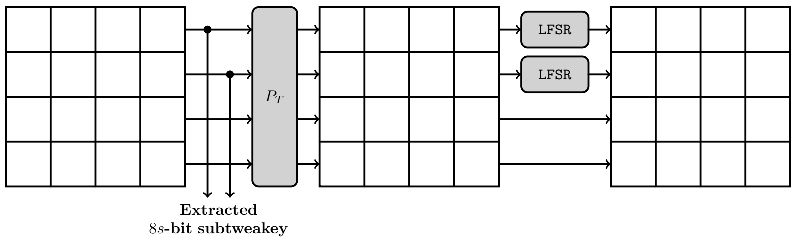


Figure 80: Tweakey schedule of SKINNY.

exchanged with the two top rows in the tweakey word in the subsequent round. The permutation is given as:

$$P_T = \{9, 15, 8, 13, 10, 14, 12, 11, 0, 1, 2, 3, 4, 5, 6, 7\}$$

The permutation P_T has a period of 16, as visualised in Figure 81. Moreover, each individual cell in the two topmost rows of TK_2 is transformed by a 4-bit LFSR to minimise the cancellation of differences from TK_1 and TK_2; TK_1 employs no LFSR transformation. The LFSR transformation L is given by

$$L(x_3, x_2, x_1, x_0) := (x_2, x_1, x_0, x_3 \oplus x_2),$$

where x_3, x_2, x_1, x_0 represent the individual bits of every tweakey nibble. The update equation for the tweak cells can also be written explicitly as:

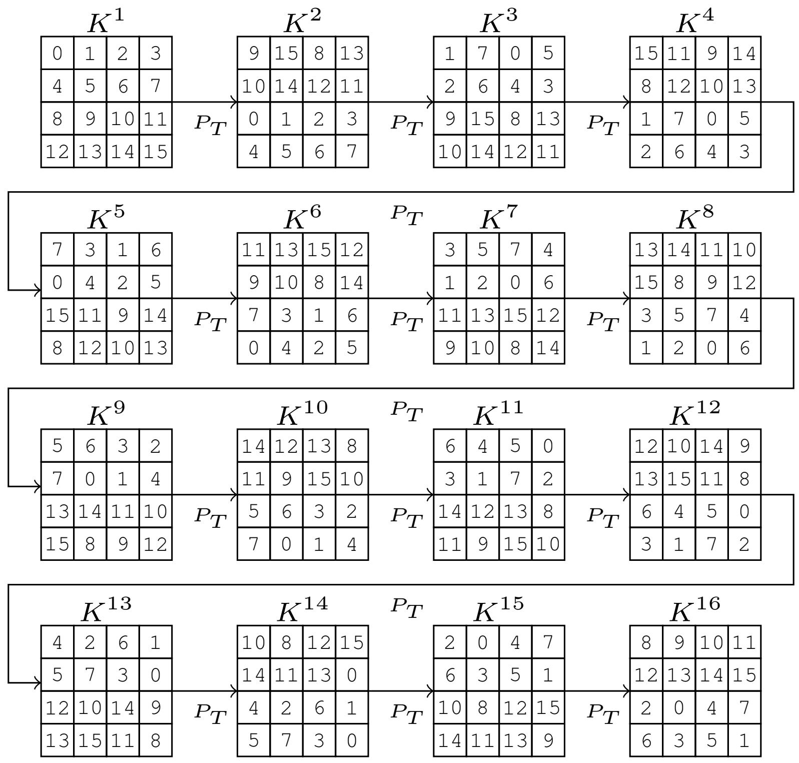
$$TK_1^{r+1}[i] = \begin{cases} TK_1^r[P[i]] & \text{for } 0 \leqslant i \leqslant 15, \end{cases}$$
$$TK_2^{r+1}[i] = \begin{cases} L(TK_2^r[P[i]]) & \text{if } 0 \leqslant i \leqslant 7, \\ TK_2^r[P[i]] & \text{otherwise.} \end{cases}$$

where $TK_a^r[i]$ represents the i^{th} nibble of TK_a $(a = 1, 2)$ in round r. Note that the r^{th}-round tweakey is given by $K^r = TK_1^r[i] \oplus TK_2^r[i]$, for $0 \leqslant i \leqslant 7$.

6.3 RELATED-KEY IMPOSSIBLE-DIFFERENTIAL ATTACK

Impossible-differential attacks were introduced independently by Biham *et al.* [69, 70] and Knudsen [200], and they are widely used as an important cryptanalytic technique. The attack starts with finding an input difference that can never result in an output difference. By adding rounds before and/or after the impossible differential, one can collect pairs with certain plaintext and ciphertext differences. If there exists a pair that meets the input and output values of the impossible differential under some subkey, these subkeys must be wrong. In this way, we filter as many wrong keys as possible and exhaustively search the rest of the keys. More details on impossible differential attacks can be found in Chapter 2.4.3.

NOTATIONS. Let us state a few notations that are used in the attack description:



K^1: 0 1 2 3 / 4 5 6 7 / 8 9 10 11 / 12 13 14 15
K^2: 9 15 8 13 / 10 14 12 11 / 0 1 2 3 / 4 5 6 7
K^3: 1 7 0 5 / 2 6 4 3 / 9 15 8 13 / 10 14 12 11
K^4: 15 11 9 14 / 8 12 10 13 / 1 7 0 5 / 2 6 4 3
K^5: 7 3 1 6 / 0 4 2 5 / 15 11 9 14 / 8 12 10 13
K^6: 11 13 15 12 / 9 10 8 14 / 7 3 1 6 / 0 4 2 5
K^7: 3 5 7 4 / 1 2 0 6 / 11 13 15 12 / 9 10 8 14
K^8: 13 14 11 10 / 15 8 9 12 / 3 5 7 4 / 1 2 0 6
K^9: 5 6 3 2 / 7 0 1 4 / 13 14 11 10 / 15 8 9 12
K^{10}: 14 12 13 8 / 11 9 15 10 / 5 6 3 2 / 7 0 1 4
K^{11}: 6 4 5 0 / 3 1 7 2 / 14 12 13 8 / 11 9 15 10
K^{12}: 12 10 14 9 / 13 15 11 8 / 6 4 5 0 / 3 1 7 2
K^{13}: 4 2 6 1 / 5 7 3 0 / 12 10 14 9 / 13 15 11 8
K^{14}: 10 8 12 15 / 14 11 13 0 / 4 2 6 1 / 5 7 3 0
K^{15}: 2 0 4 7 / 6 3 5 1 / 10 8 12 15 / 14 11 13 9
K^{16}: 8 9 10 11 / 12 13 14 15 / 2 0 4 7 / 6 3 5 1
P_T

Figure 81: The permutation P_T in the tweakey schedule has a period of 16.

K^r represents the r^{th} round key. This is equal to $TK_1^r \oplus TK_2^r$. Similarly, $k^r[i] = tk_1^r[i] \oplus tk_2^r[i]$ represents the individual i^{th} tweakey nibble in round r.

A^r represents the internal state before SC in round r.

B^r represents the internal state after SC in round r.

C^r represents the internal state after AT in round r.

D^r represents the internal state after SR in round r.

E^r represents the internal state after MC in round r. Furthermore, $E^r = A^{r+1}$.

L^t represents the t-times composition of LFSR function L.

$\overline{X}$ represents the corresponding variable X in the related-key setting.

$X[i]$ represents the i^{th} nibble of the corresponding variable X.

6.3.1 Related-Tweakey Impossible-Differential Distinguisher

Figure 82 presents the 11-round related-key differential trail that we use. While the first impossible differential distinguishers we constructed where found by hand, we then used a dedicated automated tool that was adapted from [244]. With the automated tool, we basically tried every possible position to add a starting difference

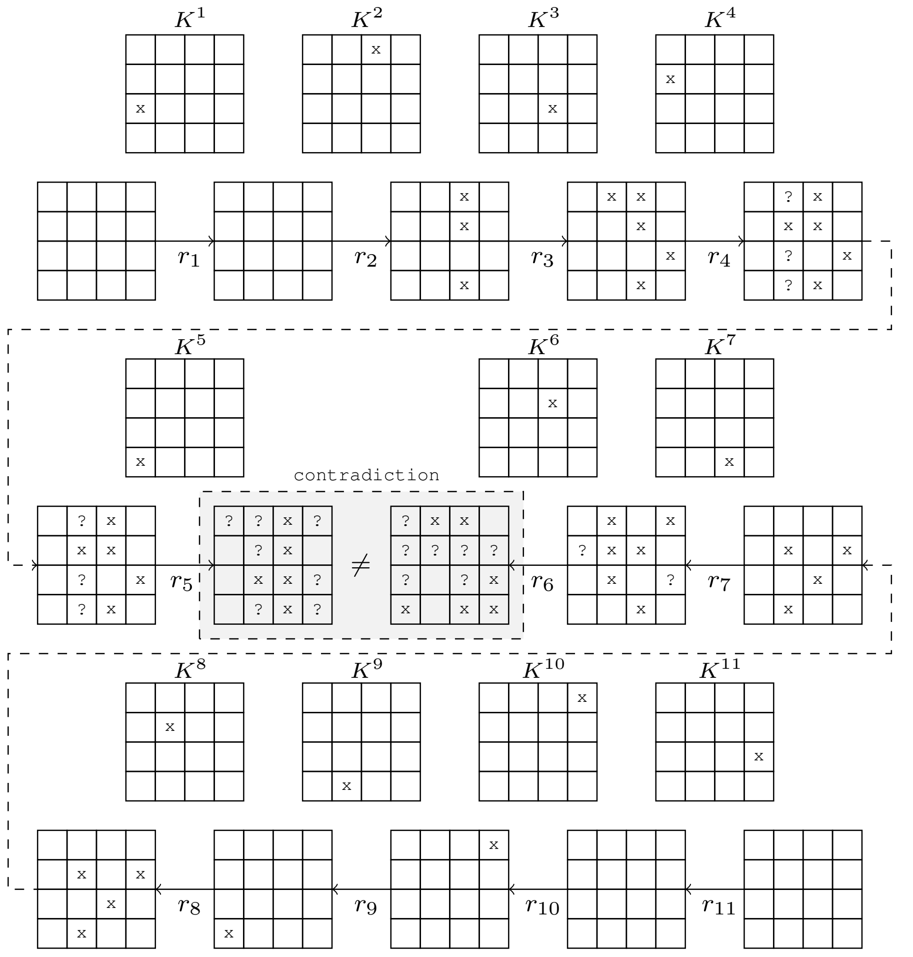


Figure 82: Related-key impossible-differential trail over 11 rounds of SKINNY-64/128.

in the tweakey, and then we selected the trails that reached the most rounds with a valid contradiction in the middle.
For the best trail we found, we introduce a difference in the cell at position 8 of the combined tweakey. Since the initial difference is in Cell 8, *i.e.*, in one of the bottom two rows of the tweakey, the state is not affected in the first round. Therefore, the first time the difference will be added to the state starts from the second round onwards. Similarly in the backward trail, the difference in the 11^{th} round-tweakey appears in Cell 11 (in a bottom row), due to which we get an extra round in the backward direction.

Lemma 1. The equation $S(x \oplus \Delta_{in}) \oplus S(x) = \Delta_{out}$ has one solution x on average for $\Delta_{in}, \Delta_{out} \neq 0$. Similar result holds for the inverse S-Box S^{-1}.

Proof. The above fact can be deduced by analysing the Differential-Distribution Table (DDT) of the S-box S as illustrated in Table 27. The average can be calculated as $\frac{1}{225} \cdot \sum_{\delta_{in}, \delta_{out} \neq 0} \mathrm{DDT}(\Delta_{in} = \delta_{in}, \Delta_{out} = \delta_{out}) \approx 1$. This also holds for the inverse S-box yielding the same result.

Lemma 2. For random values of x and $\Delta_{in}, \Delta_{out} \neq 0$, the equation $S(x \oplus \Delta_{in}) \oplus S(x) = \Delta_{out}$ holds with probability around 2^{-4}.

Proof. The above fact can also be deduced by analysing the Differential-Distribution Table (DDT) of the S-box S as illustrated in Table 27. The probability can be calculated as:

$$\begin{aligned} \Pr[(x, \Delta_{in}, \Delta_{out})] &= \sum_{\delta_{in}, \delta_{out} \neq 0} \Pr[(x, \delta_{in}, \delta_{out}) | \Delta_{in} = \delta_{in}, \Delta_{out} = \delta_{out}] \\ &\qquad \cdot \Pr[\Delta_{in} = \delta_{in}, \Delta_{out} = \delta_{out}] \\ &= \frac{1}{225} \cdot \sum_{\delta_{in}, \delta_{out} \neq 0} \mathrm{DDT}(\Delta_{in} = \delta_{in}, \Delta_{out} = \delta_{out}) \cdot 2^{-4} \approx 2^{-4} \end{aligned}$$

where $\Pr[(x, \delta_{in}, \delta_{out})]$ denotes the probability that the equation is satisfied for the triplet $x, \delta_{in}, \delta_{out}$.

6.3.2 21-Round Attack on Skinny-64/128

The impossible differential trail described in Figure 82 can be extended by six rounds in backward and four rounds in forward direction as shown in the following two Lemmas.

Lemma 3. It is possible to find plaintext pairs $P, \overline{P}$ and related-tweakey pairs $K, \overline{K}$ such that if the tweakey pairs differ only in nibble position 11, then there is no difference in the internal state after executing six rounds of Skinny-64/128 with the plaintext-tweakey pairs (P, K) and $(\overline{P}, \overline{K})$.

Proof. We will show how the required plaintext and tweakey pairs are generated. We choose the nibble at Position 11 to introduce the initial difference because after completing six rounds, the difference is shuffled to Cell 8 of the round key, which coincides with the beginning of the impossible- differential trail, shown in Figure 82. It can be seen that the AddRoundTweakey in the first round can be pushed behind the MixColumns operation by changing the first round key to $\mathrm{Lin}(K_1)$ where $\mathrm{Lin} = \mathrm{MC} \circ \mathrm{SR}$ represents the linear layer (refer to Figure 83).

$$\mathrm{Lin}(K^1) = \begin{bmatrix} k^1[0] & k^1[1] & k^1[2] & k^1[3] \\ k^1[0] & k^1[1] & k^1[2] & k^1[3] \\ k^1[7] & k^1[4] & k^1[5] & k^1[6] \\ k^1[0] & k^1[1] & k^1[2] & k^1[3] \end{bmatrix}$$

Furthermore, the initial difference between $K = TK_1^1 \oplus TK_2^1$ and $\overline{K} = \overline{TK_1^1} \oplus \overline{TK_2^1}$ can be selected in a specific form, so that in Round 6, the tweakey difference is zero.

Table 27: Difference-Distribution Table

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	16	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
1	.	.	.	.	.	.	.	.	4	4	4	4	.	.	.	.
2	.	4	.	4	.	4	4	.	.	.	.	.	.	.	.	.
3	.	.	.	.	.	.	.	.	2	2	2	2	2	2	2	2
4	.	.	4	.	.	.	2	2	.	.	.	4	2	2	.	.
5	.	.	4	.	.	.	2	2	.	.	4	.	2	2	.	.
6	.	2	.	2	2	.	.	2	2	.	2	.	.	2	2	.
7	.	2	.	2	2	.	.	2	.	2	.	2	2	.	.	2
8	.	.	.	.	4	4	.	.	.	.	.	.	2	2	2	2
9	.	.	.	.	4	4	.	.	.	.	.	.	2	2	2	2
a	.	.	.	.	.	4	4	.	2	2	2	2	.	.	.	.
b	.	4	.	4	.	.	.	.	.	.	.	.	2	2	2	2
c	.	.	4	.	.	.	2	2	4	.	.	.	.	.	2	2
d	.	.	4	.	.	.	2	2	.	4	.	.	.	.	2	2
e	.	2	.	2	2	.	.	2	.	2	.	2	.	2	2	.
f	.	2	.	2	2	.	.	2	2	.	2	.	2	.	.	2

Let us denote $\delta_1 = tk_1^1[11] \oplus \overline{tk_1^1}[11]$ and $\delta_2 = tk_2^1[11] \oplus \overline{tk_2^1}[11]$. In Round 6, the difference will appear in Cell 0 of the round key and so we want:

$$\begin{aligned} k^6[0] \oplus \overline{k^6}[0] &= tk_1^6[0] \oplus \overline{tk_1^6}[0] + tk_2^6[0] \oplus \overline{tk_2^6}[0] \\ &= tk_1^1[11] \oplus \overline{tk_1^1}[11] \oplus L^3\left(tk_2^1[11]\right) \oplus L^3\left(\overline{tk_2^1}[11]\right) \\ &= \delta_1 \oplus L^3\left(\delta_2\right) = 0 \end{aligned}$$

If the attacker chooses differences δ_1, δ_2 satisfying the equation $\delta_1 \oplus L^3(\delta_2) = 0$, then there is no difference introduced via the round-key addition in Round 6. The attacker should therefore follow the steps:

1. Take any Plaintext P and compute the state after the first round MixColumns, *i.e.,* E^1.

2. Take any three-nibble difference $\Delta_1, \Delta_3, \Delta_4$ to construct $\overline{E^1}$ such that

$$E^1 \oplus \overline{E^1} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \Delta_1 & 0 & \Delta_2 \\ \Delta_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & \Delta_4 \end{bmatrix}$$

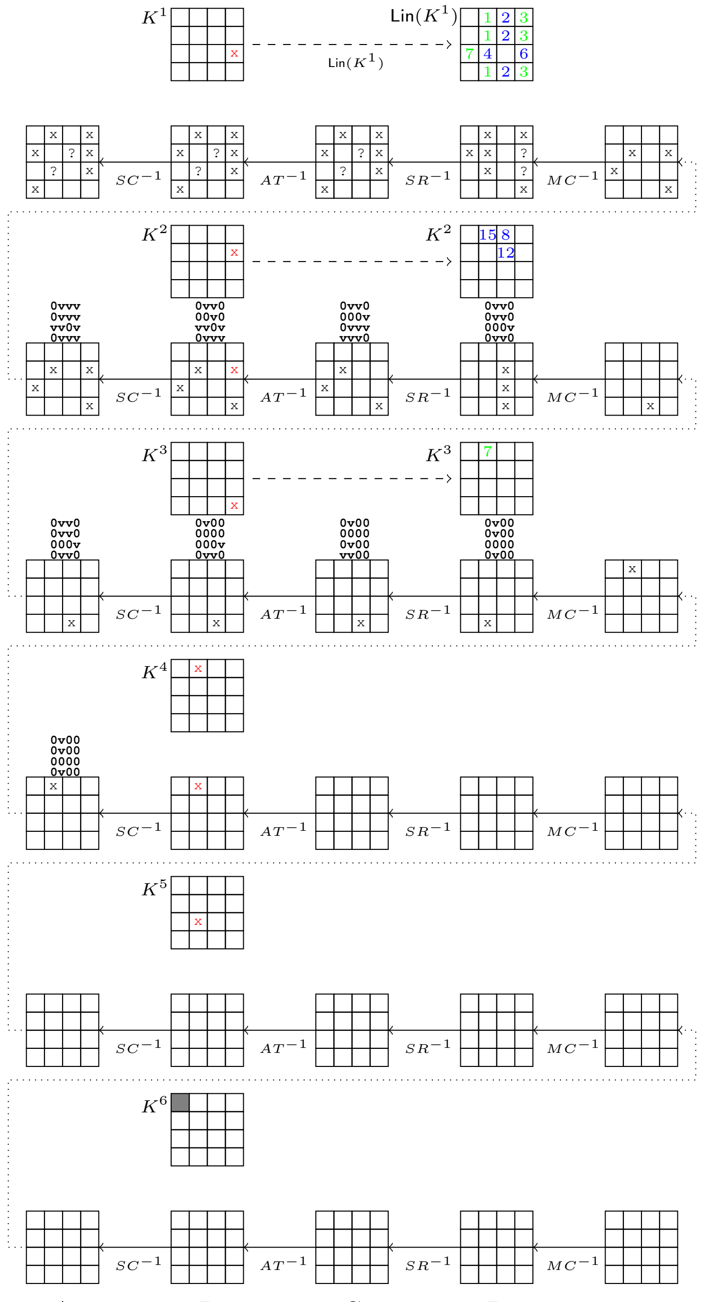


Figure 83: Trail for the six forward rounds (the values of active nibbles in red are functions of δ_1, δ_2, the dark-gray cell visualises the tweakey cancellation).

The value of Δ_2 will be determined in Equation 57. The attacker can recover $\overline{\mathsf{P}}$ by inverting the MC, SR, AC and SC layers on $\overline{\mathsf{E}^1}$.

3. The attacker chooses the difference α in Cell 14 of E^2, and calculates then $k^1[1]$, $k^1[3]$, $k^1[7]$ so that

$$B^2 \oplus \overline{B^2} = \mathrm{Lin}^{-1}(E^2) \oplus \mathrm{Lin}^{-1}(\overline{E^2}) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \alpha & 0 & \beta \\ \alpha & 0 & 0 & 0 \\ 0 & 0 & 0 & \alpha \end{bmatrix}.$$

   For example, $k^1[1]$ is a solution of the equation:

$$S\left(E^1[5] \oplus k^1[1]\right) \oplus S\left(E^1[5] \oplus \Delta_1 \oplus k^1[1]\right) = \alpha.$$

   Lemma 1 ensures that the equation above has one solution on average.

4. β needs to be equal to $k^2[7] \oplus \overline{k^2}[7] = tk_1^2[7] \oplus tk_2^2[7] \oplus \overline{tk_1^2}[7] \oplus \overline{tk_2^2}[7]$. This is equal to $tk_1^1[11] \oplus L(tk_2^1[11]) \oplus \overline{tk_1^1}[11] \oplus L(\overline{tk_2^1}[11]) = \delta_1 \oplus L(\delta_2)$. Further, the attacker chooses δ_1 and δ_2 satisfying $\delta_1 \oplus L^3(\delta_2) = 0$ and calculates $\beta = \delta_1 \oplus L(\delta_2)$. Δ_2 can then be determined as a solution of the equation:

$$S\left(E^1[7] \oplus k^1[3]\right) \oplus S\left(E^1[7] \oplus \Delta_2 \oplus k^1[3]\right) = \beta \tag{57}$$

   The attacker now has the values of Δ_1, Δ_2, Δ_3, Δ_4 and can compute $E^1, \overline{E^1}$ and hence $P, \overline{P}$.

5. However, the attacker still needs that in Round 4, the active nibble in $B^4[1]$ is equal to $\delta_1 \oplus L^2(\delta_2)$ to make all the state cells inactive in C^4, D^4, and E^4.

6. The attacker needs to guess three round-key values in Round 1 (*i.e.,* $k^1[2]$, $k^1[4]$, $k^1[6]$) and three round-key values in Round 2 (*i.e.,* $k^2[1] = tk_1^1[15] \oplus L(tk_2^1[15])$, $k^2[2] = tk_1^1[8] \oplus L(tk_2^1[8])$, $k^2[6] = tk_1^1[12] \oplus L(tk_2^1[12])$).

   If the attacker can guess these values, then the actual values (marked with v) of the state cells are known for the plaintext pair $P, \overline{P}$, as opposed to only differences (marked by 0) in both Figure 83 and Figure 84.

7. Guessing the tweakey nibbles mentioned above enables the attacker to calculate the value of $B^3[1]$. Then, $k^3[1] = tk_1^1[7] \oplus L(tk_2^1[7])$ are calculated as follows. Since $D^3[1] = B^3[1] \oplus k^3[1]$ holds, we have:

$$S\left(D^3[1] \oplus D^3[9] \oplus D^3[13]\right) \oplus S\left(D^3[1] \oplus D^3[9] \oplus \overline{D^3}[13]\right) = \delta_1 \oplus L^2(\delta_2).$$

   The knowledge of the above guessed key nibbles allows the attacker to calculate $D^3[9]$, $D^3[13]$, and $\overline{D^3}[13]$, as the attacker can calculated $k^3[1] = tk_1^1[7] \oplus L(tk_2^1[7])$, which is the solution to the equation above. Again, Lemma 1 guarantees one solution on average. Since the attacker has already determined $k^1[7] = tk_1^1[7] \oplus tk_2^1[7]$, this also determines the values of $tk_1^1[7]$ and $tk_2^1[7]$.

8. This guarantees that there are no more active nibbles after Round 4. The key difference does not add to the state in Round 5, and due to the fact that $\delta_1 \oplus L^3(\delta_2) = 0$, the tweak difference becomes 0 in Round 6.

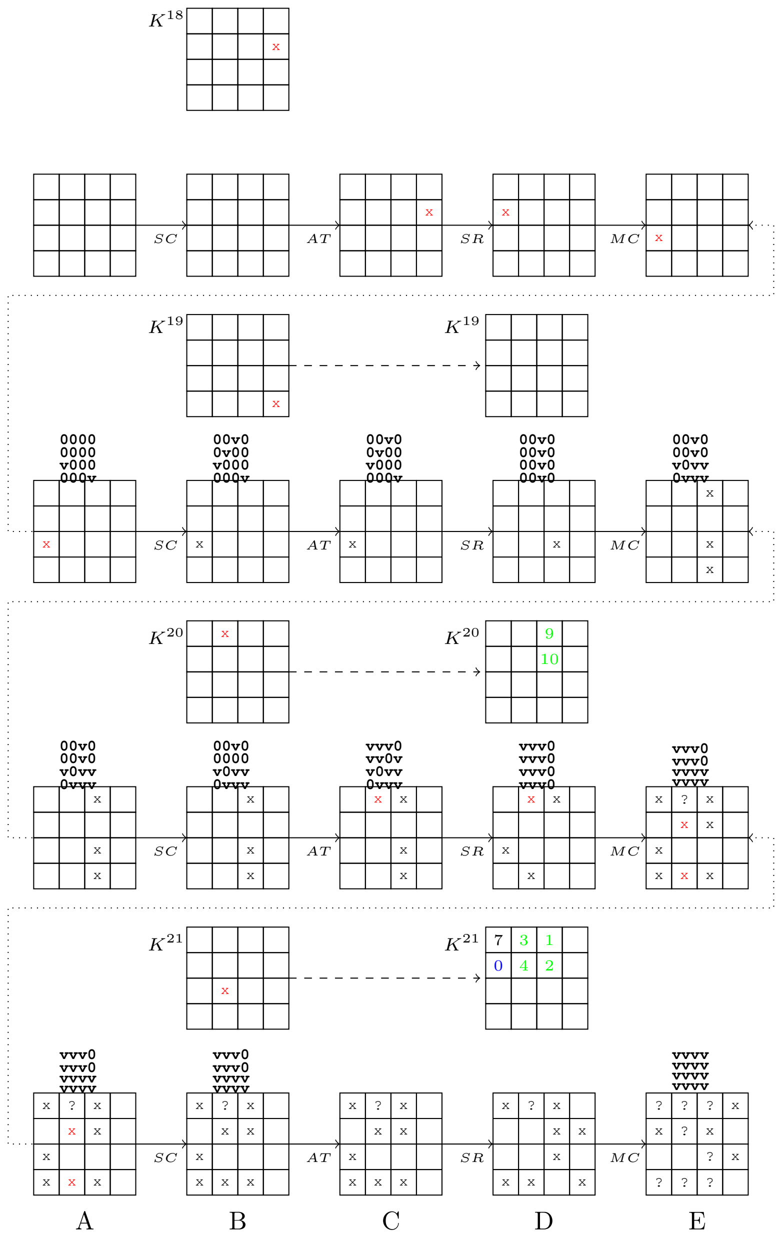


Figure 84: Trail for the four backward rounds (the values of active nibbles in red are functions of δ_1 and δ_2).

Thus, by guessing six and calculating three key nibbles, we can construct P, $\overline{P}$ and K, $\overline{K}$ so that the internal state after six rounds has no active nibbles.

Lemma 4. Let C, $\overline{C}$ be two ciphertexts after querying plaintext-tweakey pairs (P, K) and ($\overline{P}$, $\overline{K}$) to a 21-round Skinny-64/128 encryption oracle. Then for a fraction of 2^{-40} ciphertext pairs, it is possible to construct a backward trail for Round 21 to

Round 18 by guessing intermediate tweakey nibbles so that there are no active nibbles in the internal state at the end of Round 17.

Proof. The attacker starts working backward from the ciphertext pairs $C, \overline{C}$ and proceeds as follows (illustrated in Figure 84):

1. The attacker rejects ciphertext pairs that do not have seven inactive cells in Cells 3, 4, 5, 8, 9, 11, and 14) after peeling off the final MixColumns layer (*i.e.,* D^{21}). Thus, a fraction of 2^{-28} pairs are filtered after this stage.

2. Furthermore, the attacker rejects ciphertext pairs that do not have the difference $\delta_1 \oplus L^{10}(\delta_2)$ in Cell 13 of A^{21}, *i.e.,* reject if $A^{21}[13] \oplus \overline{A^{21}}[13] \neq \delta_1 \oplus L^{10}(\delta_2)$. Since calculating this cell does not require any key guess, the attacker can do this filtering instantly leaving a fraction of 2^{-4} pairs after this stage.

3. Since the bottommost two rows of the state are not affected by the tweakey addition, and since $tk_1^1[7], tk_2^1[7]$ are already known, the attacker can calculate the actual values in Cells 0, 8, and 12 in A^{21} for the ciphertext pairs. These have to be equal since they are the output of the 20th-round MixColumns operation on the leftmost column which had only one active nibble in its input. If the active Cells 8 and 12 are different, the attacker can reject the pair. This adds another filter with probability 2^{-4}.

4. Since the actual values in Cell 0 in A^{21} for the ciphertext pairs were already calculated in the previous step, the attacker checks if the value of the active Cell 0 is equal to that of Cells 8 and 12, and rejects the pair otherwise. This adds another filter of probability 2^{-4}.

5. The attacker determines $k^{21}[5] = tk_1^1[4] \oplus L^{10}(tk_2^1[4])$ so that the active nibble in Cell 5 of A^{21} is $\delta_1 \oplus L^{10}(\delta_2)$. Since $A^{21}[5] = S^{-1}\left(k^{21}[5] \oplus C^{21}[5]\right)$, $k^{21}[5]$ is a solution to the equation below:
$$S^{-1}\left(k^{21}[5] \oplus C^{21}[5]\right) \oplus S^{-1}\left(k^{21}[5] \oplus \overline{C^{21}}[5]\right) = \delta_1 \oplus L^{10}(\delta_2).$$

6. The attacker determines $k^{21}[2] = tk_1^1[1] \oplus L^{10}(tk_2^1[1])$ and $k^{21}[6] = tk_1^1[2] \oplus L^{10}(tk_2^1[2])$ so that the active nibble in Cell 2 and 6 of A^{21} are equal to the active nibble in Cell 14. Again, this works since those cells are output of the 20th-round MixColumns operation on Column 2 which had only one active nibble in its input.

7. Additionally, the attacker guesses $k^{21}[4] = tk_1^1[0] \oplus L^{10}(tk_2^1[0])$. This enables the attacker to compute the actual values for the entire leftmost column of A^{21} and hence to compute the leftmost column of D^{20}.

8. The value of the active nibble in cell 10 of A^{20} is given as:
$$\begin{aligned} A^{20}[10] \oplus \overline{A^{20}}[10] &= S^{-1}\left(B^{20}[10]\right) \oplus S^{-1}\left(\overline{B^{20}}[10]\right) \\ &= S^{-1}\left(D^{20}[8]\right) \oplus S^{-1}\left(\overline{D^{20}}[8]\right) = \eta. \end{aligned} \tag{58}$$

Since the leftmost column of D^{20} is known, the attacker can calculate η, which must be equal to Cell 14 of A^{20} since it is the output of the 19th-round MixColumns operation with one active input nibble.

$$\begin{aligned} A^{20}[14] \oplus \overline{A^{20}}[14] &= S^{-1}\left(D^{20}[13]\right) \oplus S^{-1}\left(\overline{D^{20}}[13]\right) \\ &= S^{-1}\left(A^{21}[1] \oplus A^{21}[13]\right) \oplus S^{-1}\left(\overline{A^{21}}[1] \oplus \overline{A^{21}}[13]\right). \end{aligned} \tag{59}$$

It holds that $A^{21}[1] = S^{-1}\left(C^{21}[1] \oplus k^{21}[1]\right)$ and $\overline{A^{21}}[1] = S^{-1}(\overline{C^{21}}[1] \oplus k^{21}[1])$. By calculating Equations (58) and (59), the attacker can solve for $k^{21}[1] = tk_1^1[3] \oplus L^{10}(tk_2^1[3])$. One solution on average is guaranteed by Lemma 1.

9. The values $tk_1^1[i] \oplus tk_2^1[i]$, for $i = 1, 2, 3, 4$, were already determined during the calculation of the forward trail. So, using their values the attacker can determine the actual values $tk_1^1[i]$, $tk_2^1[i]$ for $i = 1, 2, 3, 4$.

10. The attacker calculates $k^{20}[2] = tk_1^1[9] \oplus L^{10}(tk_2^1[9])$ so that the active nibble in Cell 2 in A^{20} is equal to the active value η in Cells 10 and 14 since they are output of the 19th-round MixColumns operation with one active input nibble. This is done by solving

$$\eta = A^{20}[2] \oplus \overline{A^{20}}[2] = S^{-1}\left(C^{20}[2] \oplus k^{20}[2]\right) \oplus S^{-1}\left(\overline{C^{20}}[2] \oplus k^{20}[2]\right). \tag{60}$$

11. The final condition to be satisfied is that the active nibble in Cell 8 of A^{19} has to be equal to $\delta_1 \oplus L^9(\delta_2) = \gamma$.

$$\begin{aligned} \gamma &= S^{-1}\left(D^{19}[10]\right) \oplus S^{-1}\left(\overline{D^{19}}[10]\right) \\ &= S^{-1}\left(A^{20}[6] \oplus A^{20}[14]\right) \oplus S^{-1}\left(\overline{A^{20}}[6] \oplus \overline{A^{20}}[14]\right). \end{aligned} \tag{61}$$

Note that $A^{20}[6] = S^{-1}(C^{20}[6] \oplus k^{20}[6])$ and since $\overline{A^{20}}[6] = A^{20}[6]$, solving Equation (61) helps to determine $k^{20}[6] = tk_1^1[10] \oplus L^{10}(tk_2^1[10])$.

The result follows since in the Steps 1-4, a total of $2^{-28-4-4-4} = 2^{-40}$ ciphertext pairs are filtered.

Attack Procedure

When combining the findings of Lemmas 3 and 4, one can transform them to an attack procedure as illustrated in Figure 85:

1. The attacker chooses fixed differences δ_1, δ_2 satisfying $\delta_1 = L^3(\delta_2)$.
2. Furthermore, the attacker chooses the nibble values of the random base variable E^1 in all Cells except Cells 5, 7, 8, and 15.
3. For each choice of $(E^1[5], E^1[7], E^1[8], E^1[15])$ (2^{16} choices):
    - Calculate plaintext P by inverting the first round.
    - Query the 21-round encryption oracle for P, K and $\overline{P}, \overline{K}$.

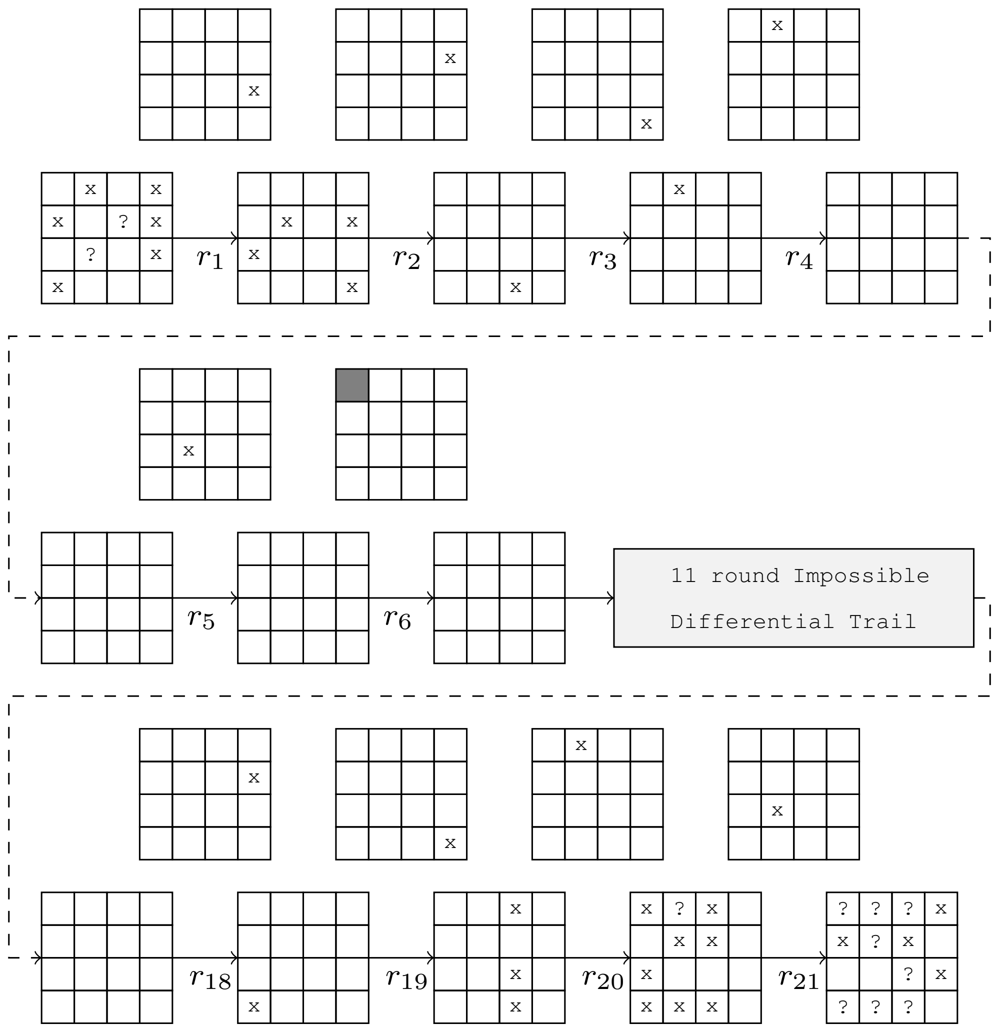


Figure 85: Related-key impossible differential attack on 21-round SKINNY 64/128 (the dark gray cell visualises the cancellation of the tweakeys).

For every choice of the base variable E^1, we have 2^{17} encryption calls. We can pair related plaintext and tweakey pairs in the following way: For every plaintext P_i, choose a plaintext P_j so that E^1 for P_i and P_j has a non-zero difference in all Cells 5, 7, 8, and 15. For every P_i, there exist $(2^4 - 1)^4 \approx 2^{15.6}$ such values of P_j, and so $2^{16+15.6} = 2^{31.6}$ pairs to work with. The attack now proceeds as follows.
For each choice of P_i, P_j ($2^{31.6}$ choices):

- Denote $P = P_i$ and $\overline{P} = P_j$.
- The attacker can choose α and proceed with the steps of the above attack with one exception: The attacker can no longer choose Δ_2 as in Step 4 of Lemma 3 since $P, \overline{P}, K, \overline{K}$ has already been chosen.

- With probability 2^{-4} (as per Lemma 2), the plaintext pair satisfies Equation 57 in Step 4 of Lemma 3 and proceeds; otherwise the attacker aborts.
- Request the ciphertexts C for (P, K) and $\overline{C}$ for $(\overline{P}, \overline{K})$.
- If $C \oplus \overline{C}$ does not pass the 2^{-36} filter (Steps 1, 2, and 3 in Lemma 4), then abort and start again.
- If they pass the filter, the attacker can guess seven tweakey cells (2^{28} guesses) and calculate 17 key/tweak cells as follows:

#	Guessed	Rnd	Calculated	Rnd
1	$tk_1^1[i] \oplus tk_2^1[i]$ for $i = 2,4,6$	1		
2	$tk_1^1[i] \oplus L(tk_2^1[i])$ for $i = 8,12,15$	2		
3	$tk_1^1[i] \oplus L^{10}(tk_2^1[i])$ for $i = 0$	21		
4			$tk_1^1[i],\ tk_2^1[i]$ for $i = 7$	3
5			$tk_1^1[i],\ tk_2^1[i]$ for $i = 1,2,3,4$	21
6			$tk_1^1[i] \oplus L^{10}(tk_2^1[i])$ for $i = 9,10$	20

The 17 tweakey nibbles used for elimination are therefore:

$$tk_1^1[i],\ tk_2^1[i] \text{ for } i = 1,2,3,4,7$$
$$tk_1^1[i] \oplus L^{10}(tk_2^1[i]) \text{ for } i = 9,10$$
$$tk_1^1[i] \oplus L^{10}(tk_2^1[i]) \text{ for } i = 0$$
$$tk_1^1[i] \oplus L(tk_2^1[i]) \text{ for } i = 8,12,15$$
$$tk_1^1[i] \oplus tk_2^1[i] \text{ for } i = 6$$

- A fraction of 2^{-4} tweakeys fulfils the condition required in Step 4 of Lemma 4.
- Therefore, the attacker has a set of $2^{28-4} = 2^{24}$ wrong key candidates.

The above procedure is repeated with 2^{55} chosen plaintexts until a single key solution remains for the 17 nibbles of the tweakey.

COMPLEXITY. For every base value of E^1, the attacker makes 2^{17} encryption calls. Out of these, the attacker has $2^{31.6}$ pairs to work with. For each pair, the attacker can then choose α in $2^4 - 1$ ways, which gives her around $2^{35.6}$ initial guesses for the forward key nibbles $k^1[1]$, $k^1[3]$, and $k^1[7]$, of which a fraction of 2^{-4} passes the filter in Equation 57. Therefore, the attacker has $2^{31.6}$ pairs to work with. In fact, for every pair (P_i, P_j) there is only one choice of α going forward on average.

$$\text{Time complexity} = \max\left\{2^{55+17} \text{ encryptions }, 2^{55-4.4+24} \text{ guesses}\right\} = 2^{55+19.6}.$$

The attacker gets $2^{55-4.4+24} = 2^{55+19.6}$ incorrect solutions for 17 nibbles. To reduce the keyspace to 1 we need:

$$2^{17\times4} \cdot \left(1 - 2^{-17\times4}\right)^{2^{55+19.6}} \approx 2^{17\times4} e^{-2^{55-48.4}} = 1.$$

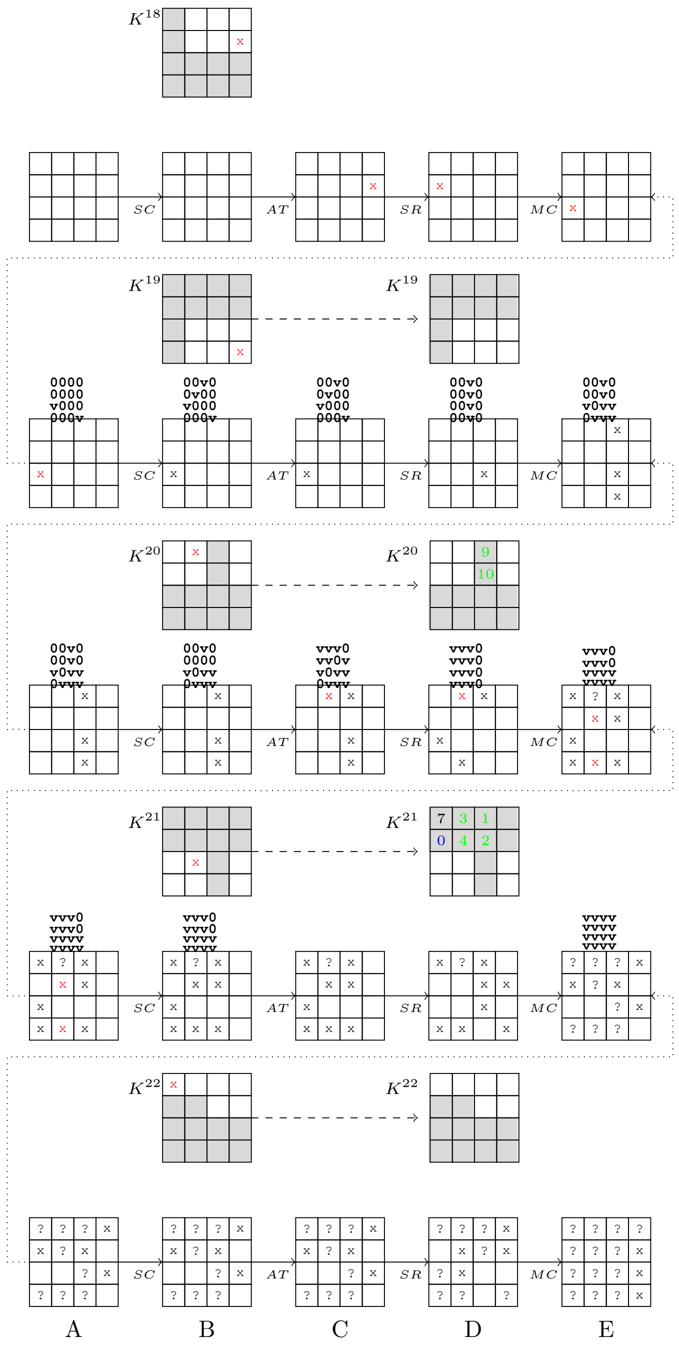


Figure 86: Trail for the five backward rounds (the values of active nibbles in red are functions of δ_1, δ_2, grey cells are the key, white cells are the tweak).

The total number of encryption calls to 21-round SKINNY-64/128 is $2^{55+17} = 2^{72}$ and the total number of guesses is $2^{74.6}$. We also need 2^{68} memory accesses, which are negligible in the total complexity. The memory complexity is upper bounded by storing one bit per key candidate which is therefore 2^{68} bits. The memory for storing the approximately $2 \cdot 2^{17}$ plaintexts and corresponding ciphertexts of a structure at each time is negligible.

6.3.3 22-Round Attack under Partially Known Tweak

The attack above can be extended to 22-round Skinny-64/128 under the assumption that 48 of the 128 bits in the tweakey are publicly known tweak (see Figure 87 for details). In particular, we assume that $tk_1^1[i], tk_2^1[i]$ for $i = 8, 11, 12, 13, 14, 15$ are reserved for the tweak. The remaining 80 bit constitute the secret key.
In this case, the attacker can add a round at the end (see Figure 86 for details). Knowing six out of eight cells in the lower half of the tweakey blocks helps in the following way. From the ciphertext (*i.e.,* E^{22}), one can revert the final round to compute E^{21} if we guess $k^{22}[4,5]$, *i.e.,* , $tk_1^1[9,10] \oplus L^{11}(tk_2^1[9,10])$. The attack is almost the same as the previous attack, except that the tweakey indices $i = 8, 11, 12, 13, 14, 15$ and their functions are known and do not need to be guessed.

1. Generate $2^{31.6}$ plaintext/ciphertext pairs from every base choice of E^1 and 2^{17} encryption calls.
2. For each choice of P_i, P_j ($2^{31.6}$ choices):
   - Denote $P = P_i$ and $\overline{P} = P_j$.
   - The attacker can choose α and calculate $k^1[1]$, $k^1[3]$, and $k^1[7]$ as per Step 3 of Lemma 3.
   - The attacker can no longer choose Δ_2 as in Step 4 of Lemma 3 since P, $\overline{P}$, K, $\overline{K}$ already have been chosen.
   - With probability 2^{-4}, the plaintext pair satisfies Equation 57 in Step 4 of Lemma 3 and proceeds; otherwise the attacker aborts.
   - The attacker does not need to guess the Round 2 tweakey nibbles since these are in the lower half of the tweakey blocks and therefore known.
   - Retrieve the ciphertext $\overline{C}$ for $(\overline{P}, \overline{K})$ and the ciphertext C for (P, K).
   - Guess $k^{22}[4,5] = tk_1^1[9,10] \oplus L^{11}(tk_2^1[9,10])$ to get E_{21}.
   - If $E_{21} \oplus \overline{E_{21}}$ does not pass the 2^{-36} filter, then abort and restart.
   - After determining $k^{20}[2] = tk_1^1[9] \oplus L^{10}(tk_2^1[9])$ and $k^{20}[6] = tk_1^1[10] \oplus L^{10}(tk_2^1[10])$ in Steps 10 and 11 of Lemma 4, the attacker can uniquely determine $tk_1^1[9,10]$ since $tk_1^1[9,10] \oplus L^{11}(tk_2^1[9,10])$ is already guessed.
   - If they pass the filter, the attacker can guess six tweakey cells (2^{24} guesses) and calculate 16 key cells as follows:

#	Guessed	Rnd	Calculated	Rnd
1	$tk_1^1[i] \oplus tk_2^1[i]$ for $i = 2,4,6$	1		
2	$tk_1^1[i] \oplus L^{10}(tk_2^1[i])$ for $i = 0$	21		
3	$tk_1^1[i] \oplus L^{11}(tk_2^1[i])$ for $i = 9,10$	22		
4			$tk_1^1[i], tk_2^1[i]$ for $i = 7$	3
5			$tk_1^1[i], tk_2^1[i]$ for $i = 1,2,3,4$	21
6			$tk_1^1[i], tk_2^1[i]$ for $i = 9,10$	20

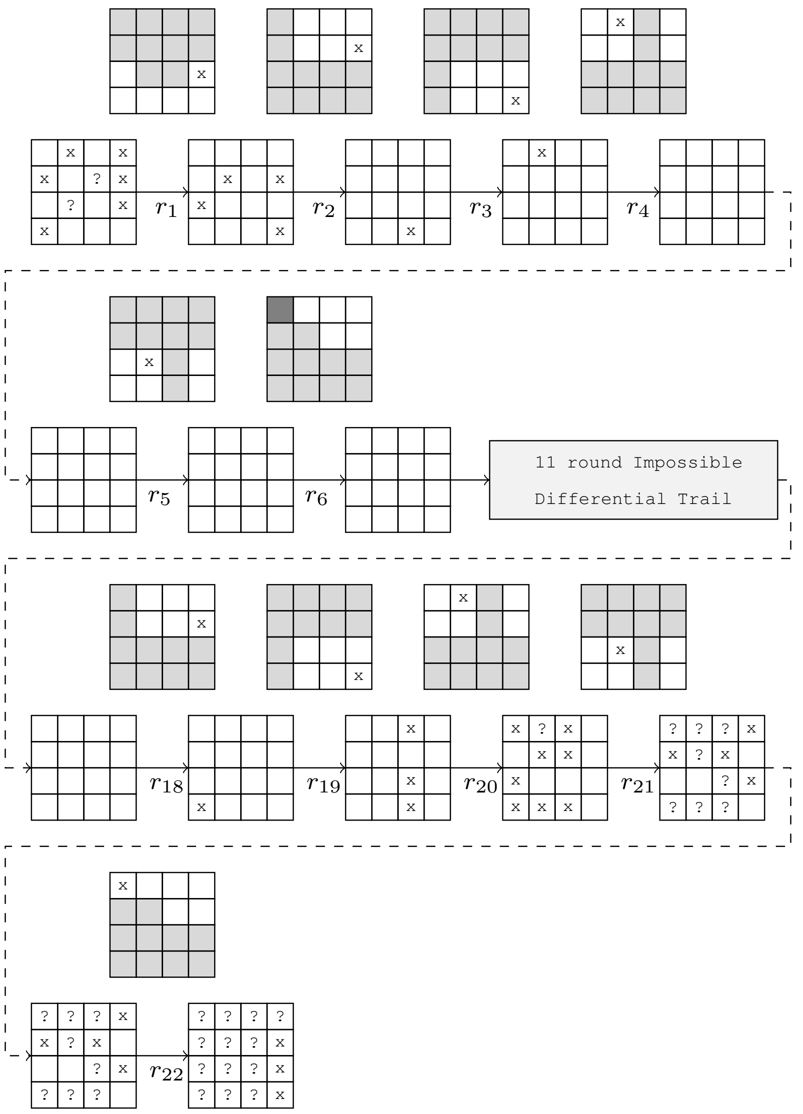


Figure 87: Related-Key Impossible Differential Attack on 22 round SKINNY 64/128 (grey cells are the key, white cells are the tweak, the dark gray cell visualises the cancellation of the tweakeys).

The 16 tweakey nibbles used for elimination are therefore:

$tk_1^1[i], tk_2^1[i]$ for $i = 1, 2, 3, 4, 7, 9, 10$.
$tk_1^1[i] \oplus L^{10}(tk_2^1[i])$ for $i = 0$.
$tk_1^1[i] \oplus tk_2^1[i]$ for $i = 6$.

- A fraction of 2^{-4} tweakeys fulfils the condition in Step 4 of Lemma 4.
- Therefore, the attacker has a set of $2^{24-4} = 2^{20}$ wrong key candidates.

The procedure above is repeated with 2^{54} chosen plaintexts until a single key solution remains for the 16 nibbles of the tweakey.

COMPLEXITY. For every base value of E^1, the attacker makes 2^{17} encryption calls. Out of these, the attacker has $2^{31.6}$ pairs to work with. For each pair, the attacker can choose then α in $2^4 - 1$ ways, which gives the attacker around $2^{35.6}$ initial guesses for the forward key nibbles $k^1[1], k^1[3], k^1[7]$, of which only a fraction of 2^{-4} passes the filter in Equation 57. Therefore, the attacker has $2^{31.6}$ pairs to work with. For every pair (P_i, P_j) there is only once choice of α going forward on average.

$$\text{Time complexity} = \max\left\{2^{54+17} \text{ encryptions}, 2^{54-4.4+20} \text{ guesses}\right\} = 2^{54+17}.$$

The attacker gets $2^{54-4.4+20} = 2^{54+15.6}$ incorrect solutions for 16 nibbles. To reduce the keyspace to 1 we need:

$$2^{16\times 4} \cdot \left(1 - 2^{-16\times 4}\right)^{2^{54+15.6}} \approx 2^{16\times 4} e^{-2^{54-48.4}} = 1.$$

The total number of encryption calls to 22-round SKINNY-64/128 is $2^{54+17} = 2^{71}$. We also need 2^{64} memory accesses, which are negligible in the total complexity. The memory complexity is upper bounded by storing one bit per key candidate which is therefore 2^{64} bits. The memory for storing the approximately $2 \cdot 2^{17}$ plaintexts and corresponding ciphertexts of a structure at each time is negligible.

6.3.4 23-Round Attack under Partially Known Tweak

We can further extend the 22 round attack to a 23 round attack by prepending one round at the beginning. In order to not disrupt the notation, we denote the additional round prepended at the beginning as the 0^{th} round. That is, the 23 rounds are labelled as Rounds 0 to 22, and the variables $A^0, B^0, \ldots$ are defined as above. The plaintext is denoted by A^0 and the ciphertext by E^{22}. Note that, from the base value of E^1, the plaintext can be calculated if we guess $k^0[9, 10]$. There are two principal differences to the 22-round attack.

1. When the attacker guesses $k^{22}[4, 5]$ which is $tk_1^1[9, 10] \oplus L^{11}(tk_2^1[9, 10])$ to invert the final round to get E_{21}, $tk_1^1[9, 10]$ and $tk_2^1[9, 10]$ is uniquely determined. This is because at the beginning of the outer loop $k^0[9, 10]$ has already been guessed by the attacker to invert the initial round.
2. As the attacker can no longer determine $k^{20}[2] = tk_1^1[9] \oplus L^{10}(tk_2^1[9])$ and $k^{20}[6] = tk_1^1[10] \oplus L^{10}(tk_2^1[10])$ using Steps 10 and 11 of Lemma 4. The probability that with the given values of $tk_1^1[9, 10]$ and $tk_2^1[9, 10]$, Equations (60) and (61) are satisfied is 2^{-8}. This decreases the probability of ciphertext filter from 2^{-36} to 2^{-44}.

For each initial guess of $k^0[9, 10]$, the guessed and calculated key bytes are:

#	Guessed	Rnd	Calculated	Rnd
1	$tk_1^1[i] \oplus tk_2^1[i]$ for $i = 2, 4, 6$	1		
2	$tk_1^1[i] \oplus L^{10}(tk_2^1[i])$ for $i = 0$	21		
3	$tk_1^1[i] \oplus L^{11}(tk_2^1[i])$ for $i = 9, 10$	22		
4			$tk_1^1[i], tk_2^1[i]$ for $i = 7$	3
5			$tk_1^1[i], tk_2^1[i]$ for $i = 1, 2, 3, 4$	21

The 14 tweakey nibbles used for elimination are therefore:

$tk_1^1[i], tk_2^1[i]$ for $i = 1, 2, 3, 4, 7$.
$tk_1^1[i] \oplus L^{10}(tk_2^1[i])$ for $i = 0$.
$tk_1^1[i] \oplus tk_2^1[i]$ for $i = 6$.
$tk_1^1[i] \oplus L^{11}(tk_2^1[i])$ for $i = 9, 10$

As before, a fraction of 2^{-4} tweakeys fulfils the condition in Step 4 of Lemma 4. Therefore, the attacker has a set of $2^{24-4} = 2^{20}$ wrong key candidates.

COMPLEXITY. For each iteration of the outer loop, the complexity is calculated as follows: For every base value of E^1, the attacker makes 2^{17} encryption calls. Out of those, the attacker has $2^{31.6}$ pairs to work with. For each pair, the attacker can choose then α in $2^4 - 1$ ways, which gives the attacker around $2^{35.6}$ initial guesses for the forward key nibbles $k^1[1]$, $k^1[3]$, $k^1[7]$, of which only a fraction of 2^{-4} passes the filter in Equation 57. For every pair (P_i, P_j) there is only one choice of α going forward on average.

$$\text{Time complexity} = \max\left\{2^{54+17} \text{ encryptions}, 2^{54+31.6-44+20} \text{ guesses}\right\} = 2^{54+17}.$$

The attacker gets $2^{54+31.6-44+20} = 2^{54+7.6}$ incorrect solutions for 14 nibbles. To reduce the keyspace to 1 we need:

$$2^{14\times4} \cdot \left(1 - 2^{-14\times4}\right)^{2^{54+7.6}} \approx 2^{14\times4} e^{-2^{54-48.4}} = 1.$$

The total number of encryption calls to 22-round SKINNY-64/128 can then be estimated with $2^{54+17} = 2^{71}$. Multiplying this by 2^8 for the outer loop gives a total complexity of $2^{71+8} = 2^{79}$ which is just short of exhaustive search for the 80-bit key. We also need $2^{54+8} = 2^{62}$ memory accesses, which are negligible in the total complexity. The memory complexity is upper bounded by storing one bit per key candidate which is therefore 2^{64} bits. The memory for storing the approximately $2 \cdot 2^{17}$ plaintexts and ciphertexts of a structure is negligible.

6.4 RELATED-KEY INTEGRAL ATTACKS

Integral attacks were first introduced by Daemen *et al.* [118] as a dedicated attack against the block cipher SQUARE and later extended to integral attacks [205]. These attacks have been shown extremely powerful against AES-like ciphers [139, 155, 321]. Integral attacks prepare a set of chosen plaintexts so that particular cells of the state are held constant, while other cells vary trough all possible values. Then an attacker considers some properties of the set, when propagating the set through several rounds of the cipher (For further details on integral attacks see Chapter 2.4.6).
The common properties used in standard integral attacks are:

- **Active (A):** The value of the cell takes all possible values in the set.

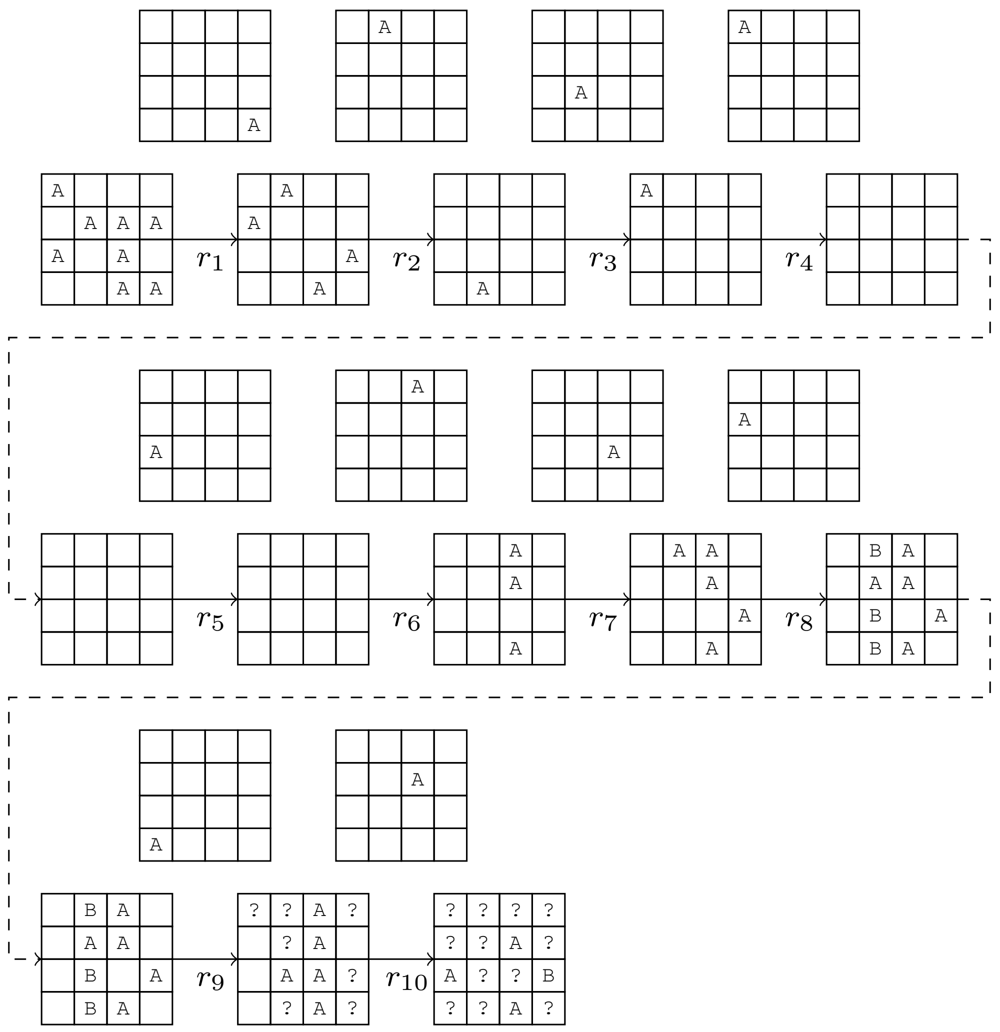


Figure 88: Related-tweakey integral distinguisher over 10 rounds of Skinny-64/128. Empty cells are constant (*i.e.,* C) values.

- **Constant (C):** The value of the cell is fixed to a constant value.
- **Unknown (?):** The value of the cell is unknown.
- **Balanced (B):** The XOR-sum of all values in the cell is zero.

After propagating the set through several rounds, an attacker then guesses a key-value and checks if the *balanced* property holds for a particular cell in the state.

6.4.1 Related-Tweakey Integral Distinguisher

The designers of Skinny [50] already published integral attacks, alongside with the specification of Skinny, as part of their initial security analysis. However, in their attacks they do not insert *active* cells in the tweakey, but rather just add *active* cells in

the state. One would assume that adding *active* cells in the tweakey would improve the distinguishers, as one can simply cancel the *active* cells from the state and the tweakey (if they consist of the exact same values), as shown in [139].
We constructed integral distinguishers, considering active cells in TK1, TK2 and TK3, which would represent Skinny-n/n, Skinny-n/2n and Skinny-n/3n. Our distinguisher is experimentally obtained by choosing one *active* cell in the state and one in each tweakey word (TK1, TK2 and TK3). We then used the trick by Dobraunig *et al.* [139] and cancel the *active* cells to obtain a constant cell, getting one round for free. The next addition of the tweakey and the state, will then introduce the *active* cell back into the state. By also considering backwards propagation of the *active* cells, we turn the square attack into an integral attack. Figure 88 illustrates the best integral distinguisher we found, by considering *active* cells in the 3rd round of the state, and in just one tweakey word (*i.e.,* TK1). We can then propagate the *active* cell backwards by three rounds, and forwards by seven rounds, reaching a 10-round distinguisher.
In the initial cryptanalysis, published alongside with the specification of Skinny [50], the designers showed integral distinguishers for up to 10-rounds. Compared to the related-tweakey integral distinguishers, it seems that one cannot reach more rounds by adding *active* cells in the tweakey to improve integral attacks on Skinny. Even though our distinguishers do not improve the attacks, they still give further insights in the related-tweakey security of Skinny.

6.4.2 14-Round Integral Attacks on Skinny-64/128 [50]

The designers of Skinny [50] further applied a key-recovery attack by appending 4-rounds after the 10-round integral distinguisher. The attack requires $2^{12c} = 2^{48}$ data and $2^{8c} = 2^{32}$ memory and the time complexity is $\max\{2^{5+12c}, 2^{5+11c}, 2^{5+8c}, 2^{10c}\}$ and results in 2^{53} computations. Due to the just few number of rounds that can be attacked with integral attacks, we did not further consider integral attacks on Skinny.

6.5 CONCLUSION

In this chapter, we analyse the tweakable block cipher Skinny-64/128 against impossible differential and integral attacks in the related-tweakey model. Skinny is an highly interesting target, as it achieves the hardware performance of the NSA cipher Simon and still offers strong security bounds against differential/linear attacks. We outline related-key impossible-differential attacks against 21-round Skinny-64/128 as well as attacks on 22 and 23 rounds under the assumption of having 48-bit of the 128-bit tweakey as public tweak. Our attacks are based on an 11-round impossible differential trail, to which we prepend six and append five rounds before and after the trail, respectively, to obtain an attack on 22 rounds. Finally, we can prepend a 23rd round under similar assumptions.
Furthermore, we study related-tweakey integral attacks on Skinny by considering active nibbles not only in the state, but also in the tweakey. Yet, it turns out that one

cannot improve the distinguishers, but we can give further insights in the security of Skinny in the related-tweakey model. However, we further study integral attacks on Skinny, where we construct the distinguishers from zero-correlation linear distinguisher as shown in Chapter 5.

7 DIFFERENTIAL CRYPTANALYSIS OF ROUND-REDUCED SPARX-64/128

CONTENTS

EXECUTIVE SUMMARY. In this chapter, we analyse the lightweight block cipher Sparx presented at ASIACRYPT'16. We present truncated-differential attacks and rectangle attacks on a round-reduced version of Sparx based on [26].
Sparx is a family of ARX-based block ciphers designed according to the *long-trail strategy* (LTS) that was introduced together with Sparx by Dinu *et al.* at ASIACRYPT 2016. Similar to the wide-trail strategy, the LTS allows provable upper bounds on the length of differential trails and linear paths. Thus, the cipher is a highly interesting target for third-party cryptanalysis. However, the only third-party cryptanalysis on Sparx-64/128 to date was given by Abdelkhalek *et al.* at AFRICACRYPT'17 who proposed impossible-differential attacks on 15 and 16 (out of 24) rounds.
We present chosen-ciphertext differential attacks on 16 rounds of Sparx-64/128. First, we show a truncated-differential analysis that requires 2^{32} chosen ciphertexts and approximately 2^{93} encryptions. Second, we illustrate the effectiveness of boomerangs on Sparx by a rectangle attack that requires approximately $2^{59.6}$ chosen ciphertexts and about $2^{122.2}$ encryption equivalents.

DECLARATION OF AUTHORSHIP. The work described in this chapter is based on the paper [26]: *Differential Cryptanalysis of Round-Reduced* Sparx-64/128 and was presented at *The* 16th *International Conference on Applied Cryptography and Network Security (ACNS'*18*)* in Leuven, Belgium. The paper is joint work with Eik List. All authors contributed equally to the results of the paper. The contributions of the author are the following:

- Searching for differential trails to construct truncated-differential, rectangle and yoyo distinguishers.
- Experimental verification of the truncated-differential distinguisher.
- Improving the key-recovery attacks to reach more rounds and to reduce the complexities.

In the full version of [27], we further present a yoyo attack, which is the contribution of Eik List.

7.1 INTRODUCTION

ARX CIPHERS. The design and cryptanalysis of block ciphers is a heuristic competition between designers and analysts. With the introduction of the wide-trail design strategy in Rijndael, designers could finally provide provable bounds for the expected probabilities and therefore for the maximal length of differential trails and linear trails of block ciphers. Rijndael followed the substitution-permutation network design approach, compared to some other ARX-ciphers (*i.e.*, IDEA [214] and RC5 [285]) that use only the omnipresent modular addition, XOR, rotation, and shift operations that nearly every processor supports out-of-the-box. As a consequence, SPNs demand an expertised tailoring of their implementations to the operating platform to be comparably efficient as bit-based designs.
However, in resource-constrained environments, the most efficient software implementations are still ciphers that employ only logical operations and/or addition, *e.g.*, ciphers based on modular additions, rotations, and XOR (ARX). Hence, until recently, there has been a gap between the provable bounds of wide-trail designs, and the efficiency of ARX-based constructions.

SPARX. At ASIACRYPT'16, Dinu *et al.* introduced SPARX [135], the first ARX-based family of block ciphers that provides provable bounds on the maximal length of differential trails and linear trails. Alongside SPARX, the authors developed the *long-trail* design strategy, a general approach for ARX-based symmetric-key primitives to obtain provable bounds. Both the long-trail strategy in general, and SPARX in particular, are interesting targets of cryptanalysis as they try to bridge the gap between efficiency and providing security bounds. The question arises if SPARX is also secure against (truncated) differential and boomerang attacks that can exploit clustering effects of many differential trails.

RELATED WORK. In the specification of SPARX, the designers reported on their results of a first automated analysis that no differential trail with probability higher than 2^{-n} nor any linear trail with bias higher than $2^{-n/2}$ exists over five or more steps. Moreover, they described integral attacks on up to five out of eight steps of SPARX-64/128, and six out of ten steps of SPARX-128/*. However, those initial attacks are naturally limited due to time constraints when designing a new ci-

Table 28: Previous and proposed attacks on SPARX-64/128. KP/CP/CC = known plaintext/-chosen plaintext/chosen ciphertext. ID = Impossible differential, TD = Truncated differentials, ZC = Zero-Correlation, MZC = Multidimensional Zero-Correlation.

Instance	Rounds	Attack type	Time	Data	Memory	Ref.
SPARX-64/128	15/24	Integral	$2^{101.0}$	$2^{37.0}$ CP	$2^{64.0}$	[135]
SPARX-64/128	15/24	ID	$2^{94.1}$	$2^{51.0}$ CP	$2^{43.5}$	[2]
SPARX-64/128	16/24	ID	$2^{94.0}$	$2^{61.5}$ KP	$2^{61.5}$	[2]
SPARX-64/128	16/24	TD	$2^{92.0}$	$2^{32.0}$ CC	$2^{61.0}$	Sect. 7.4
SPARX-64/128	16/24	Rectangle	$2^{122.2}$	$2^{59.6}$ CC	$2^{61.6}$	Sect. 7.5
SPARX-64/128	16/24	Yoyo	$2^{126.0}$	$2^{64.0}$†CP	$2^{64.0}$	[27]
SPARX-128/128	22/32	Integral	2^{105}	2^{102} CP	2^{72}	[135]
SPARX-128/128	26/32	MZC	$2^{117.25}$	$2^{116.2}$ KP	-	[329]
SPARX-128/256	24/40	Integral	2^{233}	2^{104} CP	2^{202}	[135]
SPARX-128/256	29/40	ZC	$2^{227.2}$	2^{128}‡KP	-	[329]

pher, and therefore demand a deeper analysis by the cryptographic community. At AFRICACRYPT'17, Abdelkhalek *et al.* [2] proposed 12- and 13-round impossible-differential distinguishers on SPARX-64/128, using the four-step distinguisher for balanced Type-1 Feistel networks. They extended their attacks by three rounds, where they exploited dependencies between the key words from the key-schedule. At SAC'17, Tolba *et al.* proposed multi-dimensional zero-correlation linear attacks on up to 26 rounds of SPARX-128/128, and on up to 29 rounds of SPARX-128/256 [329].

7.2 DESCRIPTION OF THE SPARX FAMILY OF CIPHERS

The SPARX-n/k family comprises three versions, SPARX-64/128, SPARX-128/128, and SPARX-128/256, where n indicates the block size, and k the key length k. The cipher is based on a Feistel network with two state words for SPARX-64 and four state words for SPARX-128, consisting of n_s Feistel steps. Each step consists of r_a rounds of an ARX-based round function; plaintexts and ciphertexts consist of $w = n/32$ words $X^0, \ldots, X^{w-1}$ of 32 bits each; the key is divided into 32-bit words $(\kappa^0, \ldots, \kappa^{v-1})$. The values for the individual versions of SPARX are summarised in Table 29, the components of the cipher are also depicted in Figure 89.

SPARX-64/128. The structure of SPARX-64 is reminiscent of a Feistel network of eight steps. Each step consists of $r_a = 3$ rounds of the ARX-box $\mathcal{A}$, (*i.e.,* three rounds of SPECKEY) on each branch. The Feistel function $\mathcal{L}$ is a linear involutive

† The 16-round yoyo attack on SPARX-64/128 in the full version of [27] required the full codebook.
‡ The 29-round zero-correlation attack on SPARX-128/256 of [329] requires the full codebook.

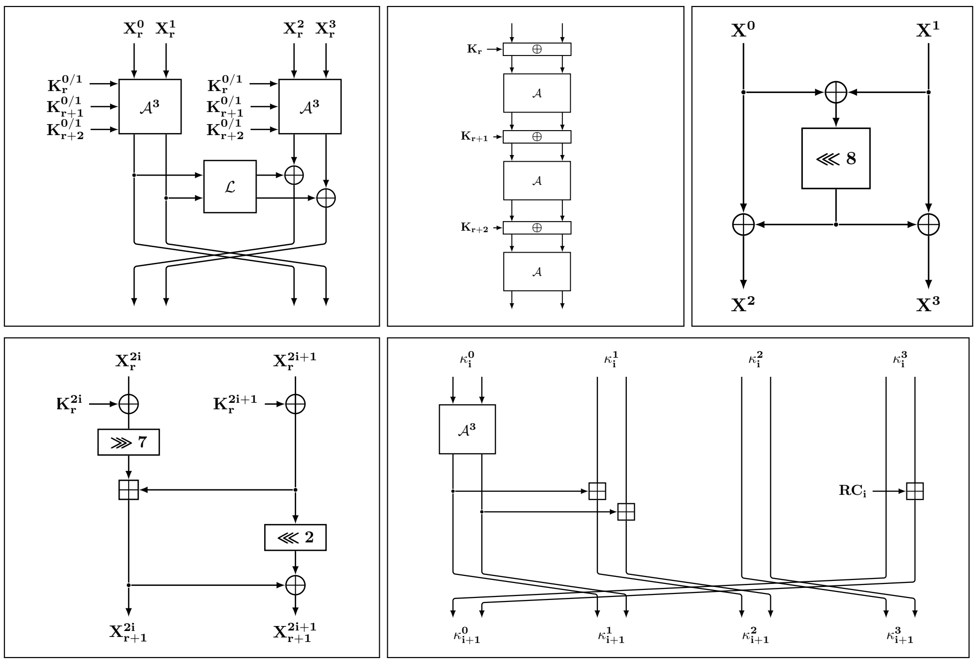

X_r^0 X_r^1 X_r^2 X_r^3
$K_r^{0/1}$ $K_{r+1}^{0/1}$ $K_{r+2}^{0/1}$
$\mathcal{A}^3$
$\mathcal{L}$
K_r K_{r+1} K_{r+2}
$\mathcal{A}$
X^0 X^1
$\lll 8$
X^2 X^3
X_r^{2i} X_r^{2i+1}
K_r^{2i} K_r^{2i+1}
$\ggg 7$
$\lll 2$
X_{r+1}^{2i} X_{r+1}^{2i+1}
κ_i^0 κ_i^1 κ_i^2 κ_i^3
RC_i
κ_{i+1}^0 κ_{i+1}^1 κ_{i+1}^2 κ_{i+1}^3

Figure 89: High-level view of SPARX-64. **Top left:** The step function. **Top center:** The $\mathcal{A}^3$ layer in SPARX-64. **Top right:** The linear layer $\mathcal{L}$. **Bottom left:** The $\mathcal{A}$ function SPECKEY. **Bottom right:** One iteration of the key schedule of SPARX-64/128.

permutation $\mathcal{L} : \mathbb{F}_2^{32} \to \mathbb{F}_2^{32}$ inspired by [190]. Given the left 32-bit state word $x \| y$, the input is split into 16-bit parts x, y, and is mapped to

$$\mathcal{L}(x\|y) \overset{\text{def}}{=} (x \oplus ((x \oplus y) \lll 8)) \| (y \oplus ((x \oplus y) \lll 8)).$$

We denote the 64-bit state after Round r interchangeably as $(L_r, R_r) = (X_r^0 \parallel X_r^1, X_r^2 \parallel X_r^3) = (X_r^L \parallel Y_r^L, X_r^R \parallel Y_r^R)$, and the round key used in Round r interchangeably as $(K_r^L, K_r^R) = (K_r^0 \| K_r^1, K_r^2 \| K_r^3)$.

THE KEY SCHEDULE OF SPARX-64. The 128-bit secret key of SPARX-64/128 is divided into four initial 32-bit words $(\kappa_0^0, \kappa_0^1, \kappa_0^2, \kappa_0^3)$. In each step, the key schedule transforms the leftmost 32-bit word κ_s^0 in one iteration of the ARX-box $\mathcal{A}$, adds the output to the right neighbouring word κ_s^1, adds a round constant RC_i to the rightmost 16-bit half of κ_{2s}^3 to prevent slide attacks, and finally rotates the four words by one position to the right. The $r_a = 3$ leftmost words κ_{2s}^0, κ_{2s}^1, κ_{2s}^2 are used as round keys for the first, second, and third round of the left branch of Step $s+1$; the $r_a = 3$ left-most words κ_{2s+1}^0, κ_{2s+1}^1, κ_{2s+1}^2 are used for the first, second, and third round of the right branch of Step $s+1$. For example, $(\kappa_0^0, \kappa_0^1, \kappa_0^2)$ are used as round keys for the left branch in the first step, and $(\kappa_1^0, \kappa_1^1, \kappa_1^2)$ are used as round keys for the right branch in the first step.

Table 29: Parameters of the individual versions of Sparx.

Cipher	#State-words w	#Key-words v	#Rounds/step r_a	#Steps n_s
Sparx-64/128	2	2	3	8
Sparx-128/128	4	4	4	8
Sparx-128/256	4	8	4	10

7.2.1 Properties and Observations

PROPERTIES OF THE KEY SCHEDULE. Abdelkhalek *et al.* [2] observed that one can obtain the rounds keys for 2.5 consecutive rounds by guessing only 64-bit of key material. More precisely, one obtains the round keys for Round $3r+1$ and the round key for the right 32-bit branch in Round $3r+2$ by guessing the 64 bit of the key material of Round $3r$:

Property 2. Given κ^2_{s+1} and κ^3_{s+1}, one can directly derive the key words $\kappa^2_s = \kappa^3_{s+1}$, $\kappa^0_{s+2} = \kappa^3_{s+1}$, $\kappa^1_{2s+3} = \mathcal{A}(\kappa^0_{s+2})$, and $\kappa^0_{s+3} = \kappa^2_{s+1}$.

PROPERTIES OF THE LINEAR LAYER. We can observe the following for the linear layer $\mathcal{L}$ of Sparx.

Property 3. Given two distinct inputs $(x \| y), (x' \| y') \in \mathbb{F}_2^m$ and define their difference $\Delta = (\Delta x \| \Delta y) = (x \oplus x') \| (y \oplus y')$. If $\Delta x = \Delta y = \Delta$, *i.e.*, $(x \oplus x') = (y \oplus y') = \Delta$, then it holds that

$$\mathcal{L}(x \| y) \oplus \mathcal{L}(x' \| y') = (\Delta \| \Delta).$$

Property 3 and the Feistel structure of Sparx imply the corollary below.

Corollary 1. If $\Delta L_i = \Delta R_i = (\Delta \| \Delta)$ after $\mathcal{A}^{r_a}$, it follows that $\Delta L_{i+1} = 0^m$ since $\Delta L_{i+1} = \Delta R_i \oplus \mathcal{L}(\Delta L_i) = (\Delta \| \Delta) \oplus (\Delta \| \Delta) = 0^m$.

PROPERTIES OF THE ARX-BOXES. Given a difference in a certain round, we can formulate the following.

Property 4. Given a 32-bit difference $\Delta_i = (\Delta x_i, \Delta y_i)$, we can directly derive $\Delta y_{i-1} \leftarrow (\Delta x_i \oplus \Delta y_i) \ggg 2$ with probability one.

Furthermore, Leurent [224] first showed Property 5. It reduces the effort of studying all combinations of pairs to that of comparing their outputs from F. One can further reduce the rank of F to $n-d$ so that outputs of F collide if and only if their inputs have one of 2^d differences.

Property 5. *Assume,* $\Delta \in \mathbb{F}_2^n$ *is a fixed difference, and* $x^0, \ldots, x^m \in \mathbb{F}_2^n$ *represent* m *values for which the goal is to find pairs* (x^i, x^j) *that result in* $x^i \oplus x^j = \Delta$. *Then, one can define a linear function* $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ *with rank* $n-1$, *s. t.* $F(\Delta) = 0^n$; *thus, all pairs* (x^i, x^j) *with* $x^i \oplus x^j = \Delta$ *will collide in* $F(x^i) = F(x^j)$.

Table 30: An optimal six-round differential trail.

Rd.	ΔL_i	ΔR_i	h_w	
0	00000000	02110a04	–	–
1	00000000	28000010	0	4
2	00000000	00400000	0	2
3	00000000	80008000	0	0
{L}	80008000	00000000	0	0
4	81008102	00000000	1	0
5	8000840a	00000000	2	0
6	850a9520	00000000	4	0
{L}	af1abf30	850a9520	0	0

Table 31: Optimal differentials for up to ten rounds of Sparx-64; t is the run time of each search.

#Rds.	Δ_{in}	Δ_{out}	h_w	t
1	(00408000, 00000000)	(00000002, 00000000)	0.00	0.02s
2	(00102000, 00000000)	(80008002, 00000000)	1.00	0.10s
3	(28000010, 00000000)	(83008302, 81008102)	3.00	0.46s
4	(00000000, 28000010)	(8000840a, 00000000)	4.99	2.40s
5	(00000000, 02110a04)	(8000840a, 00000000)	8.99	25.07s
6	(00000000, 02110a04)	(af1abf30, 850a9520)	12.99	0.06h
7	(00000000, 14881008)	(82048e0e, 8000840a)	23.95	47.80h
8	(00000000, 540a0120)	(8000840a, 8000840a)	28.53	15.20d
9	(28000010, 28000010)	(d2609263, d1209123)	32.87	22.30d
10	(28000010, 28000010)	(80818283, 80008002)	38.12	32.50d

7.3 DIFFERENTIAL TRAILS AND BOOMERANG DISTINGUISHERS

We employed a two-step approach; First, we search for optimal differential trails for up to ten rounds of Sparx-64. Those form the base of the wrapping rounds before and after the boomerang switches. Thereupon, we consider three most promising types of boomerangs over five steps.

7.3.1 Searching Optimal Differential Trails

We implemented variants of Sparx in CryptoSMT [313], an open-source tool based on the SAT/SMT solvers CryptoMiniSat [236] and STP [336] to search for optimal differential trails[§]. In this case, the problem to find optimal differential trails is modelled as a Boolean satisfiability problem, and can then be solved by a SAT solver. As the differential model of a cipher can be rather complex, we model the problem as a more general SMT (Satisfiability Modular Theories) problem. The difference to SAT problems is that SMT problems can express richer languages where *e.g.,* sets of variables can be expressed as predicates or the problem can be modelled on word level. We describe the differential behaviour of Sparx using the CVC language. This allows us to define specific constraints that can be used to limit the search space for the SAT solver. The solver then tries to find all possible valid differential trails for the given parameters with decreasing probability.
Table 30 shows an optimal six-round differential trail. Note that h_w denotes $h_w = -\log_2(p)$, for the differential probability p through a round. One can observe that optimal differential trails for Sparx-64 possess an hourglass structure, *i.e.,* the number of active bits is minimal in the middle and increases outwards. This is typical for ARX designs, a single difference is quickly expanded by the avalanche effect, when

[§] The differential models for Sparx are available at: `https://github.com/TheBananaMan/sparx-differential-attacks`

Table 32: **Top** (left to right): best trails found for our differentials of Type 1a, Type 1b, Type 1c, and Type 1d. **Middle**: best trails found for our differentials of Type 2a, Type 2b, Type 2d, and Type 2e. **Bottom**: Type 2c, Type 3a, Type 3b, and Type 3c. Σ denotes the sum of h_w over all rounds.

Rd.	ΔL_i	ΔR_i	h_w	Rd.	ΔL_i	ΔR_i	h_w	Rd.	ΔL_i	ΔR_i	h_w	Rd.	ΔL_i	ΔR_i	h_w
0	00000000	28000010	– –	0	28000010	28000010	– –	0	40404000	00400000	– –	0	80008000	80008000	– –
1	00000000	00400000	0 2	1	00400000	00400000	2 2	1	40804081	80008000	2 0	1	81008102	81008102	1 1
2	00000000	80008000	0 0	2	80008000	80008000	0 0	2	40004205	81008102	3 1	2	8004840e	8004840e	3 3
3	00000000	81008102	0 1	3	81008102	83008302	1 2	3	42854a90	8000840a	5 2	3	bd1aad20	870a9730	7 8
{L}	81008102	00000000	0 0	{L}	00000000	81008102	0 0	{L}	d78ddb92	42854a90	0 0	{L}	00000000	bd1aad20	0 0
Σ			3	Σ			7	Σ			13	Σ			23

Rd.	ΔL_i	ΔR_i	h_w	Rd.	ΔL_i	ΔR_i	h_w	Rd.	ΔL_i	ΔR_i	h_w	Rd.	ΔL_i	ΔR_i	h_w
0	02110a04	02110a04	– –	0	02110a04	00000000	– –								
1	28000010	28000010	4 4	1	28000010	00000000	4 0	0	28000010		–	0	28000010		–
2	00400000	00400000	2 2	2	00400000	00000000	2 0	1	00400000	00400000	2 –	1	00400000	00000000	2 –
3	80008000	80008000	0 0	3	80008000	80008000	0 0	2	80008000	80008000	0 0	2	80008000	80008000	0 0
{L}	00000000	80008000	0 0	{L}	00000000	80008000	0 0	{L}	00000000	80008000	0 0	{L}	00000000	80008000	0 0
4	00000000	81008102	0 1	3	81008102	81008102	1 1	3	00000000	81008102	0 1	3	81008102	81008102	1 1
5	00000000	8000840a	0 2	4	8000840a	8000840a	2 2	4	00000000	8000840a	0 2	4	8000840a	8000840a	2 2
6	00000000	850a9520	0 4	5	850a9520	2a102a10	4 4	5	00000000	850a9520	0 4	5	850a9520	850a9520	4 4
{L}	850a9520	00000000	0 0	{L}	2a102a10	850a9520	0 0	{L}	850a9520	00000000	0 0	{L}	2a102a10	850a9520	0 0
Σ			19	Σ			20	Σ			9	Σ			16

Rd.	ΔL_i	ΔR_i	h_w	Rd.	ΔL_i	ΔR_i	h_w	Rd.	ΔL_i	ΔR_i	h_w	Rd.	ΔL_i	ΔR_i	h_w
0	00000000	02110a04	– –	0	28000010	28000010	– –	0	00000000	00508402	– –				
1	00000000	28000010	0 4	1	00400000	00400000	2 2	1	00000000	24023408	0 4	0	00000000		–
2	00000000	00400000	0 2	2	80008000	80008000	0 0	2	00000000	50c080e0	0 7	1	00000000	0a204205	0 –
3	00000000	80008000	0 0	3	83008302	81008102	2 1	3	00000000	01810203	0 5	2	00000000	02110a04	0 5
{L}	80008000	00000000	0 0	{L}	00000000	83008302	0 0	{L}	01810203	00000000	0 0	{L}	02110a04	00000000	0 0
4	81008102	00000000	1 0	4	00000000	80088c02	0 5	4	000c0800	00000000	5 0	3	28000010	00000000	4 0
5	8000840a	00000000	2 0	5	00000000	8502b508	0 5	5	20000000	00000000	3 0	4	00400000	00000000	2 0
6	850a9520	00000000	4 0	6	00000000	d0020420	0 7	6	00400040	00000000	1 0	5	80008000	80008000	0 0
{L}	af1abf30	850a9520	0 0	{L}	d0020420	00000000	0 0	{L}	00400040	00400040	0 0	{L}	00000000	80008000	0 0
				7	00801000	00000000	4 0	7	80408140	80408140	2 2	6	81008102	81008102	1 1
				8	10015001	00000000	2 0	8	00400542	00400542	3 3	7	8000840a	8000840a	2 2
				9	52211224	00000000	5 0	9	8542904a	8542904a	4 4	8	850a9520	850a9520	4 4
				{L}	57611764	52211224	0 0	{L}	08150815	8542904a	0 0	{L}	2a102a10	850a9520	0 0
Σ			13	Σ			35	Σ			37	Σ			25

propagating outwards in both directions. ARX designs normally have short differential trails with a high probability, as after a few rounds the differences cannot be controlled and the probability decreases quickly.
Moreover, using the probability of the best trail is often assumed to be an adequate approximation of the probability of the best differential. However, this approximation is not always sufficiently accurate for lightweight ciphers (Note, this has been studied in Chapter 3, for many different lightweight block cipher design strategies). Therefore, we tried to evaluate the probability of differentials where feasible. For the best differentials for Sparx-64, we provide an overview in Table 31.

TYPES OF DIFFERENTIAL TRAILS. After searching differential trails incrementally for a given interval of rounds, we searched for optimal trails among the following types:

- **Type 1a.** Arbitrary single-step trails.

- **Type 1b.** Single-step trails with two active branches that have a single active branch after the step.
- **Type 1c.** Single-step trails with two active branches that have a single active branch before the step.
- **Type 1d.** Single-step trails with two active branches that have a single active branch before and afterwards.

The first category consists of single-step trails. The best trail for single-steps is a Type 1a trail with an all zero left branch. Type 1d is especially interesting for the truncated differential attack in Section 7.4.

- **Type 2a.** Two-step top trails which collide after the XOR in the right branch after the first step.
- **Type 2b.** Two-step bottom trails with only the left branch active at the first step.
- **Type 2c.** Two-step trails where only the left branch is active in the first, and therefore only the right branch is active in the second step.
- **Type 2d.** 4.5-round versions of Type 2a, but only two rounds before the collision for the left and one round before for the right branch.
- **Type 2e.** For the single-sided-top type, we further investigated the versions of Type 2a where the first step covers only one round.

The second category consists of two-step trails, which are also used in the boomerang and rectangle distinguishers in Section 7.5. We use the two-step trails of Type 2c for the top trail and Type 2b for the bottom trail of the rectangle distinguisher.
We further considered three-step trails for boomerang and rectangle attacks in our third category:

- **Type 3a.** Three-step trails where both branches are active in the first step, and only one branch is active in the subsequent steps, as is used in both top and bottom trail of the single-sided bottom type of boomerang.
- **Type 3b.** Three-step trails where the first two steps are of Type 3a, and both branches are active in the third step.
- **Type 3c.** 7.5-round versions of Type 3b, where only one round is considered for the first step.

The third category consists of three-step trails, which are also used in the boomerang and rectangle distinguishers in Section 7.5. The results for the best trails are summarised in Table 32.

7.3.2 Boomerang/Rectangle Distinguisher for Sparx-64/128

From the combination of the best identified trails, we continued to form boomerang and rectangle distinguishers. We considered three types of distinguishers over five steps.

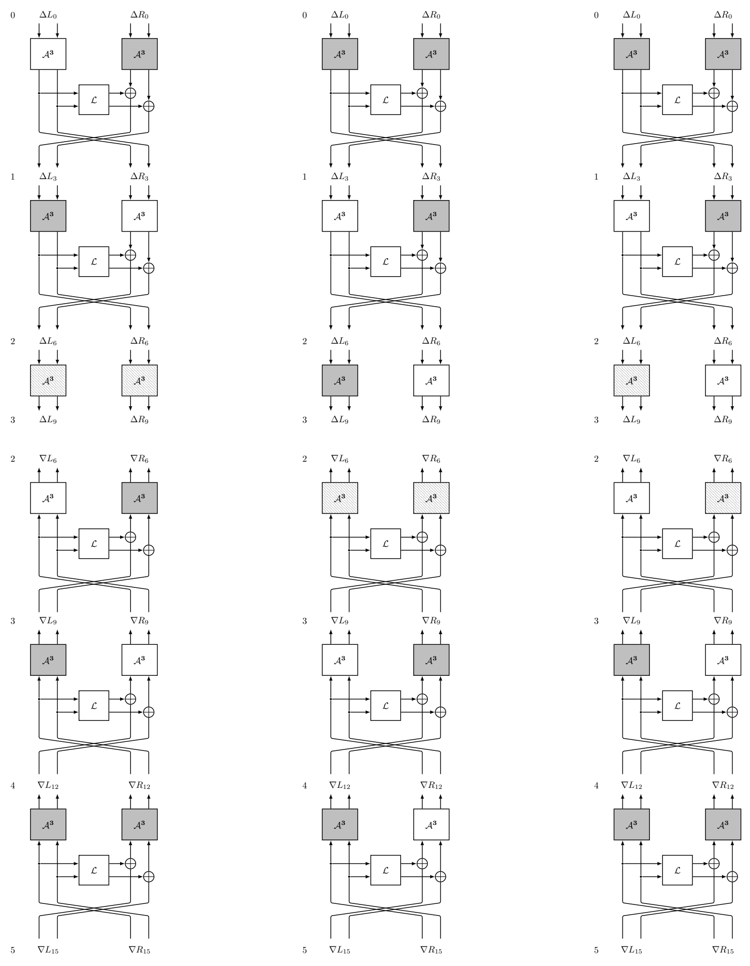


Figure 90: Types of five-step boomerangs. White $\mathcal{A}^3$ boxes are inactive (zero difference); gray $\mathcal{A}$-boxes are active (non-zero difference). Hatched boxes indicate active branches that do not have to be taken into account at the switch.

- **Free middle.** This type exploits that one can obtain the middle step for free if one chooses the top and bottom trails such that one of them possesses a zero difference in the left branch, and the other one has a zero difference in the right branch, which is a direct application of the Ladder switch. One can obtain a five-step boomerang in this way, but one will have active differences in both branches in the first and in the fifth step of the wrapping rounds.
- **Single-sided bottom.** This type has both branches active at the start of the top trail, but only one active branch at the end of the bottom trail.
- **Single-sided top.** This type has both branches active at the end of the bottom trail, but only one active branch at the beginning of the top trail.

All different types of boomerang/rectangle distinguishers are visualised in Figure 90. When studying the distinguisher, one can observe that the free-middle boomerang type allows for higher probabilities. Table 33 summarises the best boomerang distinguisher that consist of single trails that we discovered for one up to five steps. Through a single step, there exist various boomerangs with probability one:

$$\Pr\left[(\Delta L_0, \Delta R_0) \xrightarrow{1 \text{ step}} (\Delta L_3, \Delta R_3)\right] = 1,$$

for all trails with $\Delta L_0 = 0$ and $\Delta L_3 = \{L\}(\Delta R_3)$; alternatively, it also holds for all trails with $\Delta R_0 = \Delta R_3 = 0$.
Over two steps, there exist two-step boomerangs with

$$\Pr\left[(\Delta L_0, \Delta R_0) \xrightarrow{2 \text{ steps}} (\Delta L_6, \Delta R_6)\right] \geqslant 2^{-6},$$

namely for trails of the form

- $\Delta L_0 = 0$ and $\Delta R_0 \in \{\texttt{28000010}, \texttt{00400000}\}$ and $\Delta L_6 = \{L\}(\Delta R_6)$, or
- $\Delta R_0 = 0$ and $\Delta R_6 \in \{\texttt{81008102}, \texttt{8000840a}\}$ and $\Delta L_6 = \{L\}(\Delta R_6)$.

For three steps, the best boomerangs have probability 2^{-12}, using the single-step trails with the highest probability of Type 1a for the top trail, and a similar trail mirrored vertically and starting from a bottom difference of $(\Delta L_9, \Delta R_9) = (\texttt{83008302}, \texttt{81008102})$. Similarly, we obtain from the combination of the trails of Type 2a and Type 1a boomerangs with probability of 2^{-44} over four steps. Over five steps, the highest probability of a boomerang with fixed trail results from combining a trail of Type 2a with the highest probability at the top with a trail of Type 2b with the highest probability at the bottom.

NEAR-OPTIMAL DIFFERENTIAL TRAILS. Boomerangs that employ a single trail are of limited expressiveness as one can notice a strong differential clustering effect in Sparx. For boomerangs, they are particularly strong in the switching rounds. Our purpose was to find good boomerangs of five steps, where we focused on the free-middle approach. We used the best trails of Type 1b and Type 2a as top and Type 1a and Type 2b as bottom trails as a base to study their probability empirically over a feasible subset of the three steps in the middle. Moreover, our automated search for optimal differential trails yielded many near-optimal differentials with probability slightly smaller than that of the optimal ones; as one could anticipate, this small change in the probability stemmed from the fact that bits adjacent to the active bits in the optimal differentials were also active in the near-optimal ones, mainly in the first or the last round. Hence, we also considered those near-optimal trails in our investigation of potential start and end differences for boomerangs. The subset of our results is given in Table 34. We used a variant of them for our rectangle attack in Section 7.5.

7.4 TRUNCATED-DIFFERENTIAL ATTACK ON 16-ROUND SPARX-64/128

TRUNCATED DIFFERENTIAL ATTACKS. Truncated Differentials were firstly introduced by Knudsen [201] in 1994. In a nutshell, truncated differential cryptanalysis

Table 33: Best found boomerangs on step-reduced Sparx-64/128; for up to three steps, we verified them experimentally with 100 random keys and 2^{20} random pairs each. Values in parentheses are products of the empirical probabilities over the three steps in the middle from Table 34 with the theoretical probabilities over the remaining step(s).

#Steps	Input difference		Output difference		h_w	
s	ΔL_0	ΔR_0	ΔL_{3s}	ΔR_{3s}	theor.	empiric.
1	`00000000`	`00400000`	`83008302`	`81008102`	0	0
2	`00000000`	`28000010`	`8000840a`	`00000000`	6	5.11
2	`00000000`	`28000010`	`81008102`	`00000000`	6	5.16
2	`00000000`	`28000010`	`850a9520`	`00000000`	6	5.31
3	`00000000`	`28000010`	`83008302`	`81008102`	12	10.55
3	`00000000`	`28000010`	`8a048e0e`	`8000840a`	12	11.43
4	`02110a04`	`02110a04`	`83008302`	`81008102`	44	(40.34)
5	`28000010`	`28000010`	`2a102a10`	`850a9520`	78	(68.54)
5	`02110a04`	`02110a04`	`2a102a10`	`850a9520`	76	(72.18)

is a generalisation of differential cryptanalysis, were an attacker leaves parts of the differences unspecified, allowing that parts to take all possible values. This potentially improves the probability of an attack. More details on truncated differentials can be found in Chapter 2.4.3.

HIGH-LEVEL VIEW. In the following, we describe a truncated-differential attack on 16-round Sparx-64/128. On a high level, the Feistel-like structure of Sparx allows generic trails that pass through almost two steps with only one active branch. The core observation of our attack is the existence of Type 1d differential trails, *i.e.,* trails that have an inactive branch before and after a step with probability $\gg 2^{-32}$. One such trail is illustrated in Table 35. The trail is truncated after Round 9; thereupon, its precise differences are irrelevant as long as it will cancel in the right branch after the linear layer, and the branch incorporating a zero-difference can propagate through two further steps (*i.e.,* Rounds 13-18 in Table 35). Thus, an adversary can observe that only a single branch will be active after five steps; Note that the final linear layer can then easily be inverted. On the downside, the probability of truncated trails must exceed 2^{-32} for a useful distinguisher.
We employ Property 2 at the plaintext side to reduce the number of steps to trace through, and to ensure a sufficient probability of the differential. Accordingly, we obtain the round keys of Round 3, 4, and the one for the right branch of Round 5 from guessing only 64 bits of key material. At the ciphertext side, we choose structures of 2^{32} texts, such that all texts in a structure have a constant value in the right branch, and iterate over all values on the left branch through Rounds 16-18. In the

Table 34: Experimental probabilities of free-middle boomerangs over three steps. Each value represents $-\log_2(p)$, where p is the average probability of correct quartets from 100 test runs of random independent keys with 2^{30} random text pairs each.

	$(\Delta L_0, \Delta R_0)$	
$(\Delta L_9, \Delta R_9)$	(00000000,80008000)	(00000000,81008102)
(80008000,80008000)	20.18	26.54
(83008302,81008102)	16.34	22.74
(40404000,00400000)	27.21	30.32

following, we mount a chosen-ciphertext attack on 16-round Sparx-64/128 covering Rounds 3 through 18; the used differential trail is given in Table 35.

STRUCTURES AND SETS. We choose 2^m structures of 2^{32} ciphertexts each from a base text $S^0_{18} = (L_{18}, R_{18})$, and $2^{32} - 1$ derived texts $S^i_{18} = (L^i_{18}, R_{18})$ from iterating over all 2^{32} values L_{18}, and derive 2^{32} ciphertexts $C^i \leftarrow \mathcal{L}(S^i)$ that form the structure. Since we employ all 2^{32} possible values for the right branch of Rounds 16 to 18, their 2^{63} pairs will form all possible differences in this branch about 2^{31} times at any point until the end of Round 12, *i.e.*, Δ_{12}. From experiments, we observed that the truncated differential (80008000,80008000) leads to (00000000,********) with probability $2^{-17.36}$. Hence, there is a subset of *good* differences Δ_{12} that can lead to (80008000,80008000) with this accumulated probability. Since we have 2^{31} pairs for each such Δ_{12}, we expect that there are about $2^{31-17.36} \approx 2^{13.64}$ pairs with $\Delta_9 =$ (80008000,80008000), and $2^{13.64-6-5} = 2^{2.64}$ pairs that follow our trail up to Δ_5. We have approximately 2^{63} pairs in a structure that have our desired difference with probability 2^{-64}, so we expect 2^{-1} false positive pairs from a structure.

EXPERIMENTAL VERIFICATION OF THE DISTINGUISHER. We verified a variant of our distinguisher experimentally using 100 random keys and 2^{32} random pairs. For practicality, we considered it in encryption direction, *i.e.*, we chose random pairs with start difference $(\Delta L_5, \Delta R_5) =$ (00000000, 0a204205), encrypted them to the states after Round 18 and inverted the final linear layer. On average, we obtained $2^{3.75}$ pairs with zero difference in the right branch, which corresponds to a probability of $2^{3.75-32} = 2^{-28.25}$, which is close to the theoretical expected $2^{-28.36}$.

ATTACK STEPS. Using Property 5, we define a linear function $F : \{0,1\}^{32} \times \{0,1\}^{32} \to \{0,1\}^{64}$ with rank $n - 1 = 63$, so that $F(\Delta) = \{0\}^{64}$ for $\Delta =$ (00000000, 0a204205). The attack consists of the following steps:

1. Construct 2^m structures as described above. For each structure, request the corresponding 2^{32} plaintexts P^i from a 16-round decryption oracle.
2. Initialize a list $\mathcal{K}$ of 2^{64} key counters.
3. For each of the 2^{64} guesses of $K^0_2, K^1_2, K^2_2, K^3_2$:

Table 35: The truncated differential trail through 16 rounds. A $*$ symbol marks a truncated difference which can take any possible value.

Rd. i	ΔL_i	ΔR_i	h_w		Rd. i	ΔL_i	ΔR_i	h_w	
					10	********	********	?	?
2	********	********	–	–	11	********	********	?	?
3	********	********	–	–	12	********	********	?	?
$\mathcal{L}$	00000000	********	–	–	$\mathcal{L}$	00000000	********	0	0
4	00000000	********	–	–	13	00000000	********	0	?
5	00000000	0a204205	0	–	14	00000000	********	0	?
6	00000000	02110a04	0	5	15	00000000	********	0	?
$\mathcal{L}$	02110a04	00000000	0	0	$\mathcal{L}$	********	00000000	0	0
7	28000010	00000000	4	0	16	********	00000000	?	0
8	00400000	00000000	2	0	17	********	00000000	?	0
9	80008000	00000000	0	0	18	********	00000000	?	0
$\mathcal{L}$	80008000	80008000	0	0	$\mathcal{L}$	********	********	0	0

3.1 Re-encrypt all plaintexts over one round until the state after the linear layer of Round 3 and store them in $\mathcal{H}$ according to the values of their left branches. Only consider pairs that collide in $\mathcal{H}'$, which represent pairs that will collide in L_3 after the application of the linear layer $\mathcal{L}$.

3.2 For all texts, compute (L_3, R_5), apply $F(R_r)$, and store the updated states in $\mathcal{H}$. Discard all pairs that do not collide. For each colliding pair, increment the counter for the current key candidate in $\mathcal{K}$.

4. Output the keys in descending order of their corresponding counters.

COMPLEXITY. The computational complexity results from:

- **Step 1** requires 2^{m+32} 16-round decryption. We assume the computational costs for a decryption and encryption are equal.
- **Step 3a** requires $2^{64} \cdot 2^{m+32} \cdot 1/16 \cdot 2 \approx 2^{m+92}$ encryption equivalents since we consider one out of 16 rounds. From the $\binom{2^{32}}{2} \approx 2^{63}$ pairs of one structure, we expect $2^{63-32} = 2^{31}$ false positive pairs for each structure at this step.
- We approximate the costs for a call to F by those of a call to two SPECKEY rounds since both branches are used. The complexity of **Step 3b** is therefore given by $2^{64} \cdot 2^{31+m} \cdot 4/32 \approx 2^{m+92}$ encryption equivalents on average. We expect about $2^{63-64} = 2^{-1}$ false-positive pairs per structure and key candidate, whereas we have $2^{31-28.36} \approx 2^{2.64}$ correct pairs for the correct key candidate, again per structure.

Table 36: The differential trails for the top (left) and bottom (right) that are used in the 16-round rectangle attack on Sparx-64/128.

Rd. i	ΔL_i	ΔR_i	h_w	
4	28000010		–	–
5	00400000	00400000	2	–
6	80008000	80008000	0	0
$\mathcal{L}$	00000000	80008000	0	0
7	00000000	********	0	–
8	00000000	********	0	–
9	00000000	********	0	–
$\mathcal{L}$	********	00000000	0	–
10	********	00000000	–	0
11	********	00000000	–	0
12	********	00000000	–	0
$\mathcal{L}$	********	********	–	0

Rd. i	ΔL_i	ΔR_i	h_w	
10	00000000	********	0	–
11	00000000	********	0	–
12	00000000	********	0	–
$\mathcal{L}$	02110a04	00000000	0	0
13	********	00000000	–	0
14	********	00000000	–	0
15	********	00000000	–	0
$\mathcal{L}$	80008000	80008000	0	0
16	81008102	81008102	1	1
17	8000840a	8000840a	2	2
18	850a9520	850a9520	4	4
$\mathcal{L}$	2a102a10	850a9520	0	0

The time complexity then sums up to

$$2^{m+32} + 2^{m+92} + 2^{m+92} \approx 2^{m+92} \text{ Encryptions.}$$

The memory complexity stems from storing a byte counter for the current key candidate, *i.e.*, $2^{64} \cdot 8/64 = 2^{61}$ states, plus 2^{32} texts. The data complexity is given by 2^{m+32}. A single structure, *i.e.*, $m = 1$, is sufficient to obtain at least two correct pairs for the correct key.

7.5 RECTANGLE ATTACK ON 16-ROUND SPARX-64/128

BOOMERANG AND RECTANGLE ATTACKS. *Boomerang attacks* were firstly introduced by Wagner [338]. As *Boomerang attacks* are chosen plaintext and adaptively-chosen ciphertext attacks, Biham *et al.* [71] turned them into purely chosen-plaintext attacks and called them *Rectangle Attacks*.
In a nutshell, in a boomerang/rectangle attack an attacker splits the cipher in two parts, $E = E_2 \circ E_1$, such that $E(P) \overset{\text{def}}{=} E_2(E_1(P))$, and then combines two high probable differential trails for each of the sub-ciphers. There are three well known approaches to combine the differential trails in the middle, the Feistel switch, S-box switch and ladder switch [77]. Recently, Cid *et al.* [114] also studied new ways, which they defined as *Boomerang Connectivity Tables*. In the rectangle attack on Sparx-64/128 we applied a ladder switch, but it remains an open problem if the attack can be improved using a *Boomerang Connectivity Table* approach. This would

however require to find an efficient way to compute the *Boomerang Connectivity Table* for a 16-bit modular addition. More details about boomerang and rectangle attacks can be found in Chapter 2.4.3.

HIGH-LEVEL VIEW. In the following we describe a rectangle attack on 16-round SPARX-64/128. The attack starts after the second round of the cipher, *i.e.,* it starts with Round 3. Again, we guess 64 key bits to get through Rounds 3 and 4 and the right branch of Round 5. In total the attack covers then Rounds 3 through 18.

DIFFERENTIAL TRAILS. Table 36 illustrates the employed differential trails. The top trail covers Rounds 3 through 9 and the right part of Rounds 10 to 12 since the right part contains a zero difference which propagates for free through the $\mathcal{A}^3$ box of Rounds 10 to 12. The bottom trail covers Rounds 13 through 18, and the left part of Rounds 10 through 12 in decryption direction. Again, the bottom trail has a zero difference in that part in the bottom trail which propagates for free through the $\mathcal{A}^3$ box until the begin of Round 10.

EXPERIMENTAL VERIFICATION OF THE MIDDLE ROUNDS. We experimentally verified the boomerang switch in the middle. From 100 experiments with random keys and 2^{26} independently at random chosen pairs (P, P') with difference $\alpha = (\texttt{80008000}, \texttt{80008000})$, encrypted through three steps to (C, C'), applied the δ-shift $(\texttt{80008000}, \texttt{80008000})$ to obtain (D, D'), decrypted those back to (Q, Q'), and counted the number of times that $Q \oplus Q' = \alpha$. We observed an average probability of approximately $2^{-20.18}$. So, for the correct key, we obtain a probability of approximately $(\widehat{p}\widehat{q})^2 \approx \left(2^{-2}\right)^2 \cdot 2^{-20.18} \cdot \left(2^{-14}\right)^2 \approx 2^{-52.18}$ for a valid quartet.

ATTACK PROCEDURE. Choose a linear function $F : \{0,1\}^{64} \to \{0,1\}^{64}$ of rank 63 such that $F(\Delta L_4 \| \Delta R_5) = 0^{64}$. The attack consists of the following steps:

1. Initialise a list of key counters $\mathcal{K}$ to zero, for all 2^{64} possible values for the round keys of Round 2. Initialise two empty hash maps $\mathcal{Q}$ and $\mathcal{R}$.

2. Choose 2^m ciphertext pairs (C, D) with difference $(\texttt{2a102a10}, \texttt{850a9520})$, and ask for their corresponding plaintexts (P, Q). Store the pairs into $\mathcal{H}$ indexed by P.

3. For each of the 2^{64} guesses of $(K_2^0, K_2^1, K_2^2, K_2^3)$:

   3.1 Partially re-encrypt all plaintext pairs (P, Q) to their corresponding states (L_4^P, R_5^P) and (L_4^Q, R_5^Q).

   3.2 Apply $F((L_4, R_5))$ to all states and store the corresponding outputs $(\widehat{L}_4^P, \widehat{R}_5^P)$ and $(\widehat{L}_4^Q, \widehat{R}_5^Q)$ into a hash table $\mathcal{Q}$. Only consider pairs $p = (\widehat{L}_4^P, \widehat{R}_5^P)$, $q = (\widehat{L}_4^Q, \widehat{R}_5^Q)$, $p' = (\widehat{L}_4^{P'}, \widehat{R}_5^{P'})$, $q' = (\widehat{L}_4^{Q'}, \widehat{R}_5^{Q'})$ that collide in either $(p, q) = (p', q')$ or $(p, q) = (q', p')$ and discard all further quartets. We expect $2^{2m} \cdot 2^{2 \cdot -64} \approx 2^{2m-128}$ quartets on average.

   3.3 If a quartet survives, increment the counter for the current key guess. Choose a plaintext pair with our desired difference – w.l.o.g., (p, p') –

from the current quartet, and check for all remaining key bits if it follows our path until Round 6. If yes, encrypt it further round-wise until Round 9. If all round-wise checks pass, check for p if it encrypts to ciphertext C. If yes, test again for (q, q') and output the key candidate if it also matches.

4. If no key candidate has been returned, return $\perp$.

For $m = 58.6$ pairs, we can expect $(2^m \widehat{p}\widehat{q})^2/2^n \approx 2^{117.2} \cdot 2^{-52.18}/2^{64} \approx 2$ valid quartets for the correct key guess. In contrast, we can expect $2^{117.2-2\cdot 64} = 2^{-10.8}$ quartets for a wrong key guess.

COMPLEXITY. The computational complexity results from:

- **Step 2** requires $2 \cdot 2^m = 2 \cdot 2^{58.6} \approx 2^{59.6}$ 16-round decryptions. We assume that the computational costs for a decryption and encryption are equal.
- **Steps 3a and 3b** require $2^{64} \cdot 2 \cdot 2^m \cdot 6/32 = 2^{64} \cdot 2 \cdot 2^{58.6} \cdot 6/32 \approx 2^{122.2}$ encryption equivalents since we consider five out of 32 SPECKEY rounds in the 16-round cipher for re-encryption and approximate the costs for computing F by the costs of a SPECKEY round.
- **Step 3b** will require $2^{64} \cdot 2 \cdot 2^m = 2^{m+65}$ memory accesses (MAs) and comparisons.
- **Step 3c** will require at most $2^{64} \cdot 2^{2m-128} \cdot 2^{64} \approx 2^{117.2}$ encryption equivalents to identify the correct key.

Hence, the computations are upper bounded by approximately

$$2^{59.6} + 2^{122.2} \approx 2^{122.2} \text{ encryptions} \quad \text{and} \quad 2^{59.6} + 2^{123.6} \approx 2^{123.6} \text{ MAs.}$$

The data complexity is upper bounded by $2^{59.6}$ chosen ciphertexts. The memory complexity is upper bounded by storing at most $4 \cdot 2^{59.6}$ states at a time, which is equivalent to storing approximately $2^{61.6}$ states.

7.6 CONCLUSION

In this chapter, we analyse the lightweight cipher SPARX-64/128 against several cryptographic attacks based on differential cryptanalysis. SPARX has been introduced at ASIACRYPT'16 alongside with the *long trail* design strategy, that firstly allows designers to prove security bounds against differential and linear cryptanalysis for an ARX-based cipher. In our work, we therefore not only analyse SPARX but also can give an analysis of this new design strategy.
We present two standard differential attacks using truncated differentials and rectangle attacks on 16-round SPARX-64/128. The former attack builds upon a nine-round (three-step) differential trail that is extended by a six-round (two-step) truncated trail. Adopting the observation by Abdelkhalek *et al.* [2], we can turn the distinguishers in a 16-round chosen-ciphertext attack and recover the round keys

by just guessing 64-bit of the key material. Our truncated differential attack requires approximately 2^{32} chosen ciphertexts, about 2^{32} states, and approximately 2^{93} encryption equivalents. Our proposed rectangle attack exploits the Feistel structure of SPARX using differential trails with inactive branches over their middle step.
We stress that our attacks do not threaten the security of SPARX-64/128, but provide deeper insights in its security against attacks in the single-key setting. We can observe a strong clustering effects of many differential trails in our studies and exploit them in all our attacks; it remains subject to further studies to employ them for further rounds. For public verification and future works, our trails, tests, and implementations of SPARX-64/128 are published into the public domain[¶].

[¶] `https://github.com/TheBananaMan/sparx-differential-attacks`

8 CONCLUSION

In this thesis, we have presented novel research contributions in the area of lightweight cryptography. We analysed several lightweight and efficient symmetric-key primitives and further proposed a new technique in the cryptanalysis of tweakable block ciphers. Moreover, we studied the foundations of differential cryptanalysis and provided an in-depth analysis of all 4-bit S-boxes, that are one of the main building blocks of lightweight Substitution Permutation Networks. In the following, we give an short summary of the contributions in this thesis and we discuss open problems and future work in the research area of lightweight cryptography.

8.1 SUMMARY OF CONTRIBUTIONS

In the first part of this thesis, we presented novel research in the foundations of symmetric-key cryptography. Chapter 3 studies the effects of many differential trails in different lightweight block cipher designs. We showed that there exists a significant gap between single differential trails and differentials, that ignore the intermediate differences, and therefore allow for higher differential probabilities. We can show that this gap is significantly higher in some recently proposed lightweight block cipher design strategies. We conclude that designers have to be careful when claiming security based on single differential trails, especially by using some aggressive block cipher design strategies with very small security margins.
Chapter 4 focused on low-energy 4-bit S-boxes. Many devices in resource constrained environments, the Internet of Things (IoT) in general, are powered by batteries. Especially some devices such as medical implants operate on a tight energy/power budget. Security and privacy is crucial in the communication channels of those devices. While in recent lightweight ciphers, the focus was on improving the area/power/latency, we analyse low-energy designs. In particular we analyse all possible 4-bit S-boxes. We propose two optimal block cipher design strategies for low-energy consumption, based on PRESENT and PRINCE-like designs, and further recommend a selection of low-energy S-boxes.

The second part of this thesis focused on the cryptanalysis of symmetric-key primitives. In Chapter 5 we presented a novel technique for analysing tweakable block ciphers. We consider for the first time the effect of using the tweak in zero-correlation linear attacks. We show that using this technique we can get distinguishers with more rounds. We turn the zero-correlation distinguisher into integral distinguishers to reduce the data complexity and mount key-recovery attacks on the recently proposed tweakable block ciphers Qarma, Mantis and Skinny.
Moreover, Chapter 6 presented related-tweakey impossible differential attacks on a round-reduced version of the tweakable block cipher Skinny-64/128. Skinny is a recently proposed lightweight tweakable block cipher that intends to offer an alternative to the NSA designs Simon and Speck. In our attack we exploited the slow diffusion of Skinny and some features of the tweakey schedule. We conclude that despite using some tricks to extend the number of rounds in the key-recovery, Skinny still offers a large security margin.
Finally, in Chapter 7 we analysed the recently proposed block cipher Sparx. Sparx has been designed according to the long-trail strategy, which allows to provide provable upper bounds for differential and linear attacks. We provided an extensive analysis on the differential effects of Sparx and we showed truncated-differential and rectangle attacks on reduced-round versions of Sparx. We concluded that Sparx still offers a large enough security margin, but still further third-party cryptanalysis is required.

8.2 OPEN PROBLEMS AND FUTURE WORK

While we already discussed some of the open problems and ideas for future research in the individual chapters, we want to provide a summary of open research problems in the following.
The analysis of block ciphers against differential cryptanalysis started in the early 1990 with the publication of differential cryptanalysis by Biham and Shamir [73]. Since then it has become one of the major tools in the analysis of symmetric-key primitives, and many extensions have been proposed. While we analysed several lightweight symmetric-key primitives in Chapter 3, there are many more primitives that need a more detailed analysis regarding differential cryptanalysis. Moreover, the theory of differential cryptanalysis is well studied [87, 88, 306]. However, an interesting research question would be to study the *Hypothesis of stochastic equivalence* [215] in more details for several recently proposed block cipher designs. Moreover, the key-dependence of several block ciphers in regard to differential cryptanalysis is not well studied. We showed that for ciphers Skinny, Speck and Midori, the later two have an interesting distribution over the keys, which could be a potential direction for further research.
In Chapter 4 we studied energy-efficient block cipher design strategies and provided an detailed overview for 4-bit S-boxes. While we managed to reduce the search space of all 4-bit S-boxes from 2^{64} to $2^{44.25}$ by just considering permutations and we further looked into affine equivalence of S-boxes, it is still a computationally challenging task to analyse all S-boxes. We further showed that when we im-

plement an S-box in an actual cipher design, sometimes the improvement gained by the S-box is significantly reduced in the energy consumption of the whole block cipher. Therefore, it would be an interesting research question to further look into energy-efficient block cipher designs as a whole, for example by also considering the diffusion layers. In that context non-linear diffusion layers as suggested by Liu *et al.* [231] might be a direction to explore more energy-efficient block cipher designs.

Chapter 5 presents a novel technique in the cryptanalysis of tweakable block ciphers, by incorporating the tweak when searching for zero-correlation linear distinguishers. While we demonstrate the attack by applying it to the block ciphers Qarma, Mantis and Skinny, there are further block ciphers published that were designed using the Tweakey framework. It would be of interest if the current best attacks on Deoxys [189], Joltik [186], Kiasu [187] can be improved. Moreover, very recently ForkAES [15] was published, again based on the Tweakey framework and Kiasu. It would be of interest to see if our technique can lead to improved attacks on this cipher.

Chapter 6 shows related-tweakey impossible differential attacks on a round-reduced version of the tweakable block cipher Skinny-64/128. While there is already a lot of third party cryptanalysis for Skinny [230, 295, 316, 328], it remains an open task for cryptanalysts to come up with new attacks techniques, for example as we showed in Chapter 5. Moreover, it would be interesting to use the block cipher Skinny in a mode for authenticated encryption, for example as required in the upcoming lightweight block cipher competition by NIST [43].

In Chapter 7 we analyse the block cipher Sparx and the long-trail design strategy. As Sparx has been recently proposed further third party cryptanalysis is required to strengthen the confidence in the cipher. As ARX-based block ciphers are computationally hard to analyse (*i.e.,* in terms of efficiently finding long distinguishers) it would be of interest to analyse the cipher in more detail regarding yoyo attacks [68], differential-linear attacks [217] and even against standard differential attacks, if better distinguishers can be found (*i.e.,* by combining several trails to a long one as it has been shown successfully in the analysis of Speck by Fu *et al.* [159]).

8.3 PERSONAL CONCLUSIONS

This thesis is a summary of the work that I did during three years of doctoral research while being enrolled in the PhD program in Information Security at Royal Holloway University of London. While it is hard to conclude three years of work in just a few pages, this section focuses on my personal achievements during this three years.

My PhD is definitely the biggest achievement in my professional career. When I started my PhD I immediately felt the need to achieve something, to break a cipher or to find a new technique for cryptanalysis. Little did I know, research takes time. Moreover, research is hard. It took me about the first year in my PhD to realise that I had to change my way of thinking. The easy problems in research are boring to work on. Most of the hard problems are already solved. Finally, there are the *impossible hard* problems, that nobody likes to work on. Solving such an *impossible hard* problem is however very interesting and furthermore often very impactful. So how does one solve an *impossible hard* research problem? After three years in my PhD I can suggest the following strategy:

- One has to break the *impossible hard* problems into little parts. Often, this little parts are then *hard* problems, which again are interesting, and actually possible to solve.
- One has to show some resilience. Research problems are hard, and often it takes several days/weeks/months to achieve something. Don't give up. Keep searching. There is a reason why it is called *re-search*.
- Be creative. Often the best ideas come from unexpected areas. It is very helpful to visit the research seminars of other groups and departments in the university to see on what kind of problems they are working and what techniques they are using to solve them.
- Networking. Conferences and workshops are not exclusively to present your new paper. They are a place to find skilled co-authors. Often by simply discussing your problems, or writing a draft email, one has to put the problem in simple words, that then gives different views at looking at the problem and sometimes even shows the solution.
- Finally, it is my strong believe that one is ready to finish a PhD if one accepts that one understands nothing. This doesn't mean that one knows nothing or can't achieve anything. It means that one has to accept that there will always be a bigger something. Some more experiments to do, another improvement to work on, another paper to write. Research is an ever expanding field.

Ultimately, in the case of the analysis of symmetric-key primitives I can conclude that it will always remains an active field of research, as in general, attacks just get stronger. However, my hope is that our contributions provide designers of symmetric-key primitives with better tools for analysis and and further insights in the security of block cipher designs.

BIBLIOGRAPHY

[1] Abdalla, M., Bjørstad, T.E., Cid, C., Gierlichs, B., Hülsing, A., Luykx, A., Paterson, K.G., Preneel, B., Sadeghi, A.R., Spies, T., Stam, M., Ward, M., Warinschi, B., Watson, G.: Algorithms, key size and protocols report (2018) (February 2018), `http://www.ecrypt.eu.org/csa/documents/D5.4-FinalAlgKeySizeProt.pdf`

[2] Abdelkhalek, A., Tolba, M., Youssef, A.M.: Impossible differential attack on reduced round SPARX-64/128. In: Joye, M., Nitaj, A. (eds.) AFRICACRYPT 17: 9th International Conference on Cryptology in Africa. Lecture Notes in Computer Science, vol. 10239, pp. 135–146. Springer, Heidelberg, Germany, Dakar, Senegal (May 24–26, 2017)

[3] Abdelraheem, M.A., Ågren, M., Beelen, P., Leander, G.: On the distribution of linear biases: Three instructive examples. In: Safavi-Naini, R., Canetti, R. (eds.) Advances in Cryptology – CRYPTO 2012. Lecture Notes in Computer Science, vol. 7417, pp. 50–67. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2012)

[4] Abdelraheem, M.A., Beelen, P., Bogdanov, A., Tischhauser, E.: Twisted polynomials and forgery attacks on GCM. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015, Part I. Lecture Notes in Computer Science, vol. 9056, pp. 762–786. Springer, Heidelberg, Germany, Sofia, Bulgaria (Apr 26–30, 2015)

[5] Abed, F., Fluhrer, S.R., Foley, J., Forler, C., List, E., Lucks, S., McGrew, D., Wenzel, J.: The POET Family of On-Line Authenticated Encryption Schemes. `https://competitions.cr.yp.to/round2/poetv20.pdf` (2015)

[6] Abed, F., Forler, C., Lucks, S.: General overview of the authenticated schemes for the first round of the CAESAR competition. Cryptology ePrint Archive, Report 2014/792 (2014), `http://eprint.iacr.org/2014/792`

[7] Abed, F., List, E., Lucks, S., Wenzel, J.: Differential cryptanalysis of round-reduced Simon and Speck. In: Cid, C., Rechberger, C. (eds.) Fast Software Encryption – FSE 2014. Lecture Notes in Computer Science, vol. 8540, pp. 525–545. Springer, Heidelberg, Germany, London, UK (Mar 3–5, 2015)

[8] AlFardan, N.J., Paterson, K.G.: Lucky thirteen: Breaking the TLS and DTLS record protocols. In: 2013 IEEE Symposium on Security and Privacy. pp. 526–540. IEEE Computer Society Press, Berkeley, CA, USA (May 19–22, 2013)

[9] Alizadeh, J., Aref, M.R., Bagheri, N.: Artemia v1. https://competitions.cr.yp.to/round1/artemiav1.pdf (2014)

[10] Alomair, B.: AVALANCHEv1. https://competitions.cr.yp.to/round1/avalanchev1.pdf (2014)

[11] Amazon: Amazon Web Services (AWS) EC2 On-Demand Pricing. https://aws.amazon.com/ec2/pricing/on-demand/ (2018)

[12] Andreeva, E., Bilgin, B., Bogdanov, A., Luykx, A., Mendel, F., Mennink, B., Mouha, N., Wang, Q., Yasuda, K.: PRIMATEs v1.02. https://competitions.cr.yp.to/round2/primatesv102.pdf (2014)

[13] Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Tischhauser, E., Yasuda, K.: Parallelizable and authenticated online ciphers. In: Sako, K., Sarkar, P. (eds.) Advances in Cryptology – ASIACRYPT 2013, Part I. Lecture Notes in Computer Science, vol. 8269, pp. 424–443. Springer, Heidelberg, Germany, Bengalore, India (Dec 1–5, 2013)

[14] Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Tischhauser, E., Yasuda, K.: AES-COPA v.2. https://competitions.cr.yp.to/round2/aescopav2.pdf (2015)

[15] Andreeva, E., Reyhanitabar, R., Varici, K., Vizár, D.: Forking a blockcipher for authenticated encryption of very short messages. Cryptology ePrint Archive, Report 2018/916 (2018), https://eprint.iacr.org/2018/916

[16] Ankele, Ralph, Ankele, Robin: Software benchmarking of the 2$^{\text{nd}}$ round CAESAR candidates. Cryptology ePrint Archive, Report 2016/740 (2016), http://eprint.iacr.org/2016/740

[17] Ankele, R., Banegas, G., Boss, E., Božilov, D., Friedberger, S., Lacharité, M.S., Li, C., Martinoli, M., Minelli, M., Minihold, M., Panny, L., Rosie, R., Šijačić, D., Soria-Vázquez, E., Wang, J.: Technical report on implementations for IoT & cloud. Tech. rep., European Integrated Research Training Network on Advanced Cryptographic Technologies for the Internet of Things and the Cloud (2018)

[18] Ankele, R., Banegas, G., Boss, E., Božilov, D., Friedberger, S., Lacharité, M.S., Li, C., Martinoli, M., Minelli, M., Minihold, M., Rosie, R., Šijačić, D., Soria-Vázquez, E.: Technical report on first designs for IoT & cloud. Tech. rep., European Integrated Research Training Network on Advanced Cryptographic Technologies for the Internet of Things and the Cloud (2017)

[19] Ankele, R., Banik, S., Chakraborti, A., List, E., Mendel, F., Sim, S.M., Wang, G.: Related-key impossible-differential attack on reduced-round SKINNY. Cryptology ePrint Archive, Report 2016/1127 (2016), http://eprint.iacr.org/2016/1127

[20] Ankele, R., Banik, S., Chakraborti, A., List, E., Mendel, F., Sim, S.M., Wang, G.: Related-key impossible-differential attack on reduced-round Skinny. In: Gollmann, D., Miyaji, A., Kikuchi, H. (eds.) ACNS 17: 15th International Conference on Applied Cryptography and Network Security. Lecture Notes in Computer Science, vol. 10355, pp. 208–228. Springer, Heidelberg, Germany, Kanazawa, Japan (Jul 10–12, 2017)

[21] Ankele, R., Böhl, F., Friedberger, S.: MergeMAC: A MAC for authentication with strict time constraints and limited bandwidth. In: Preneel, B., Vercauteren, F. (eds.) ACNS 18: 16th International Conference on Applied Cryptography and Network Security. Lecture Notes in Computer Science, vol. 10892, pp. 381–399. Springer, Heidelberg, Germany, Leuven, Belgium (Jul 2–4, 2018)

[22] Ankele, R., Böhl, F., Friedberger, S.: MergeMAC: A MAC for authentication with strict time constraints and limited bandwidth. Cryptology ePrint Archive, Report 2018/342 (2018), `https://eprint.iacr.org/2018/342`

[23] Ankele, R., Dobraunig, C., Guo, J., Lambooij, E., Leander, G., Todo, Y.: Zero-correlation attacks on the TWEAKEY framework. IACR Transactions on Symmetric Cryptology 2018(4) (Sep 2018), submitted

[24] Ankele, R., Kölbl, S.: Mind the gap - a closer look at the security of block ciphers against differential cryptanalysis. In: Cid, C., Jacobson, M.J. (eds.) Selected Areas in Cryptography – SAC 2018. Springer International Publishing, Cham (2018)

[25] Ankele, R., Kölbl, S.: Mind the gap - a closer look at the security of block ciphers against differential cryptanalysis. In: Cid, C., Jacobson Jr., M.J. (eds.) Selected Areas in Cryptography – SAC 2018. pp. 163–190. Springer International Publishing, Cham (2019)

[26] Ankele, R., List, E.: Differential cryptanalysis of round-reduced Sparx-64/128. In: Preneel, B., Vercauteren, F. (eds.) ACNS 18: 16th International Conference on Applied Cryptography and Network Security. Lecture Notes in Computer Science, vol. 10892, pp. 459–475. Springer, Heidelberg, Germany, Leuven, Belgium (Jul 2–4, 2018)

[27] Ankele, R., List, E.: Differential cryptanalysis of round-reduced Sparx-64/128. Cryptology ePrint Archive, Report 2018/332 (2018), `https://eprint.iacr.org/2018/332`

[28] Anonymous: Rc4 source code (September 1994), `http://cypherpunks.venona.com/date/1994/09/msg00304.html`

[29] Aoki, K., Ichikawa, T., Kanda, M., Matsui, M., Moriai, S., Nakajima, J., Tokita, T.: Camellia: A 128-bit block cipher suitable for multiple platforms - Design and analysis. In: Stinson, D.R., Tavares, S.E. (eds.) SAC 2000: 7th Annual International Workshop on Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 2012, pp. 39–56. Springer, Heidelberg, Germany, Waterloo, Ontario, Canada (Aug 14–15, 2001)

[30] Ashur, T., Bar-On, A., Dunkelman, O.: Cryptanalysis of GOST2. IACR Transactions on Symmetric Cryptology 2017(1), 203–214 (2017)

[31] Aumasson, J.P., Bernstein, D.J.: SipHash: A fast short-input PRF. In: Galbraith, S.D., Nandi, M. (eds.) Progress in Cryptology - INDOCRYPT 2012: 13th International Conference in Cryptology in India. Lecture Notes in Computer Science, vol. 7668, pp. 489–508. Springer, Heidelberg, Germany, Kolkata, India (Dec 9–12, 2012)

[32] Aumasson, J.P., Jovanovic, P., Neves, S.: Analysis of NORX: Investigating differential and rotational properties. In: Aranha, D.F., Menezes, A. (eds.) Progress in Cryptology - LATINCRYPT 2014: 3rd International Conference on Cryptology and Information Security in Latin America. Lecture Notes in Computer Science, vol. 8895, pp. 306–324. Springer, Heidelberg, Germany, Florianópolis, Brazil (Sep 17–19, 2015)

[33] Aumasson, J.P., Jovanovic, P., Neves, S.: NORX v2.0. `https://competitions.cr.yp.to/round2/norxv20.pdf` (2015)

[34] Avanzi, R.: The QARMA block cipher family. IACR Transactions on Symmetric Cryptology 2017(1), 4–44 (2017)

[35] Babbage, S., Dodd, M.: The stream cipher MICKEY 2.0 (2006), `http://www.ecrypt.eu.org/stream/p3ciphers/mickey/mickey_p3.pdf`

[36] Bahack, L.: Julius: Secure Mode of Operation for Authenticated Encryption Based on ECB and Finite Field multiplications. `https://competitions.cr.yp.to/round1/juliusv10.pdf` (2014)

[37] Banik, S., Bogdanov, A., Isobe, T., Shibutani, K., Hiwatari, H., Akishita, T., Regazzoni, F.: Midori: A block cipher for low energy. In: Iwata, T., Cheon, J.H. (eds.) Advances in Cryptology – ASIACRYPT 2015, Part II. Lecture Notes in Computer Science, vol. 9453, pp. 411–436. Springer, Heidelberg, Germany, Auckland, New Zealand (Nov 30 – Dec 3, 2015)

[38] Banik, S., Bogdanov, A., Regazzoni, F.: Atomic-AES: A compact implementation of the AES encryption/decryption core. In: Dunkelman, O., Sanadhya, S.K. (eds.) Progress in Cryptology - INDOCRYPT 2016: 17th International Conference in Cryptology in India. Lecture Notes in Computer Science, vol. 10095, pp. 173–190. Springer, Heidelberg, Germany, Kolkata, India (Dec 11–14, 2016)

[39] Banik, S., Bogdanov, A., Regazzoni, F.: Exploring energy efficiency of lightweight block ciphers. In: Dunkelman, O., Keliher, L. (eds.) SAC 2015: 22nd Annual International Workshop on Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 9566, pp. 178–194. Springer, Heidelberg, Germany, Sackville, NB, Canada (Aug 12–14, 2016)

[40] Banik, S., Pandey, S.K., Peyrin, T., Sasaki, Y., Sim, S.M., Todo, Y.: GIFT: A small present - towards reaching the limit of lightweight encryption. In: Fischer, W., Homma, N. (eds.) Cryptographic Hardware and Embedded Systems

– CHES 2017. Lecture Notes in Computer Science, vol. 10529, pp. 321–345. Springer, Heidelberg, Germany, Taipei, Taiwan (Sep 25–28, 2017)

[41] Barkan, E., Biham, E., Keller, N.: Instant ciphertext-only cryptanalysis of GSM encrypted communication. In: Boneh, D. (ed.) Advances in Cryptology – CRYPTO 2003. Lecture Notes in Computer Science, vol. 2729, pp. 600–616. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2003)

[42] Barkan, E., Biham, E., Keller, N.: Instant ciphertext-only cryptanalysis of GSM encrypted communication. Journal of Cryptology 21(3), 392–429 (Jul 2008)

[43] Bassham, L., Çalik, c., Mouha, N., McKay, K., Sönmez Turan, M.: Draft Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process. https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/Draft-LWC-Submission-Requirements-April2018.pdf (April 2018)

[44] Batina, L., Jakobovic, D., Mentens, N., Picek, S., Piedra, A.D.L., Sisejkovic, D.: S-box pipelining using genetic algorithms for high-throughput AES implementations: How fast can we go? In: Meier, W., Mukhopadhyay, D. (eds.) Progress in Cryptology - INDOCRYPT 2014: 15th International Conference in Cryptology in India. Lecture Notes in Computer Science, vol. 8885, pp. 322–337. Springer, Heidelberg, Germany, New Delhi, India (Dec 14–17, 2014)

[45] Bay, A., Mashatan, A., Vaudenay, S.: A related-key attack against multiple encryption based on fixed points. In: Obaidat, M.S., Sevillano, J.L., Filipe, J. (eds.) ICETE 2011. Communications in Computer and Information Science, vol. 314, pp. 264–280. Springer (2011)

[46] Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK families of lightweight block ciphers. Cryptology ePrint Archive, Report 2013/404 (2013), http://eprint.iacr.org/2013/404

[47] Becker, M., Desoky, A.: A study of the dvd content scrambling system (css) algorithm. In: Proceedings of the Fourth IEEE International Symposium on Signal Processing and Information Technology, 2004. pp. 353–356 (Dec 2004)

[48] Beierle, C., Canteaut, A., Leander, G., Rotella, Y.: Proving resistance against invariant attacks: How to choose the round constants. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology – CRYPTO 2017, Part II. Lecture Notes in Computer Science, vol. 10402, pp. 647–678. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 20–24, 2017)

[49] Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: Cryptanalysis competition. https://sites.google.com/site/skinnycipher/cryptanalysis-competition (2016)

[50] Beierle, C., Jean, J., Kölbl, S., Leander, G., Moradi, A., Peyrin, T., Sasaki, Y., Sasdrich, P., Sim, S.M.: The SKINNY family of block ciphers and its low-latency variant MANTIS. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology –

CRYPTO 2016, Part II. Lecture Notes in Computer Science, vol. 9815, pp. 123–153. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 14–18, 2016)

[51] Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: Koblitz, N. (ed.) Advances in Cryptology – CRYPTO'96. Lecture Notes in Computer Science, vol. 1109, pp. 1–15. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 1996)

[52] Bellare, M., Guérin, R., Rogaway, P.: XOR MACs: New methods for message authentication using finite pseudorandom functions. In: Coppersmith, D. (ed.) Advances in Cryptology – CRYPTO'95. Lecture Notes in Computer Science, vol. 963, pp. 15–28. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 27–31, 1995)

[53] Bellare, M., Kilian, J., Rogaway, P.: The security of the cipher block chaining message authentication code. Journal of Computer and System Sciences 61(3), 362–399 (2000)

[54] Bellare, M., Kohno, T., Namprempre, C.: Authenticated encryption in SSH: Provably fixing the SSH binary packet protocol. In: Atluri, V. (ed.) ACM CCS 02: 9th Conference on Computer and Communications Security. pp. 1–11. ACM Press, Washington D.C., USA (Nov 18–22, 2002)

[55] Bellare, M., Namprempre, C.: Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In: Okamoto, T. (ed.) Advances in Cryptology – ASIACRYPT 2000. Lecture Notes in Computer Science, vol. 1976, pp. 531–545. Springer, Heidelberg, Germany, Kyoto, Japan (Dec 3–7, 2000)

[56] Berbain, C., Billet, O., Canteaut, A., Courtois, N., Gilbert, H., Goubin, L., Granboulan, L., Lauradoux, C., Minier, M., Pornin, T., Sibert, H.: Sosemanuk, a fast software-oriented stream cipher (2004), http://www.ecrypt.eu.org/stream/p3ciphers/sosemanuk/sosemanuk_p3.pdf

[57] Bernstein, D.J.: The poly1305-AES message-authentication code. In: Gilbert, H., Handschuh, H. (eds.) Fast Software Encryption – FSE 2005. Lecture Notes in Computer Science, vol. 3557, pp. 32–49. Springer, Heidelberg, Germany, Paris, France (Feb 21–23, 2005)

[58] Bernstein, D.J.: The Salsa20 family of stream ciphers (2005), https://cr.yp.to/snuffle/salsafamily-20071225.pdf

[59] Bernstein, D.J.: ChaCha, a variant of Salsa20 (January 2008), https://cr.yp.to/chacha/chacha-20080120.pdf

[60] Bernstein, D.J.: Caesar: Competition for authenticated encryption: Security, applicability, and robustness. https://competitions.cr.yp.to/caesar.html (2014)

[61] Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: The Keccak SHA-3 submission. https://keccak.team/files/Keccak-submission-3.pdf (January 2011)

[62] Bertoni, G., Daemen, J., Peeters, M., Van Assche, G., Van Keer, R.: CAESAR submission: KETJE v1. https://competitions.cr.yp.to/round1/ketjev1.pdf (2014)

[63] Bertoni, G., Daemen, J., Peeters, M., Van Assche, G., Van Keer, R.: CAESAR submission: KEYAK v2. https://competitions.cr.yp.to/round2/keyakv21.pdf (2015)

[64] Bertoni, G., Macchetti, M., Negri, L., Fragneto, P.: Power-efficient ASIC synthesis of cryptographic sboxes. In: Proceedings of the 14th ACM Great Lakes Symposium on VLSI. pp. 277–281. GLSVLSI '04, ACM, New York, NY, USA (2004), http://doi.acm.org/10.1145/988952.989019

[65] Beyne, T.: Block cipher invariants as eigenvectors of correlation matrices. In: Peyrin, T., Galbraith, S. (eds.) Advances in Cryptology – ASIACRYPT 2018. pp. 3–31. Springer International Publishing, Cham (2018)

[66] Biggs, J.: Hackers release source code for a powerful DDoS app called Mirai. https://techcrunch.com/2016/10/10/hackers-release-source-code-for-a-powerful-ddos-app-called-mirai/?guccounter=1 (2016)

[67] Biham, E., Anderson, R.J., Knudsen, L.R.: Serpent: A new block cipher proposal. In: Vaudenay, S. (ed.) Fast Software Encryption – FSE'98. Lecture Notes in Computer Science, vol. 1372, pp. 222–238. Springer, Heidelberg, Germany, Paris, France (Mar 23–25, 1998)

[68] Biham, E., Biryukov, A., Dunkelman, O., Richardson, E., Shamir, A.: Initial observations on Skipjack: Cryptanalysis of Skipjack-3XOR (invited talk). In: Tavares, S.E., Meijer, H. (eds.) SAC 1998: 5th Annual International Workshop on Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 1556, pp. 362–376. Springer, Heidelberg, Germany, Kingston, Ontario, Canada (Aug 17–18, 1999)

[69] Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. In: Stern, J. (ed.) Advances in Cryptology – EUROCRYPT'99. Lecture Notes in Computer Science, vol. 1592, pp. 12–23. Springer, Heidelberg, Germany, Prague, Czech Republic (May 2–6, 1999)

[70] Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. Journal of Cryptology 18(4), 291–311 (Sep 2005)

[71] Biham, E., Dunkelman, O., Keller, N.: The rectangle attack - rectangling the Serpent. In: Pfitzmann, B. (ed.) Advances in Cryptology – EUROCRYPT 2001. Lecture Notes in Computer Science, vol. 2045, pp. 340–357. Springer, Heidelberg, Germany, Innsbruck, Austria (May 6–10, 2001)

[72] Biham, E., Dunkelman, O., Keller, N.: New results on boomerang and rectangle attacks. In: Daemen, J., Rijmen, V. (eds.) Fast Software Encryption – FSE 2002. Lecture Notes in Computer Science, vol. 2365, pp. 1–16. Springer, Heidelberg, Germany, Leuven, Belgium (Feb 4–6, 2002)

[73] Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. In: Menezes, A.J., Vanstone, S.A. (eds.) Advances in Cryptology – CRYPTO'90. Lecture Notes in Computer Science, vol. 537, pp. 2–21. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 11–15, 1991)

[74] Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. Journal of Cryptology 4(1), 3–72 (Jan 1991)

[75] Biryukov, A., Derbez, P., Perrin, L.: Differential analysis and meet-in-the-middle attack against round-reduced TWINE. In: Leander, G. (ed.) Fast Software Encryption – FSE 2015. Lecture Notes in Computer Science, vol. 9054, pp. 3–27. Springer, Heidelberg, Germany, Istanbul, Turkey (Mar 8–11, 2015)

[76] Biryukov, A., Khovratovich, D.: PAEQ v1. `https://competitions.cr.yp.to/round1/paeqv1.pdf` (2014)

[77] Biryukov, A., Khovratovich, D.: Related-key cryptanalysis of the full AES-192 and AES-256. In: Matsui, M. (ed.) Advances in Cryptology – ASIACRYPT 2009. Lecture Notes in Computer Science, vol. 5912, pp. 1–18. Springer, Heidelberg, Germany, Tokyo, Japan (Dec 6–10, 2009)

[78] Biryukov, A., Lano, J., Preneel, B.: Cryptanalysis of the alleged SecurID hash function. In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003: 10th Annual International Workshop on Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 3006, pp. 130–144. Springer, Heidelberg, Germany, Ottawa, Ontario, Canada (Aug 14–15, 2004)

[79] Biryukov, A., Leurent, G., Perrin, L.: Cryptanalysis of Feistel networks with secret round functions. In: Dunkelman, O., Keliher, L. (eds.) SAC 2015: 22nd Annual International Workshop on Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 9566, pp. 102–121. Springer, Heidelberg, Germany, Sackville, NB, Canada (Aug 12–14, 2016)

[80] Biryukov, A., Leurent, G., Roy, A.: Cryptanalysis of the "kindle" cipher. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012: 19th Annual International Workshop on Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 7707, pp. 86–103. Springer, Heidelberg, Germany, Windsor, Ontario, Canada (Aug 15–16, 2013)

[81] Biryukov, A., Perrin, L.: State of the art in lightweight symmetric cryptography. Cryptology ePrint Archive, Report 2017/511 (2017), `http://eprint.iacr.org/2017/511`

[82] Biryukov, A., Perrin, L., Udovenko, A.: Reverse-engineering the S-box of streebog, kuznyechik and STRIBOBr1. In: Fischlin, M., Coron, J.S. (eds.) Advances in Cryptology – EUROCRYPT 2016, Part I. Lecture Notes in Computer Science, vol. 9665, pp. 372–402. Springer, Heidelberg, Germany, Vienna, Austria (May 8–12, 2016)

[83] Biryukov, A., Roy, A., Velichkov, V.: Differential analysis of block ciphers SIMON and SPECK. In: Cid, C., Rechberger, C. (eds.) Fast Software Encryption – FSE 2014. Lecture Notes in Computer Science, vol. 8540, pp. 546–570. Springer, Heidelberg, Germany, London, UK (Mar 3–5, 2015)

[84] Biryukov, A., Velichkov, V.: Automatic search for differential trails in ARX ciphers. In: Benaloh, J. (ed.) Topics in Cryptology – CT-RSA 2014. Lecture Notes in Computer Science, vol. 8366, pp. 227–250. Springer, Heidelberg, Germany, San Francisco, CA, USA (Feb 25–28, 2014)

[85] Black, J., Rogaway, P.: CBC MACs for arbitrary-length messages: The three-key constructions. In: Bellare, M. (ed.) Advances in Cryptology – CRYPTO 2000. Lecture Notes in Computer Science, vol. 1880, pp. 197–215. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 20–24, 2000)

[86] Black, J., Rogaway, P.: A block-cipher mode of operation for parallelizable message authentication. In: Knudsen, L.R. (ed.) Advances in Cryptology – EUROCRYPT 2002. Lecture Notes in Computer Science, vol. 2332, pp. 384–397. Springer, Heidelberg, Germany, Amsterdam, The Netherlands (Apr 28 – May 2, 2002)

[87] Blondeau, C., Gérard, B.: On the data complexity of statistical attacks against block ciphers (full version). Cryptology ePrint Archive, Report 2009/064 (2009), `http://eprint.iacr.org/2009/064`

[88] Blondeau, C., Gérard, B.: Multiple differential cryptanalysis: Theory and practice. In: Joux, A. (ed.) Fast Software Encryption – FSE 2011. Lecture Notes in Computer Science, vol. 6733, pp. 35–54. Springer, Heidelberg, Germany, Lyngby, Denmark (Feb 13–16, 2011)

[89] Blondeau, C., Nyberg, K.: Improved parameter estimates for correlation and capacity deviates in linear cryptanalysis. IACR Transactions on Symmetric Cryptology 2016(2), 162–191 (2016), `http://tosc.iacr.org/index.php/ToSC/article/view/570`

[90] Blondeau, C., Nyberg, K.: Improved parameter estimates for correlation and capacity deviates in linear cryptanalysis. IACR Transactions on Symmetric Cryptology 2016(2), 162–191 (Feb 2017), `https://tosc.iacr.org/index.php/ToSC/article/view/570`

[91] Boesgaard, M., Vesterager, M., Christensen, T., Zenner, E.: The stream cipher Rabbit (2004), `http://www.ecrypt.eu.org/stream/p3ciphers/rabbit/rabbit_p3.pdf`

[92] Bogdanov, A., Boura, C., Rijmen, V., Wang, M., Wen, L., Zhao, J.: Key difference invariant bias in block ciphers. In: Sako, K., Sarkar, P. (eds.) Advances in Cryptology – ASIACRYPT 2013, Part I. Lecture Notes in Computer Science, vol. 8269, pp. 357–376. Springer, Heidelberg, Germany, Bengalore, India (Dec 1–5, 2013)

[93] Bogdanov, A., Geng, H., Wang, M., Wen, L., Collard, B.: Zero-correlation linear cryptanalysis with FFT and improved attacks on ISO standards Camellia and CLEFIA. In: Lange, T., Lauter, K., Lisonek, P. (eds.) SAC 2013: 20th Annual International Workshop on Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 8282, pp. 306–323. Springer, Heidelberg, Germany, Burnaby, BC, Canada (Aug 14–16, 2014)

[94] Bogdanov, A., Knežević, M., Leander, G., Toz, D., Varici, K., Verbauwhede, I.: Spongent: A lightweight hash function. In: Preneel, B., Takagi, T. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2011. Lecture Notes in Computer Science, vol. 6917, pp. 312–325. Springer, Heidelberg, Germany, Nara, Japan (Sep 28 – Oct 1, 2011)

[95] Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: An ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2007. Lecture Notes in Computer Science, vol. 4727, pp. 450–466. Springer, Heidelberg, Germany, Vienna, Austria (Sep 10–13, 2007)

[96] Bogdanov, A., Leander, G., Nyberg, K., Wang, M.: Integral and multidimensional linear distinguishers with correlation zero. In: Wang, X., Sako, K. (eds.) Advances in Cryptology – ASIACRYPT 2012. Lecture Notes in Computer Science, vol. 7658, pp. 244–261. Springer, Heidelberg, Germany, Beijing, China (Dec 2–6, 2012)

[97] Bogdanov, A., Rijmen, V.: Linear hulls with correlation zero and linear cryptanalysis of block ciphers. Designs, Codes and Cryptography 70(3), 369–383 (Mar 2014), https://doi.org/10.1007/s10623-012-9697-z

[98] Bogdanov, A., Tischhauser, E., Vejre, P.S.: Multivariate linear cryptanalysis: The past and future of PRESENT. Cryptology ePrint Archive, Report 2016/667 (2016), http://eprint.iacr.org/2016/667

[99] Bogdanov, A., Wang, M.: Zero correlation linear cryptanalysis with reduced data complexity. In: Canteaut, A. (ed.) Fast Software Encryption – FSE 2012. Lecture Notes in Computer Science, vol. 7549, pp. 29–48. Springer, Heidelberg, Germany, Washington, DC, USA (Mar 19–21, 2012)

[100] Bono, S.C., Green, M., Stubblefield, A., Juels, A., Rubin, A.D., Szydlo, M.: Security analysis of a cryptographically-enabled rfid device. In: Proceedings of the 14th Conference on USENIX Security Symposium - Volume 14. pp. 1–1. SSYM'05, USENIX Association, Berkeley, CA, USA (2005), http://dl.acm.org/citation.cfm?id=1251398.1251399

[101] Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knežević, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thomsen, S.S., Yalçin, T.: PRINCE - A low-latency block cipher for pervasive computing applications - extended abstract. In: Wang, X., Sako, K. (eds.) Advances in Cryptology – ASIACRYPT 2012. Lecture Notes in Computer Science, vol. 7658, pp. 208–225. Springer, Heidelberg, Germany, Beijing, China (Dec 2–6, 2012)

[102] Borghoff, J., Knudsen, L.R., Leander, G., Matusiewicz, K.: Cryptanalysis of C2. In: Halevi, S. (ed.) Advances in Cryptology – CRYPTO 2009. Lecture Notes in Computer Science, vol. 5677, pp. 250–266. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 2009)

[103] Bosselaers, A., Vercauteren, F.: Yaes v1. `https://competitions.cr.yp.to/round1/yaesv1.pdf` (2014)

[104] Briceno, M., Goldberg, I., Wagner, D.: A pedagogical implementation of the GSM A5/1 and A5/2 voice privacy encryption algorithms (1999), `http://www.scard.org/gsm/a51.html`

[105] Cannière, C.D., Dunkelman, O., Knežević, M.: KATAN and KTANTAN - a family of small and efficient hardware-oriented block ciphers. In: Clavier, C., Gaj, K. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2009. Lecture Notes in Computer Science, vol. 5747, pp. 272–288. Springer, Heidelberg, Germany, Lausanne, Switzerland (Sep 6–9, 2009)

[106] Canright, D.: A very compact S-box for AES. In: Rao, J.R., Sunar, B. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2005. Lecture Notes in Computer Science, vol. 3659, pp. 441–455. Springer, Heidelberg, Germany, Edinburgh, UK (Aug 29 – Sep 1, 2005)

[107] Canteaut, A.: Differential cryptanalysis of feistel ciphers and differentially uniform mappings. Selected Areas on Cryptography, SAC'97 pp. 172–184 (1997)

[108] Canteaut, A.: Lecture notes on cryptographic boolean functions (March 2016)

[109] Canteaut, A., Fuhr, T., Gilbert, H., Naya-Plasencia, M., Reinhard, J.R.: Multiple differential cryptanalysis of round-reduced PRINCE. In: Cid, C., Rechberger, C. (eds.) Fast Software Encryption – FSE 2014. Lecture Notes in Computer Science, vol. 8540, pp. 591–610. Springer, Heidelberg, Germany, London, UK (Mar 3–5, 2015)

[110] Canvel, B., Hiltgen, A.P., Vaudenay, S., Vuagnoux, M.: Password interception in a SSL/TLS channel. In: Boneh, D. (ed.) Advances in Cryptology – CRYPTO 2003. Lecture Notes in Computer Science, vol. 2729, pp. 583–599. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2003)

[111] Carlet, C.: Boolean Functions for Cryptography and Error-Correcting Codes, p. 257–397. Encyclopedia of Mathematics and its Applications, Cambridge University Press (2010)

[112] Chakraborti, A., Mridul, N.: TriviA-ck-v2. `https://competitions.cr.yp.to/round2/triviackv2.pdf` (2015)

[113] Cid, C., Huang, T., Peyrin, T., Sasaki, Y., Song, L.: A security analysis of Deoxys and its internal tweakable block ciphers. IACR Transactions on Symmetric Cryptology 2017(3), 73–107 (2017)

[114] Cid, C., Huang, T., Peyrin, T., Sasaki, Y., Song, L.: Boomerang connectivity table: A new cryptanalysis tool. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2018, Part II. Lecture Notes in Computer Science, vol. 10821, pp. 683–714. Springer, Heidelberg, Germany, Tel Aviv, Israel (Apr 29 – May 3, 2018)

[115] Cogliani, S., Maimut, D., Naccache, D., Portella, R., Reyhanitabar, R., Vaudenay, S., Vizar, D.: Offset Merkle-Damgard (OMD) version 2.0. `https://competitions.cr.yp.to/round2/omdv20.pdf` (2015)

[116] Crowley, P.: Truncated differential cryptanalysis of five rounds of Salsa20. Cryptology ePrint Archive, Report 2005/375 (2005), `http://eprint.iacr.org/2005/375`

[117] Daemen, J.: Cipher and hash function design, strategies based on linear and differential cryptanalysis, PhD Thesis. K.U.Leuven (1995), `http://jda.noekeon.org/`

[118] Daemen, J., Knudsen, L.R., Rijmen, V.: The block cipher Square. In: Biham, E. (ed.) Fast Software Encryption – FSE'97. Lecture Notes in Computer Science, vol. 1267, pp. 149–165. Springer, Heidelberg, Germany, Haifa, Israel (Jan 20–22, 1997)

[119] Daemen, J., Lamberger, M., Pramstaller, N., Rijmen, V., Vercauteren, F.: Computational aspects of the expected differential probability of 4-round AES and AES-like ciphers. Computing 85(1), 85–104 (Jun 2009), `https://doi.org/10.1007/s00607-009-0034-y`

[120] Daemen, J., Rijmen, V.: The wide trail design strategy. In: Honary, B. (ed.) 8th IMA International Conference on Cryptography and Coding. Lecture Notes in Computer Science, vol. 2260, pp. 222–238. Springer, Heidelberg, Germany, Cirencester, UK (Dec 17–19, 2001)

[121] Daemen, J., Rijmen, V.: The Design of Rijndael. Springer-Verlag New York, Inc., Secaucus, NJ, USA (2002)

[122] Daemen, J., Rijmen, V.: Plateau characteristics. IET Information Security 1(1), 11–17 (2007)

[123] Damgård, I.: A design principle for hash functions. In: Brassard, G. (ed.) Advances in Cryptology – CRYPTO'89. Lecture Notes in Computer Science, vol. 435, pp. 416–427. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 20–24, 1990)

[124] Datta, N., Nandi, M.: ELmD v2.0. `https://competitions.cr.yp.to/round2/elmdv20.pdf` (2015)

[125] Davies, D.W., Murphy, S.: Pairs and triplets of DES S-boxes. Journal of Cryptology 8(1), 1–25 (Dec 1995)

[126] Davies, D.W., Price, W.L.: Digital signatures, an update. In Proceedings 5th international conference on computer communication, pages 845-849 (Oktober 1984)

[127] Davis, M., Logemann, G., Loveland, D.: A machine program for theorem-proving. Commun. ACM 5(7), 394–397 (Jul 1962), `http://doi.acm.org/10.1145/368273.368557`

[128] De Cannière, C.: Analysis and Design of Symmetric Encryption Algorithms. Ph.D. thesis, Katholieke Universiteit Leuven (May 2007)

[129] De Cannière, C., Preneel, B.: Trivium specifications (2004), `http://www.ecrypt.eu.org/stream/p3ciphers/trivium/trivium_p3.pdf`

[130] De Cannière, C., Rechberger, C.: Finding SHA-1 characteristics: General results and applications. In: Lai, X., Chen, K. (eds.) Advances in Cryptology – ASIACRYPT 2006. Lecture Notes in Computer Science, vol. 4284, pp. 1–20. Springer, Heidelberg, Germany, Shanghai, China (Dec 3–7, 2006)

[131] De Koning Gans, G., Hoepman, J.H., Garcia, F.D.: A practical attack on the MIFARE classic. In: Grimaud, G., Standaert, F.X. (eds.) Smart Card Research and Advanced Applications. pp. 267–282. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)

[132] Dhanuskodi, S.N., Holcomb, D.: Energy optimization of unrolled block ciphers using combinational checkpointing. In: Hancke, G.P., Markantonakis, K. (eds.) Radio Frequency Identification and IoT Security. pp. 47–61. Springer International Publishing, Cham (2017)

[133] Diffie, W., Hellman, M.: Privacy and authentication: An introduction to cryptography. Proceedings of the IEEE 67, 397–427 (1979)

[134] Dinu, D., Perrin, L., Udovenko, A., Velichkov, V., Großschädl, J., Biryukov, A.: private communication

[135] Dinu, D., Perrin, L., Udovenko, A., Velichkov, V., Großschädl, J., Biryukov, A.: Design strategies for ARX with provable bounds: Sparx and LAX. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology – ASIACRYPT 2016, Part I. Lecture Notes in Computer Science, vol. 10031, pp. 484–513. Springer, Heidelberg, Germany, Hanoi, Vietnam (Dec 4–8, 2016)

[136] Dobbertin, H.: Cryptanalysis of MD4. In: Gollmann, D. (ed.) Fast Software Encryption – FSE'96. Lecture Notes in Computer Science, vol. 1039, pp. 53–69. Springer, Heidelberg, Germany, Cambridge, UK (Feb 21–23, 1996)

[137] Dobraunig, C., Eichlseder, M., Kales, D., Mendel, F.: Practical key-recovery attack on MANTIS5. IACR Transactions on Symmetric Cryptology 2016(2), 248–260 (2016), `http://tosc.iacr.org/index.php/ToSC/article/view/573`

[138] Dobraunig, C., Eichlseder, M., Mendel, F.: Heuristic tool for linear cryptanalysis with applications to CAESAR candidates. In: Iwata, T., Cheon, J.H.

(eds.) Advances in Cryptology – ASIACRYPT 2015, Part II. Lecture Notes in Computer Science, vol. 9453, pp. 490–509. Springer, Heidelberg, Germany, Auckland, New Zealand (Nov 30 – Dec 3, 2015)

[139] Dobraunig, C., Eichlseder, M., Mendel, F.: Square attack on 7-round kiasu-BC. In: Manulis, M., Sadeghi, A.R., Schneider, S. (eds.) ACNS 16: 14th International Conference on Applied Cryptography and Network Security. Lecture Notes in Computer Science, vol. 9696, pp. 500–517. Springer, Heidelberg, Germany, Guildford, UK (Jun 19–22, 2016)

[140] Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: ASCON v1.1. https://competitions.cr.yp.to/round2/asconv11.pdf (2015)

[141] Dobraunig, C., List, E.: Impossible-differential and boomerang cryptanalysis of round-reduced kiasu-BC. In: Handschuh, H. (ed.) Topics in Cryptology – CT-RSA 2017. Lecture Notes in Computer Science, vol. 10159, pp. 207–222. Springer, Heidelberg, Germany, San Francisco, CA, USA (Feb 14–17, 2017)

[142] Dolmatov, V.: GOST 28147-89: Encryption, Decryption, and Message Authentication Code (MAC) Algorithms. RFC 5830 (Mar 2010), https://rfc-editor.org/rfc/rfc5830.txt

[143] Driessen, B., Hund, R., Willems, C., Paar, C., Holz, T.: Don't trust satellite phones: A security analysis of two satphone standards. In: 2012 IEEE Symposium on Security and Privacy. pp. 128–142 (May 2012)

[144] Dunkelman, O., Keller, N., Shamir, A.: A practical-time related-key attack on the KASUMI cryptosystem used in GSM and 3G telephony. In: Rabin, T. (ed.) Advances in Cryptology – CRYPTO 2010. Lecture Notes in Computer Science, vol. 6223, pp. 393–410. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 15–19, 2010)

[145] Dworkin, M.: Recommendation for Block Cipher Modes of Operation: Methods and Techniques. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf (2001)

[146] Dworkin, M.J.: Sp 800-38c. recommendation for block cipher modes of operation: The CCM mode for authentication and confidentiality. Tech. rep., Gaithersburg, MD, United States (2004)

[147] Dworkin, M.J.: Sp 800-38d. recommendation for block cipher modes of operation: Galois/counter mode (GCM) and GMAC. Tech. rep., Gaithersburg, MD, United States (2007)

[148] Eastlake, D., Jones, P.: US secure hash algorithm 1 (SHA1) (2001)

[149] Eastlake, D.E.: Transport Layer Security (TLS) Extensions: Extension Definitions. RFC 6066 (Jan 2011), https://rfc-editor.org/rfc/rfc6066.txt

[150] Ehrsam, W.F., Meyer, C.H., Smith, J.L., Tuchman, W.L.: Message verification and transmission error detection by block chaining. US Patent 4074066 https://patents.google.com/patent/US4074066A/en (1976)

[151] Eichlseder, M., Kales, D.: Clustering related-tweak characteristics: Application to MANTIS-6. IACR Transactions on Symmetric Cryptology 2018(2), 111–132 (2018)

[152] Eskandari, Z., Kidmose, A.B., Kölbl, S., Tiessen, T.: Finding integral distinguishers with ease. In: Cid, C., Jacobson Jr., M.J. (eds.) Selected Areas in Cryptography – SAC 2018. pp. 115–138. Springer International Publishing, Cham (2019)

[153] Even, S., Mansour, Y.: A construction of a cipher from a single pseudorandom permutation. In: Imai, H., Rivest, R.L., Matsumoto, T. (eds.) Advances in Cryptology – ASIACRYPT'91. Lecture Notes in Computer Science, vol. 739, pp. 210–224. Springer, Heidelberg, Germany, Fujiyoshida, Japan (Nov 11–14, 1993)

[154] Feistel, H.: Cryptography and Computer Privacy. Scientific American (1973), https://books.google.co.uk/books?id=Y0BenQEACAAJ

[155] Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D., Whiting, D.: Improved cryptanalysis of Rijndael. In: Schneier, B. (ed.) Fast Software Encryption – FSE 2000. Lecture Notes in Computer Science, vol. 1978, pp. 213–230. Springer, Heidelberg, Germany, New York, NY, USA (Apr 10–12, 2001)

[156] Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: The Skein hash function family (2010), http://www.skein-hash.info/sites/default/files/skein.pdf

[157] Freier, A.O., Karlton, P., Kocher, P.C.: The Secure Sockets Layer (SSL) Protocol Version 3.0. RFC 6101 (Aug 2011), https://rfc-editor.org/rfc/rfc6101.txt

[158] Frenkel, S.: A cute toy just brought a hacker into your home. https://www.nytimes.com/2017/12/21/technology/connected-toys-hacking.html (2017)

[159] Fu, K., Wang, M., Guo, Y., Sun, S., Hu, L.: MILP-based automatic search algorithms for differential and linear trails for speck. In: Peyrin, T. (ed.) Fast Software Encryption – FSE 2016. Lecture Notes in Computer Science, vol. 9783, pp. 268–288. Springer, Heidelberg, Germany, Bochum, Germany (Mar 20–23, 2016)

[160] Garcia, F.D., Koning Gans, G., Verdult, R.: Wirelessly lockpicking a smart card reader. Int. J. Inf. Secur. 13(5), 403–420 (Oct 2014), http://dx.doi.org/10.1007/s10207-014-0234-0

[161] Garcia, F.D., van Rossum, P., Verdult, R., Wichers Schreur, R.: Dismantling securememory, cryptomemory and cryptorf. In: Proceedings of the 17th ACM Conference on Computer and Communications Security. pp. 250–259. CCS '10, ACM, New York, NY, USA (2010), http://doi.acm.org/10.1145/1866307.1866336

[162] Gérault, D., Lafourcade, P.: Related-key cryptanalysis of Midori. In: Dunkelman, O., Sanadhya, S.K. (eds.) Progress in Cryptology - INDOCRYPT 2016: 17th International Conference in Cryptology in India. Lecture Notes in Computer Science, vol. 10095, pp. 287–304. Springer, Heidelberg, Germany, Kolkata, India (Dec 11–14, 2016)

[163] Gligoroski, D., Mihajloska, H., Samardjiska, S., Jacobsen, H., El-Hadedy, M., Jensen, R.E., Otte, D.: π-cipher v2.0. https://competitions.cr.yp.to/round2/picipherv20.pdf (2015)

[164] Grosso, V., Leurent, G., Standaert, F.X., Varici, K., Journault, A., Durvaux, F., Gaspar, L., Kerckhof, S.: SCREAM: Side-Channel Resistant Authenticated Encryption with Masking. https://competitions.cr.yp.to/round2/screamv3.pdf (2015)

[165] Gueron, S.: Intel® advanced encryption standard (aes) new instructions set. Tech. rep., Intel Corporation (2012)

[166] Guo, J., Jean, J., Nikolic, I., Qiao, K., Sasaki, Y., Sim, S.M.: Invariant subspace attack against Midori64 and the resistance criteria for S-box designs. IACR Transactions on Symmetric Cryptology 2016(1), 33–56 (2016), http://tosc.iacr.org/index.php/ToSC/article/view/534

[167] Guo, J., Peyrin, T., Poschmann, A.: The PHOTON family of lightweight hash functions. In: Rogaway, P. (ed.) Advances in Cryptology – CRYPTO 2011. Lecture Notes in Computer Science, vol. 6841, pp. 222–239. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 14–18, 2011)

[168] Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.J.B.: The LED block cipher. In: Preneel, B., Takagi, T. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2011. Lecture Notes in Computer Science, vol. 6917, pp. 326–341. Springer, Heidelberg, Germany, Nara, Japan (Sep 28 – Oct 1, 2011)

[169] Harris, S.: The Enchilada authenticated ciphers, v1. https://competitions.cr.yp.to/round1/enchiladav1.pdf (2014)

[170] Hell, M., Johansson, T., Meier, W.: Grain - a stream cipher for constrained environments (2004), http://www.ecrypt.eu.org/stream/p3ciphers/grain/Grain_p3.pdf

[171] Hellman, M.: A cryptanalytic time-memory trade-off. IEEE Transactions on Information Theory 26(4), 401–406 (July 1980)

[172] Heys, H.M.: A tutorial on linear and differential cryptanalysis. Cryptologia 26(3), 189–221 (Jul 2002), http://dx.doi.org/10.1080/0161-110291890885

[173] Hoang, V.T., Krovetz, T., Rogaway, P.: AEZ v4.1: authenticated encryption by enciphering. https://competitions.cr.yp.to/round2/aezv41.pdf (2015)

[174] Hong, D., Sung, J., Hong, S., Lim, J., Lee, S., Koo, B.S., Lee, C., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J., Chee, S.: HIGHT: A new block cipher suitable

for low-resource device. In: Goubin, L., Matsui, M. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2006. Lecture Notes in Computer Science, vol. 4249, pp. 46–59. Springer, Heidelberg, Germany, Yokohama, Japan (Oct 10–13, 2006)

[175] Hosseini, H., Khazaei, S.: CBA Mode (v1). `https://competitions.cr.yp.to/round1/cbav1.pdf` (2014)

[176] Hsu, C.L., Lin, J.C.C.: An empirical examination of consumer adoption of internet of things services: Network externalities and concern for information privacy perspectives. Computers in Human Behavior 62, 516 – 527 (2016), `http://www.sciencedirect.com/science/article/pii/S0747563216302990`

[177] Huang, T., Wu, H.: The authenticated cipher MORUS (v1). `https://competitions.cr.yp.to/round2/morusv11.pdf` (2015)

[178] Igoe, K., Solinas, J.: AES Galois Counter Mode for the Secure Shell Transport Layer Protocol. RFC 5647 (Aug 2009), `https://rfc-editor.org/rfc/rfc5647.txt`

[179] Indesteege, S., Keller, N., Dunkelman, O., Biham, E., Preneel, B.: A practical attack on KeeLoq. In: Smart, N.P. (ed.) Advances in Cryptology – EUROCRYPT 2008. Lecture Notes in Computer Science, vol. 4965, pp. 1–18. Springer, Heidelberg, Germany, Istanbul, Turkey (Apr 13–17, 2008)

[180] Isobe, T.: A single-key attack on the full GOST block cipher. In: Joux, A. (ed.) Fast Software Encryption – FSE 2011. Lecture Notes in Computer Science, vol. 6733, pp. 290–305. Springer, Heidelberg, Germany, Lyngby, Denmark (Feb 13–16, 2011)

[181] Iwata, T., Kurosawa, K.: OMAC: One-key CBC MAC. In: Johansson, T. (ed.) Fast Software Encryption – FSE 2003. Lecture Notes in Computer Science, vol. 2887, pp. 129–153. Springer, Heidelberg, Germany, Lund, Sweden (Feb 24–26, 2003)

[182] Iwata, T., Kurosawa, K.: OMAC: One-Key CBC MAC – Addendum. `https://csrc.nist.gov/csrc/media/projects/block-cipher-techniques/documents/bcm/proposed-modes/omac/omac-ad.pdf` (March 2003)

[183] Iwata, T., Minematsu, K., Guo, J., Morioka, S.: CLOC: Compact Low-Overhead CFB. `https://competitions.cr.yp.to/round2/clocv2.pdf` (2015)

[184] Iwata, T., Minematsu, K., Guo, J., Morioka, S., Kobayashi, E.: SILC: SImple Lightweight CFB. `https://competitions.cr.yp.to/round2/silcv2.pdf` (2015)

[185] Jakobsen, T., Knudsen, L.R.: The interpolation attack on block ciphers. In: Biham, E. (ed.) Fast Software Encryption – FSE'97. Lecture Notes in Computer Science, vol. 1267, pp. 28–40. Springer, Heidelberg, Germany, Haifa, Israel (Jan 20–22, 1997)

[186] Jean, J., Nikolic, I., Peyrin, T.: Joltik v1.3. https://competitions.cr.yp.to/round2/joltikv13.pdf (2015)

[187] Jean, J., Nikolic, I., Peyrin, T.: Kiasu v1. https://competitions.cr.yp.to/round1/kiasuv1.pdf (2015)

[188] Jean, J., Nikolic, I., Peyrin, T.: Tweaks and keys for block ciphers: The TWEAKEY framework. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology – ASIACRYPT 2014, Part II. Lecture Notes in Computer Science, vol. 8874, pp. 274–288. Springer, Heidelberg, Germany, Kaoshiung, Taiwan, R.O.C. (Dec 7–11, 2014)

[189] Jean, J., Nikolić, I., Peyrin, T., Seurin, Y.: Deoxys v1.41. https://competitions.cr.yp.to/round3/deoxysv141.pdf (October 2016)

[190] Joan Daemen, Michaël Peeters, Gilles Van Assche, Vincent Rijmen: Nessie Proposal: NOEKEON (2000), http://gro.noekeon.org/Noekeon-spec.pdf

[191] Jonsson, J.: On the security of CTR + CBC-MAC. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002: 9th Annual International Workshop on Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 2595, pp. 76–93. Springer, Heidelberg, Germany, St. John's, Newfoundland, Canada (Aug 15–16, 2003)

[192] Joux, A.: Authentication failures in nist version of GCM. https://csrc.nist.gov/csrc/media/projects/block-cipher-techniques/documents/bcm/joux_comments.pdf (2006)

[193] Kaliski Jr., B.S., Robshaw, M.J.B.: Linear cryptanalysis using multiple approximations. In: Desmedt, Y. (ed.) Advances in Cryptology – CRYPTO'94. Lecture Notes in Computer Science, vol. 839, pp. 26–39. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 21–25, 1994)

[194] Kanda, M., Matsumoto, T.: Security of Camellia against truncated differential cryptanalysis. In: Matsui, M. (ed.) Fast Software Encryption – FSE 2001. Lecture Notes in Computer Science, vol. 2355, pp. 286–299. Springer, Heidelberg, Germany, Yokohama, Japan (Apr 2–4, 2002)

[195] Kavun, E.B., Lauridsen, M.M., Leander, G., Rechberger, C., Schwabe, P., Yalçın, T.: Prøst v1. https://competitions.cr.yp.to/round1/proestv1.pdf (2014)

[196] Keliher, L., Sui, J.: Exact maximum expected differential and linear probability for two-round advanced encryption standard. IET Information Security 1(2), 53–57 (2007), https://doi.org/10.1049/iet-ifs:20060161

[197] Kelsey, J., Kohno, T., Schneier, B.: Amplified boomerang attacks against reduced-round MARS and Serpent. In: Schneier, B. (ed.) Fast Software Encryption – FSE 2000. Lecture Notes in Computer Science, vol. 1978, pp. 75–93. Springer, Heidelberg, Germany, New York, NY, USA (Apr 10–12, 2001)

[198] Kerckhoffs, A.: La cryptographie militaire. Journal des sciences militaires IX, 5–38 (Jan 1883), http://petitcolas.net/kerckhoffs/crypto_militaire_1.pdf (in french)

[199] Knežević, M., Nikov, V., Rombouts, P.: Low-latency encryption - is "lightweight = light + wait"? In: Prouff, E., Schaumont, P. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2012. Lecture Notes in Computer Science, vol. 7428, pp. 426–446. Springer, Heidelberg, Germany, Leuven, Belgium (Sep 9–12, 2012)

[200] Knudsen, L.: DEAL - A 128-bit Block Cipher. In: NIST AES Proposal (1998)

[201] Knudsen, L.R.: Truncated and higher order differentials. In: Preneel, B. (ed.) Fast Software Encryption – FSE'94. Lecture Notes in Computer Science, vol. 1008, pp. 196–211. Springer, Heidelberg, Germany, Leuven, Belgium (Dec 14–16, 1995)

[202] Knudsen, L.R., Berson, T.A.: Truncated differentials of SAFER. In: Gollmann, D. (ed.) Fast Software Encryption – FSE'96. Lecture Notes in Computer Science, vol. 1039, pp. 15–26. Springer, Heidelberg, Germany, Cambridge, UK (Feb 21–23, 1996)

[203] Knudsen, L.R., Rijmen, V.: Known-key distinguishers for some block ciphers. In: Kurosawa, K. (ed.) Advances in Cryptology – ASIACRYPT 2007. Lecture Notes in Computer Science, vol. 4833, pp. 315–324. Springer, Heidelberg, Germany, Kuching, Malaysia (Dec 2–6, 2007)

[204] Knudsen, L.R., Robshaw, M.J.B., Wagner, D.: Truncated differentials and Skipjack. In: Wiener, M.J. (ed.) Advances in Cryptology – CRYPTO'99. Lecture Notes in Computer Science, vol. 1666, pp. 165–180. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 15–19, 1999)

[205] Knudsen, L.R., Wagner, D.: Integral cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) Fast Software Encryption – FSE 2002. Lecture Notes in Computer Science, vol. 2365, pp. 112–127. Springer, Heidelberg, Germany, Leuven, Belgium (Feb 4–6, 2002)

[206] Kölbl, S., Leander, G., Tiessen, T.: Observations on the SIMON block cipher family. In: Gennaro, R., Robshaw, M.J.B. (eds.) Advances in Cryptology – CRYPTO 2015, Part I. Lecture Notes in Computer Science, vol. 9215, pp. 161–185. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 2015)

[207] Kölbl, S., Roy, A.: A brief comparison of Simon and Simeck. In: Bogdanov, A. (ed.) Lightweight Cryptography for Security and Privacy. pp. 69–88. Springer International Publishing, Cham (2017)

[208] Kranz, T., Leander, G., Wiemer, F.: Linear cryptanalysis: Key schedules and tweakable block ciphers. IACR Transactions on Symmetric Cryptology 2017(1), 474–505 (2017)

[209] Kranz, T., Wiemer, F., Leander, G.: Linear cryptanalysis: Key schedules and tweakable block ciphers. Cryptology ePrint Archive, Report 2017/154 (2017), `http://eprint.iacr.org/2017/154`

[210] Krovetz, T.: HS1-SIV (v2). `https://competitions.cr.yp.to/round2/hs1sivv2.pdf` (2015)

[211] Krovetz, T., Rogaway, P.: The software performance of authenticated-encryption modes. In: Joux, A. (ed.) Fast Software Encryption – FSE 2011. Lecture Notes in Computer Science, vol. 6733, pp. 306–327. Springer, Heidelberg, Germany, Lyngby, Denmark (Feb 13–16, 2011)

[212] Krovetz, T., Rogaway, P.: OCB (v1). `https://competitions.cr.yp.to/round1/ocbv1.pdf` (2014)

[213] Lai, X.: Higher Order Derivatives and Differential Cryptanalysis, pp. 227–233. Springer US, Boston, MA (1994), `https://doi.org/10.1007/978-1-4615-2694-0_23`

[214] Lai, X., Massey, J.L.: A proposal for a new block encryption standard. In: Damgård, I. (ed.) Advances in Cryptology – EUROCRYPT'90. Lecture Notes in Computer Science, vol. 473, pp. 389–404. Springer, Heidelberg, Germany, Aarhus, Denmark (May 21–24, 1991)

[215] Lai, X., Massey, J.L., Murphy, S.: Markov ciphers and differential cryptanalysis. In: Davies, D.W. (ed.) Advances in Cryptology – EUROCRYPT'91. Lecture Notes in Computer Science, vol. 547, pp. 17–38. Springer, Heidelberg, Germany, Brighton, UK (Apr 8–11, 1991)

[216] Landecker, W., Shrimpton, T., Terashima, R.S.: Tweakable blockciphers with beyond birthday-bound security. In: Safavi-Naini, R., Canetti, R. (eds.) Advances in Cryptology – CRYPTO 2012. Lecture Notes in Computer Science, vol. 7417, pp. 14–30. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2012)

[217] Langford, S.K., Hellman, M.E.: Differential-linear cryptanalysis. In: Desmedt, Y. (ed.) Advances in Cryptology – CRYPTO'94. Lecture Notes in Computer Science, vol. 839, pp. 17–25. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 21–25, 1994)

[218] Larson, S.: Fda confirms that st. jude's cardiac devices can be hacked. `https://money.cnn.com/2017/01/09/technology/fda-st-jude-cardiac-hack/` (2017)

[219] Leander, G., Poschmann, A.: On the Classification of 4 Bit S-Boxes, pp. 159–176. Springer Berlin Heidelberg, Berlin, Heidelberg (2007), `https://doi.org/10.1007/978-3-540-73074-3_13`

[220] Leander, G., Abdelraheem, M.A., AlKhzaimi, H., Zenner, E.: A cryptanalysis of PRINTcipher: The invariant subspace attack. In: Rogaway, P. (ed.) Advances in Cryptology – CRYPTO 2011. Lecture Notes in Computer Science,

vol. 6841, pp. 206–221. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 14–18, 2011)

[221] Leander, G., Tezcan, C., Wiemer, F.: Searching for subspace trails and truncated differentials. IACR Transactions on Symmetric Cryptology 2018(1), 74–100 (2018)

[222] Leurent, G.: Analysis of differential attacks in ARX constructions. In: Wang, X., Sako, K. (eds.) Advances in Cryptology – ASIACRYPT 2012. Lecture Notes in Computer Science, vol. 7658, pp. 226–243. Springer, Heidelberg, Germany, Beijing, China (Dec 2–6, 2012)

[223] Leurent, G.: Construction of differential characteristics in ARX designs application to Skein. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology – CRYPTO 2013, Part I. Lecture Notes in Computer Science, vol. 8042, pp. 241–258. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2013)

[224] Leurent, G.: Improved differential-linear cryptanalysis of 7-round Chaskey with partitioning. In: Fischlin, M., Coron, J.S. (eds.) Advances in Cryptology – EUROCRYPT 2016, Part I. Lecture Notes in Computer Science, vol. 9665, pp. 344–371. Springer, Heidelberg, Germany, Vienna, Austria (May 8–12, 2016)

[225] Li, R., Jin, C.: Meet-in-the-middle attacks on reduced-round QARMA-64/128. Comput. J. 61(8), 1158–1165 (2018)

[226] Lipmaa, H., Moriai, S.: Efficient algorithms for computing differential properties of addition. In: Matsui, M. (ed.) Fast Software Encryption – FSE 2001. Lecture Notes in Computer Science, vol. 2355, pp. 336–350. Springer, Heidelberg, Germany, Yokohama, Japan (Apr 2–4, 2002)

[227] Liskov, M., Rivest, R.L., Wagner, D.: Tweakable block ciphers. In: Yung, M. (ed.) Advances in Cryptology – CRYPTO 2002. Lecture Notes in Computer Science, vol. 2442, pp. 31–46. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2002)

[228] Liskov, M., Rivest, R.L., Wagner, D.: Tweakable block ciphers. Journal of Cryptology 24(3), 588–613 (Jul 2011)

[229] Liu, G.Q., Jin, C.H.: Differential cryptanalysis of present-like cipher. Designs, Codes and Cryptography 76(3), 385–408 (Sep 2015), `https://doi.org/10.1007/s10623-014-9965-1`

[230] Liu, G., Ghosh, M., Song, L.: Security analysis of SKINNY under related-tweakey settings (long paper). IACR Transactions on Symmetric Cryptology 2017(3), 37–72 (2017)

[231] Liu, Y., Rijmen, V., Leander, G.: Nonlinear diffusion layers. Designs, Codes and Cryptography (Jan 2018), `https://doi.org/10.1007/s10623-018-0458-5`

[232] Liu, Z., Li, Y., Wang, M.: Optimal differential trails in SIMON-like ciphers. IACR Transactions on Symmetric Cryptology 2017(1), 358–379 (2017)

[233] Lonvick, C.M., Ylonen, T.: The Secure Shell (SSH) Transport Layer Protocol. RFC 4253 (Jan 2006), `https://rfc-editor.org/rfc/rfc4253.txt`

[234] Luby, M., Rackoff, C.: How to construct pseudorandom permutations from pseudorandom functions. SIAM Journal on Computing 17(2) (1988)

[235] Lucks, S., Schuler, A., Tews, E., Weinmann, R.P., Wenzel, M.: Attacks on the DECT authentication mechanisms. In: Fischlin, M. (ed.) Topics in Cryptology – CT-RSA 2009. Lecture Notes in Computer Science, vol. 5473, pp. 48–65. Springer, Heidelberg, Germany, San Francisco, CA, USA (Apr 20–24, 2009)

[236] Mate Soos: CryptoMiniSat SAT solver (2009), `https://github.com/msoos/cryptominisat/`

[237] Matsuda, S., Moriai, S.: Lightweight cryptography for the cloud: Exploit the power of bitslice implementation. In: Prouff, E., Schaumont, P. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2012. Lecture Notes in Computer Science, vol. 7428, pp. 408–425. Springer, Heidelberg, Germany, Leuven, Belgium (Sep 9–12, 2012)

[238] Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) Advances in Cryptology – EUROCRYPT'93. Lecture Notes in Computer Science, vol. 765, pp. 386–397. Springer, Heidelberg, Germany, Lofthus, Norway (May 23–27, 1994)

[239] Matsui, M., Murakami, Y.: Aes smaller than s-box. In: Eisenbarth, T., Öztürk, E. (eds.) Lightweight Cryptography for Security and Privacy. pp. 51–66. Springer International Publishing, Cham (2015)

[240] Matsui, M., Yamagishi, A.: A new method for known plaintext attack of FEAL cipher. In: Rueppel, R.A. (ed.) Advances in Cryptology – EUROCRYPT'92. Lecture Notes in Computer Science, vol. 658, pp. 81–91. Springer, Heidelberg, Germany, Balatonfüred, Hungary (May 24–28, 1993)

[241] Matyas, S.M., Meyer, C.H., Oseas, J.: Generating strong one-way functions with cryptographic algorithm. In IBM Technical Disclosure Bulletin vol. 27(10A), 5658–5659 (1985)

[242] Maxwell, P.: Wheesht: an aead stream cipher. `https://competitions.cr.yp.to/round1/wheeshtv03.pdf` (2014)

[243] McGrew, D.A., Viega, J.: The security and performance of the Galois/counter mode (GCM) of operation. In: Canteaut, A., Viswanathan, K. (eds.) Progress in Cryptology - INDOCRYPT 2004: 5th International Conference in Cryptology in India. Lecture Notes in Computer Science, vol. 3348, pp. 343–355. Springer, Heidelberg, Germany, Chennai, India (Dec 20–22, 2004)

[244] Mendel, F., Nad, T., Schläffer, M.: Finding SHA-2 characteristics: Searching through a minefield of contradictions. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology – ASIACRYPT 2011. Lecture Notes in Computer Science, vol. 7073, pp. 288–307. Springer, Heidelberg, Germany, Seoul, South Korea (Dec 4–8, 2011)

[245] Mennink, B.: Optimally secure tweakable blockciphers. In: Leander, G. (ed.) Fast Software Encryption – FSE 2015. Lecture Notes in Computer Science, vol. 9054, pp. 428–448. Springer, Heidelberg, Germany, Istanbul, Turkey (Mar 8–11, 2015)

[246] Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) Advances in Cryptology – CRYPTO'89. Lecture Notes in Computer Science, vol. 435, pp. 218–238. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 20–24, 1990)

[247] Miller, C., Valasek, C.: Remote exploitation of an unaltered passenger vehicle. `https://www.blackhat.com/us-15/briefings.html#remote-exploitation-of-an-unaltered-passenger-vehicle` (2015)

[248] Minematsu, K.: Beyond-birthday-bound security based on tweakable block cipher. In: Dunkelman, O. (ed.) Fast Software Encryption – FSE 2009. Lecture Notes in Computer Science, vol. 5665, pp. 308–326. Springer, Heidelberg, Germany, Leuven, Belgium (Feb 22–25, 2009)

[249] Minematsu, K.: AES-OTR v3. `https://competitions.cr.yp.to/round2/aesotrv3.pdf` (2016)

[250] Miyaguchi, S.: The FEAL cipher family (impromptu talk). In: Menezes, A.J., Vanstone, S.A. (eds.) Advances in Cryptology – CRYPTO'90. Lecture Notes in Computer Science, vol. 537, pp. 627–638. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 11–15, 1991)

[251] Miyaguchi, S., Iwata, M., Ohta, K.: New 128-bit hash function. In Proceedings of 4th International Joint Workshop on Computer Communications pp. 279–288 (July 1989)

[252] Möller, B., Duong, T., Kotowicz, K.: This POODLE Bites: Exploiting The SSL 3.0 Fallback. `https://www.openssl.org/~bodo/ssl-poodle.pdf` (September 2014)

[253] Montes, M., Penazzi, D.: AES-CPFB v1. `https://competitions.cr.yp.to/round1/aescpfbv1.pdf` (2014)

[254] Morawiecki, P., Gaj, K., Homsirikamol, E., Matusiewicz, K., Pieprzyk, J., Rogawski, M., Srebrny, M., Wojcik, M.: ICEPOLE v2 (2015)

[255] Mouha, N., Mennink, B., Herrewege, A.V., Watanabe, D., Preneel, B., Verbauwhede, I.: Chaskey: An efficient MAC algorithm for 32-bit microcontrollers. In: Joux, A., Youssef, A.M. (eds.) SAC 2014: 21st Annual International

Workshop on Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 8781, pp. 306–323. Springer, Heidelberg, Germany, Montreal, QC, Canada (Aug 14–15, 2014)

[256] Mouha, N., Preneel, B.: Towards finding optimal differential characteristics for ARX: Application to Salsa20. Cryptology ePrint Archive, Report 2013/328 (2013), http://eprint.iacr.org/2013/328

[257] Mouha, N., Wang, Q., Gu, D., Preneel, B.: Differential and linear cryptanalysis using mixed-integer linear programming. In: Wu, C.K., Yung, M., Lin, D. (eds.) Information Security and Cryptology. pp. 57–76. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)

[258] Murphy, S.: The return of the cryptographic boomerang. IEEE Trans. Information Theory 57(4), 2517–2521 (2011)

[259] National Institute of Standards and Technology: Fips pub 197, advanced encryption standard (aes) (2001), https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.197.pdf

[260] National Institute of Standards and Technology (NIST): Secure Hash Standard - SHS: Federal Information Processing Standards Publication 180-4. CreateSpace Independent Publishing Platform, USA (2012)

[261] National Institute of Standards and Technology (NIST): SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf (August 2015)

[262] National Institute of Standards and Technology (NIST): The SHA-3 Cryptographic Hash Algorithm Competition. https://csrc.nist.gov/Projects/Hash-Functions/SHA-3-Project (November 2007 - October 2012)

[263] National Security Agency: Suite b cryptography. http://www.nsa.gov/ia/programs/suiteb_cryptography/ (2005)

[264] Niemetz, A., Preiner, M., Biere, A.: Boolector 2.0 system description. Journal on Satisfiability, Boolean Modeling and Computation 9, 53–58 (2014 (published 2015))

[265] Nikolic, I.: Tiaoxin-346. https://competitions.cr.yp.to/round2/tiaoxinv2.pdf (2015)

[266] Nohl, K., Evans, D., Starbug, S., Plötz, H.: Reverse-engineering a cryptographic rfid tag. In: Proceedings of the 17th Conference on Security Symposium. pp. 185–193. SS'08, USENIX Association, Berkeley, CA, USA (2008), http://dl.acm.org/citation.cfm?id=1496711.1496724

[267] Nohl, K., Tews, E., Weinmann, R.P.: Cryptanalysis of the DECT standard cipher. In: Hong, S., Iwata, T. (eds.) Fast Software Encryption – FSE 2010. Lecture Notes in Computer Science, vol. 6147, pp. 1–18. Springer, Heidelberg, Germany, Seoul, Korea (Feb 7–10, 2010)

[268] Nordum, A.: Popular internet of things forecast of 50 billion devices by 2020 is outdated. https://spectrum.ieee.org/tech-talk/telecom/internet/popular-internet-of-things-forecast-of-50-billion-devices-by-2020-is-outdated (2016)

[269] Nyberg, K.: On the construction of highly nonlinear permutations. In: Rueppel, R.A. (ed.) Advances in Cryptology – EUROCRYPT'92. Lecture Notes in Computer Science, vol. 658, pp. 92–98. Springer, Heidelberg, Germany, Balatonfüred, Hungary (May 24–28, 1993)

[270] Nyberg, K.: Differentially uniform mappings for cryptography. In: Helleseth, T. (ed.) Advances in Cryptology – EUROCRYPT'93. Lecture Notes in Computer Science, vol. 765, pp. 55–64. Springer, Heidelberg, Germany, Lofthus, Norway (May 23–27, 1994)

[271] Nyberg, K.: Linear approximation of block ciphers (rump session). In: Santis, A.D. (ed.) Advances in Cryptology – EUROCRYPT'94. Lecture Notes in Computer Science, vol. 950, pp. 439–444. Springer, Heidelberg, Germany, Perugia, Italy (May 9–12, 1995)

[272] Nyberg, K., Knudsen, L.R.: Provable security against differential cryptanalysis (rump session). In: Brickell, E.F. (ed.) Advances in Cryptology – CRYPTO'92. Lecture Notes in Computer Science, vol. 740, pp. 566–574. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 1993)

[273] Nyberg, K., Knudsen, L.R.: Provable security against a differential attack. Journal of Cryptology 8(1), 27–37 (Dec 1995)

[274] O. Sarrinen, M.J., Brumley, B.B.: STRIBOBr2: "WHIRLBOB". https://competitions.cr.yp.to/round2/stribobr2.pdf (2015)

[275] Osvik, D.A., Bos, J.W., Stefan, D., Canright, D.: Fast software AES encryption. In: Hong, S., Iwata, T. (eds.) Fast Software Encryption – FSE 2010. Lecture Notes in Computer Science, vol. 6147, pp. 75–93. Springer, Heidelberg, Germany, Seoul, Korea (Feb 7–10, 2010)

[276] Penazzi, D., Montes, M.: Silver v.1. https://competitions.cr.yp.to/round1/silverv1.pdf (2014)

[277] Peyrin, T., Seurin, Y.: Counter-in-tweak: Authenticated encryption modes for tweakable block ciphers. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology – CRYPTO 2016, Part I. Lecture Notes in Computer Science, vol. 9814, pp. 33–63. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 14–18, 2016)

[278] Preneel, B., Govaerts, R., Vandewalle, J.: Cryptographically secure hash functions: an overview. ESAT Internal Report, KU Leuven (1989)

[279] Procter, G., Cid, C.: On weak keys and forgery attacks against polynomial-based MAC schemes. In: Moriai, S. (ed.) Fast Software Encryption – FSE 2013. Lecture Notes in Computer Science, vol. 8424, pp. 287–304. Springer, Heidelberg, Germany, Singapore (Mar 11–13, 2014)

[280] Prouff, E., Rivain, M.: Masking against side-channel attacks: A formal security proof. In: Johansson, T., Nguyen, P.Q. (eds.) Advances in Cryptology – EUROCRYPT 2013. Lecture Notes in Computer Science, vol. 7881, pp. 142–159. Springer, Heidelberg, Germany, Athens, Greece (May 26–30, 2013)

[281] Recacha, F.: ++AE v1.0. `https://competitions.cr.yp.to/round1/aev10.pdf` (2014)

[282] Rescorla, E., Dierks, T.: The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Aug 2008), `https://rfc-editor.org/rfc/rfc5246.txt`

[283] Rivest, R.: The MD5 message-digest algorithm (1992)

[284] Rivest, R.L.: DESX (1984), unpublished

[285] Rivest, R.L.: The RC5 encryption algorithm. In: Preneel, B. (ed.) Fast Software Encryption – FSE'94. Lecture Notes in Computer Science, vol. 1008, pp. 86–96. Springer, Heidelberg, Germany, Leuven, Belgium (Dec 14–16, 1995)

[286] Rogaway, P., Wagner, D.: A critique of CCM. Cryptology ePrint Archive, Report 2003/070 (2003), `http://eprint.iacr.org/2003/070`

[287] Rogaway, P.: Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In: Lee, P.J. (ed.) Advances in Cryptology – ASIACRYPT 2004. Lecture Notes in Computer Science, vol. 3329, pp. 16–31. Springer, Heidelberg, Germany, Jeju Island, Korea (Dec 5–9, 2004)

[288] Rogaway, P., Bellare, M., Black, J., Krovetz, T.: OCB mode. Cryptology ePrint Archive, Report 2001/026 (2001), `http://eprint.iacr.org/2001/026`

[289] Ronen, E., Shamir, A., Weingarten, A.O., O'Flynn, C.: IoT goes nuclear: Creating a zigbee chain reaction. In: 2017 IEEE Symposium on Security and Privacy (SP). pp. 195–212 (May 2017)

[290] Rønjom, S., Bardeh, N.G., Helleseth, T.: Yoyo tricks with AES. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology – ASIACRYPT 2017, Part I. Lecture Notes in Computer Science, vol. 10624, pp. 217–243. Springer, Heidelberg, Germany, Hong Kong, China (Dec 3–7, 2017)

[291] Rothaus, O.: On "bent" functions. Journal of Combinatorial Theory, Series A 20(3), 300 – 305 (1976), `http://www.sciencedirect.com/science/article/pii/0097316576900248`

[292] S, J., Rao, M., Srinivas, J., Vishwanath, P., H, U., Rao, J.: Clock gating for power optimization in asic design cycle theory & practice. In: Proceedings of the 2008 International Symposium on Low Power Electronics & Design. pp. 307–308. ISLPED '08, ACM, New York, NY, USA (2008), `http://doi.acm.org/10.1145/1393921.1394003`

[293] Saarinen, M.J.O.: Cryptographic analysis of all 4×4-bit S-boxes. In: Miri, A., Vaudenay, S. (eds.) SAC 2011: 18th Annual International Workshop on Selected Areas in Cryptography. Lecture Notes in Computer Science, vol.

7118, pp. 118–133. Springer, Heidelberg, Germany, Toronto, Ontario, Canada (Aug 11–12, 2012)

[294] Saarinen, M.J.O.: Cycling attacks on GCM, GHASH and other polynomial MACs and hashes. In: Canteaut, A. (ed.) Fast Software Encryption – FSE 2012. Lecture Notes in Computer Science, vol. 7549, pp. 216–225. Springer, Heidelberg, Germany, Washington, DC, USA (Mar 19–21, 2012)

[295] Sadeghi, S., Mohammadi, T., Bagheri, N.: Cryptanalysis of reduced round SKINNY block cipher. IACR Transactions on Symmetric Cryptology 2018(3), 124–162 (2018)

[296] Salowey, J.A., McGrew, D.D.A., Choudhury, A.: AES galois counter mode (GCM) cipher suites for TLS. RFC 5288 (Aug 2008), `https://rfc-editor.org/rfc/rfc5288.txt`

[297] Sarkar, S., Syed, H.: Bounds on differential and linear branch number of permutations. In: Susilo, W., Yang, G. (eds.) Information Security and Privacy. pp. 207–224. Springer International Publishing, Cham (2018)

[298] Sasaki, Y.: Improved related-tweakey boomerang attacks on deoxys-BC. In: Joux, A., Nitaj, A., Rachidi, T. (eds.) AFRICACRYPT 18: 10th International Conference on Cryptology in Africa. Lecture Notes in Computer Science, vol. 10831, pp. 87–106. Springer, Heidelberg, Germany, Marrakesh, Morocco (May 7–9, 2018)

[299] Sasaki, Y., Todo, Y.: New impossible differential search tool from design and cryptanalysis aspects - revealing structural properties of several ciphers. In: Coron, J., Nielsen, J.B. (eds.) Advances in Cryptology – EUROCRYPT 2017, Part III. Lecture Notes in Computer Science, vol. 10212, pp. 185–215. Springer, Heidelberg, Germany, Paris, France (Apr 30 – May 4, 2017)

[300] Sasaki, Y., Todo, Y., Aoki, K., Naito, Y., Sugawara, T., Murakami, Y., Matsui, M., Hirose, S.: Minalpher v1.1. `https://competitions.cr.yp.to/round2/minalpherv11.pdf` (2015)

[301] Sasaki, Y., Wang, L.: Meet-in-the-middle technique for integral attacks against Feistel ciphers. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012: 19th Annual International Workshop on Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 7707, pp. 234–251. Springer, Heidelberg, Germany, Windsor, Ontario, Canada (Aug 15–16, 2013)

[302] Satoh, A., Morioka, S., Takano, K., Munetoh, S.: A compact Rijndael hardware architecture with S-box optimization. In: Boyd, C. (ed.) Advances in Cryptology – ASIACRYPT 2001. Lecture Notes in Computer Science, vol. 2248, pp. 239–254. Springer, Heidelberg, Germany, Gold Coast, Australia (Dec 9–13, 2001)

[303] Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., Ferguson, N.: Twofish: A 128-bit block cipher. In: in First Advanced Encryption Standard (AES) Conference (1998)

[304] Schroeppel, R.: The Hasty Pudding Cipher. `http://richard.schroeppel.name:8015/hpc/hpc-spec` (June 1998)

[305] Schwabe, P., Stoffelen, K.: All the AES you need on Cortex-M3 and M4. In: Avanzi, R., Heys, H.M. (eds.) SAC 2016: 23rd Annual International Workshop on Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 10532, pp. 180–194. Springer, Heidelberg, Germany, St. John's, NL, Canada (Aug 10–12, 2016)

[306] Selçuk, A.A., Biçak, A.: On probability of success in linear and differential cryptanalysis. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 02: 3rd International Conference on Security in Communication Networks. Lecture Notes in Computer Science, vol. 2576, pp. 174–185. Springer, Heidelberg, Germany, Amalfi, Italy (Sep 12–13, 2003)

[307] Seo, K., Kent, S.: Security Architecture for the Internet Protocol. RFC 4301 (Dec 2005), `https://rfc-editor.org/rfc/rfc4301.txt`

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

[309] Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., Shirai, T.: Piccolo: An ultra-lightweight blockcipher. In: Preneel, B., Takagi, T. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2011. Lecture Notes in Computer Science, vol. 6917, pp. 342–357. Springer, Heidelberg, Germany, Nara, Japan (Sep 28 – Oct 1, 2011)

[310] Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-bit blockcipher CLEFIA (extended abstract). In: Biryukov, A. (ed.) Fast Software Encryption – FSE 2007. Lecture Notes in Computer Science, vol. 4593, pp. 181–195. Springer, Heidelberg, Germany, Luxembourg, Luxembourg (Mar 26–28, 2007)

[311] Siegenthaler, T.: Correlation-immunity of nonlinear combining functions for cryptographic applications (corresp.). IEEE Transactions on Information Theory 30(5), 776–780 (September 1984)

[312] Song, L., Huang, Z., Yang, Q.: Automatic differential analysis of ARX block ciphers with application to SPECK and LEA. In: Liu, J.K., Steinfeld, R. (eds.) ACISP 16: 21st Australasian Conference on Information Security and Privacy, Part II. Lecture Notes in Computer Science, vol. 9723, pp. 379–394. Springer, Heidelberg, Germany, Melbourne, VIC, Australia (Jul 4–6, 2016)

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

[314] Stevens, M., Bursztein, E., Karpman, P., Albertini, A., Markov, Y.: The first collision for full SHA-1. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology – CRYPTO 2017, Part I. Lecture Notes in Computer Science, vol. 10401, pp. 570–596. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 20–24, 2017)

[315] Sun, B., Liu, Z., Rijmen, V., Li, R., Cheng, L., Wang, Q., AlKhzaimi, H., Li, C.: Links among impossible differential, integral and zero correlation linear cryptanalysis. In: Gennaro, R., Robshaw, M.J.B. (eds.) Advances in Cryptology – CRYPTO 2015, Part I. Lecture Notes in Computer Science, vol. 9215, pp. 95–115. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 2015)

[316] Sun, S., Gerault, D., Lafourcade, P., Yang, Q., Todo, Y., Qiao, K., Hu, L.: Analysis of AES, SKINNY, and others with constraint programming. IACR Transactions on Symmetric Cryptology 2017(1), 281–306 (2017)

[317] Sun, S., Hu, L., Wang, M., Wang, P., Qiao, K., Ma, X., Shi, D., Song, L., Fu, K.: 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 (2014), http://eprint.iacr.org/2014/747

[318] Suzaki, T., Minematsu, K., Morioka, S., Kobayashi, E.: TWINE : A lightweight block cipher for multiple platforms. In: Knudsen, L.R., Wu, H. (eds.) SAC 2012: 19th Annual International Workshop on Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 7707, pp. 339–354. Springer, Heidelberg, Germany, Windsor, Ontario, Canada (Aug 15–16, 2013)

[319] Teicher, J.: The little-known story of the first IoT device. https://www.ibm.com/blogs/industries/little-known-story-first-iot-device/ (2018)

[320] Tezcan, C., Okan, G.O., Şenol, A., Doğan, E., Yücebaş, F., Baykal, N.: Differential attacks on lightweight block ciphers Present, Pride, and Rectangle revisited. In: Bogdanov, A. (ed.) Lightweight Cryptography for Security and Privacy. pp. 18–32. Springer International Publishing, Cham (2017)

[321] Todo, Y.: Integral cryptanalysis on full MISTY1. In: Gennaro, R., Robshaw, M.J.B. (eds.) Advances in Cryptology – CRYPTO 2015, Part I. Lecture Notes in Computer Science, vol. 9215, pp. 413–432. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 2015)

[322] Todo, Y.: Structural evaluation by generalized integral property. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015, Part I. Lecture Notes in Computer Science, vol. 9056, pp. 287–314. Springer, Heidelberg, Germany, Sofia, Bulgaria (Apr 26–30, 2015)

[323] Todo, Y.: Integral cryptanalysis on full MISTY1. Journal of Cryptology 30(3), 920–959 (Jul 2017)

[324] Todo, Y., Aoki, K.: FFT key recovery for integral attack. In: Gritzalis, D., Kiayias, A., Askoxylakis, I.G. (eds.) CANS 14: 13th International Conference on Cryptology and Network Security. Lecture Notes in Computer Science, vol. 8813, pp. 64–81. Springer, Heidelberg, Germany, Heraklion, Crete, Greece (Oct 22–24, 2014)

[325] Todo, Y., Leander, G., Sasaki, Y.: Nonlinear invariant attack - practical attack on full SCREAM, iSCREAM, and Midori64. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology – ASIACRYPT 2016, Part II. Lecture Notes in Computer Science, vol. 10032, pp. 3–33. Springer, Heidelberg, Germany, Hanoi, Vietnam (Dec 4–8, 2016)

[326] Todo, Y., Morii, M.: Bit-based division property and application to Simon family. In: Peyrin, T. (ed.) Fast Software Encryption – FSE 2016. Lecture Notes in Computer Science, vol. 9783, pp. 357–377. Springer, Heidelberg, Germany, Bochum, Germany (Mar 20–23, 2016)

[327] Tolba, M., Abdelkhalek, A., Youssef, A.M.: A meet in the middle attack on reduced round kiasu-bc. IEICE Transactions 99-A(10), 1888–1890 (2016), `http://search.ieice.org/bin/summary.php?id=e99-a_10_1888`

[328] Tolba, M., Abdelkhalek, A., Youssef, A.M.: Impossible differential cryptanalysis of reduced-round SKINNY. In: Joye, M., Nitaj, A. (eds.) AFRICACRYPT 17: 9th International Conference on Cryptology in Africa. Lecture Notes in Computer Science, vol. 10239, pp. 117–134. Springer, Heidelberg, Germany, Dakar, Senegal (May 24–26, 2017)

[329] Tolba, M., Abdelkhalek, A., Youssef, A.M.: Multidimensional zero-correlation linear cryptanalysis of reduced round SPARX-128. In: Adams, C., Camenisch, J. (eds.) SAC 2017: 24th Annual International Workshop on Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 10719, pp. 423–441. Springer, Heidelberg, Germany, Ottawa, ON, Canada (Aug 16–18, 2017)

[330] Trostle, J.: AES-CMCC v1. `https://competitions.cr.yp.to/round1/aescmccv1.pdf` (2014)

[331] Ueno, R., Morioka, S., Homma, N., Aoki, T.: A high throughput/gate AES hardware architecture by compressing encryption and decryption datapaths - toward efficient CBC-mode implementation. In: Gierlichs, B., Poschmann, A.Y. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2016. Lecture Notes in Computer Science, vol. 9813, pp. 538–558. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–19, 2016)

[332] Vaudenay, S.: Security flaws induced by CBC padding - applications to SSL, IPSEC, WTLS... In: Knudsen, L.R. (ed.) Advances in Cryptology – EUROCRYPT 2002. Lecture Notes in Computer Science, vol. 2332, pp. 534–546. Springer, Heidelberg, Germany, Amsterdam, The Netherlands (Apr 28 – May 2, 2002)

[333] Verdult, R., Garcia, F.D., Balasch, J.: Gone in 360 seconds: Hijacking with hitag2. In: Proceedings of the 21st USENIX Conference on Security Symposium. pp. 37–37. Security'12, USENIX Association, Berkeley, CA, USA (2012), `http://dl.acm.org/citation.cfm?id=2362793.2362830`

[334] Verdult, R., Garcia, F.D., Ege, B.: Dismantling megamos crypto: Wirelessly lockpicking a vehicle immobilizer. In: 22nd USENIX Security Symposium

(USENIX Security 13). USENIX Association, Washington, D.C. (2013), https://www.usenix.org/conference/usenixsecurity13/dismantling-megamos-crypto-wirelessly-lockpicking-vehicle-immobilizer

[335] Viega, J., McGrew, D.D.A.: The use of galois/counter mode (GCM) in ipsec encapsulating security payload (ESP). RFC 4106 (Jun 2005), https://rfc-editor.org/rfc/rfc4106.txt

[336] Vijay Ganesh and Trevor Hansen and Mate Soos and Dan Liew and Ryan Govostes: STP constraint solver (2007), https://github.com/stp/stp

[337] Vuckovac, R.: Raviyoyla v1. https://competitions.cr.yp.to/round1/raviyoylav1.pdf (2014)

[338] Wagner, D.: The boomerang attack. In: Knudsen, L.R. (ed.) Fast Software Encryption – FSE'99. Lecture Notes in Computer Science, vol. 1636, pp. 156–170. Springer, Heidelberg, Germany, Rome, Italy (Mar 24–26, 1999)

[339] Wagner, D., Schneier, B., Kelsey, J.: Cryptanalysis of the cellular encryption algorithm. In: Kaliski Jr., B.S. (ed.) Advances in Cryptology – CRYPTO'97. Lecture Notes in Computer Science, vol. 1294, pp. 526–537. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 1997)

[340] Wagner, D., Simpson, L., Dawson, E., Kelsey, J., Millan, W., Schneier, B.: Cryptanalysis of ORYX. In: Tavares, S.E., Meijer, H. (eds.) SAC 1998: 5th Annual International Workshop on Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 1556, pp. 296–305. Springer, Heidelberg, Germany, Kingston, Ontario, Canada (Aug 17–18, 1999)

[341] Wang, L.: SHELL v2.0. https://competitions.cr.yp.to/round2/shellv20.pdf (2015)

[342] Wang, L., Guo, J., Zhang, G., Zhao, J., Gu, D.: How to build fully secure tweakable blockciphers from classical blockciphers. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology – ASIACRYPT 2016, Part I. Lecture Notes in Computer Science, vol. 10031, pp. 455–483. Springer, Heidelberg, Germany, Hanoi, Vietnam (Dec 4–8, 2016)

[343] Wang, M., Sun, Y., Tischhauser, E., Preneel, B.: A model for structure attacks, with applications to PRESENT and Serpent. In: Canteaut, A. (ed.) Fast Software Encryption – FSE 2012. Lecture Notes in Computer Science, vol. 7549, pp. 49–68. Springer, Heidelberg, Germany, Washington, DC, USA (Mar 19–21, 2012)

[344] Wang, N., Wang, X., Jia, K.: Improved impossible differential attack on reduced-round LBlock. In: Kwon, S., Yun, A. (eds.) ICISC 15: 18th International Conference on Information Security and Cryptology. Lecture Notes in Computer Science, vol. 9558, pp. 136–152. Springer, Heidelberg, Germany, Seoul, Korea (Nov 25–27, 2016)

[345] Wang, X., Feng, D., Lai, X., Yu, H.: Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD. Cryptology ePrint Archive, Report 2004/199 (2004), http://eprint.iacr.org/2004/199

[346] Weinmann, R.P., Wirt, K.: Analysis of the DVB common scrambling algorithm. In: Chadwick, D., Preneel, B. (eds.) Communications and Multimedia Security. pp. 195–207. Springer US, Boston, MA (2005)

[347] Whiting, D., Housley, R., Ferguson, N.: Counter with CBC-MAC (CCM). RFC 3610 (Sep 2003), https://rfc-editor.org/rfc/rfc3610.txt

[348] Winternitz, R.S.: A secure one-way hash function built from DES. In: 1984 IEEE Symposium on Security and Privacy. pp. 88–88 (April 1984)

[349] Winternitz, R.S.: Producing a one-way hash function from DES. In: Chaum, D. (ed.) Advances in Cryptology – CRYPTO'83. pp. 203–207. Plenum Press, New York, USA, Santa Barbara, CA, USA (1983)

[350] Wu, H.: The stream cipher HC-128 (2004), http://www.ecrypt.eu.org/stream/p3ciphers/hc/hc128_p3.pdf

[351] Wu, H.: ACORN: a Lightweight Authenticated Cipher (vs). https://competitions.cr.yp.to/round2/acornv2.pdf (2015)

[352] Wu, H., Huang, T.: The JAMBU Lightweight Authentication Encryption Mode (v2). https://competitions.cr.yp.to/round2/aesjambuv2.pdf (2015)

[353] Wu, H., Preneel, B.: AEGIS: A Fast Authenticated Encryption Algorithm (v1). https://competitions.cr.yp.to/round1/aegisv1.pdf (2014)

[354] peng XING, J., cheng ZOU, X., GUO, X.: Ultra-low power S-boxes architecture for AES. The Journal of China Universities of Posts and Telecommunications 15(1), 112 – 117 (2008), http://www.sciencedirect.com/science/article/pii/S1005888508600722

[355] Yang, D., feng Qi, W., jin Chen, H.: Impossible differential attack on Qarma family of block ciphers. Cryptology ePrint Archive, Report 2018/334 (2018), https://eprint.iacr.org/2018/334

[356] Yang, D., Qi, W.F., Chen, H.J.: Impossible differential attacks on the SKINNY family of block ciphers. IET Information Security 11(6), 377–385 (2017), https://doi.org/10.1049/iet-ifs.2016.0488

[357] Zhang, B., Shi, Z., Xu, C., Yao, Y., Li, Z.: Sablier v1. https://competitions.cr.yp.to/round1/sablierv1.pdf (2014)

[358] Zhang, B., Xu, C., Feng, D.: Real time cryptanalysis of Bluetooth encryption with condition masking (extended abstract). In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology – CRYPTO 2013, Part I. Lecture Notes in Computer Science, vol. 8042, pp. 165–182. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2013)

[359] Zhang, L., Wu, W., Wang, Y., Wu, S., Zhang, J.: Lac: A lightweight authenticated encryption cipher. https://competitions.cr.yp.to/round1/lacv1.pdf (2014)

[360] Zhang, L., Wu, W., Sui, H., Wang, P.: iFeed[AES] v1. https://competitions.cr.yp.to/round1/ifeedaesv1.pdf (2014)

[361] Zhang, W., Bao, Z., Lin, D., Rijmen, V., Yang, B., Verbauwhede, I.: Rectangle: a bit-slice lightweight block cipher suitable for multiple platforms. Science China Information Sciences 58(12), 1–15 (Dec 2015), https://doi.org/10.1007/s11432-015-5459-7

[362] Zhang, W., Rijmen, V.: Division cryptanalysis of block ciphers with a binary diffusion layer. Cryptology ePrint Archive, Report 2017/188 (2017), http://eprint.iacr.org/2017/188

[363] Zong, R., Dong, X.: Meet-in-the-middle attack on QARMA block cipher. Cryptology ePrint Archive, Report 2016/1160 (2016), http://eprint.iacr.org/2016/1160

[364] Zong, R., Dong, X., Wang, X.: MILP-aided related-tweak/key impossible differential attack and its applications to Qarma, Joltik-BC. Cryptology ePrint Archive, Report 2018/142 (2018), https://eprint.iacr.org/2018/142