Provable Security for Lightweight Message Authentication and Encryption

Inauguraldissertation zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften der Universität Mannheim

vorgelegt von
Alexander Julian Moch, M.Sc.
aus Mannheim

Mannheim, 2023

Dekan: Prof. Dr. Claus Hertling, Universität Mannheim
Erstgutachter: Prof. Dr. Matthias Krause, Universität Mannheim
Zweitgutachter: Prof. Dr. Stefan Lucks, Bauhaus-Universität Weimar
Drittgutachter: Prof. Dr. Frederik Armknecht, Universität Mannheim

Tag der mündlichen Prüfung: 17. Oktober 2023

Abstract

The birthday bound often limits the security of a cryptographic scheme to half of the block size or internal state size. This implies that cryptographic schemes require a block size or internal state size that is twice the security level, resulting in larger and more resource-intensive designs. In this thesis, we introduce abstract constructions for message authentication codes and stream ciphers that we demonstrate to be secure beyond the birthday bound.

Our message authentication codes were inspired by previous work, specifically the message authentication code EWCDM by Cogliati and Seurin, as well as the work by Mennink and Neves, which demonstrates easy proofs of security for the sum of permutations and an improved bound for EWCDM. We enhance the sum of permutations by incorporating a hash value and a nonce in our stateful design, and in our stateless design, we utilize two hash values. One advantage over EWCDM is that the permutation calls, or block cipher calls, can be parallelized, whereas in EWCDM they must be performed sequentially. We demonstrate that our constructions provide a security level of $2n/3$ bits in the nonce-respecting setting. Subsequently, this bound was further improved to $3n/4$ bits of security. Additionally, it was later discovered that security degrades gracefully with nonce repetitions, unlike EWCDM, where the security drops to the birthday bound with a single nonce repetition.

Contemporary stream cipher designs aim to minimize the hardware module's resource requirements by incorporating an externally available resource, all while maintaining a high level of security. The security level is typically measured in relation to the size of the volatile internal state, i.e., the state cells within the cipher's hardware module. Several designs have been proposed that continuously access the externally available non-volatile secret key during keystream generation. However, there exists a generic distinguishing attack with birthday bound complexity. We propose schemes that continuously access the externally available non-volatile initial value. For all constructions, conventional or contemporary, we provide proofs of security against generic attacks in the random oracle model. Notably, stream ciphers that use the non-volatile initial value during keystream generation offer security beyond the birthday bound. Based on these findings, we propose a new stream cipher design called Draco.

Zusammenfassung

Häufig beschränkt das Geburtstagsparadoxon die Sicherheit eines kryptographischen Systems auf die Hälfte der Blockgröße oder der internen Zustandsgröße. Das bedeutet, dass kryptographische Verfahren eine Blockgröße oder interne Zustandsgröße benötigen, die doppelt so groß ist wie das gewünschte Sicherheitsniveau, was zu größeren und ressourcenintensiveren Designs führt. In dieser Arbeit führen wir abstrakte Konstruktionen für Message Authentication Codes (MACs) und Stromchiffren ein, von denen wir zeigen, dass sie über die Geburtstagsgrenze (*Birthday-Bound*) hinaus sicher sind.

Unsere MACs wurden durch frühere Arbeiten inspiriert, insbesondere durch EWCDM von Cogliati und Seurin sowie eine Arbeit von Mennink und Neves, welche einfache Sicherheitsbeweise für die Summe von Permutationen und eine verbesserte untere Schranke für EWCDM zeigt. Wir erweitern die Summe von Permutationen durch Einbindung eines Hashwertes und einer Nonce in unserem ersten Design und in unserem zweiten Design verwenden wir zwei Hashwerte. Ein Vorteil gegenüber EWCDM ist, dass die Permutationsaufrufe oder Blockchiffrenaufrufe parallelisiert werden können, während sie bei EWCDM sequenziell durchgeführt werden müssen. Wir zeigen, dass unsere Konstruktionen ein Sicherheitsniveau von $2n/3$-Bit im *nonce-respecting* Szenario bieten. Diese Schranke wurde später auf ein Sicherheitsniveau von $3n/4$-Bit Sicherheit verbessert. Zusätzlich wurde später entdeckt, dass die Sicherheit bei Wiederholungen der Nonce nur allmählich abnimmt (*gracefully degrades*), im Gegensatz zu EWCDM, bei dem die Sicherheit bei einer einzigen Nonce-Wiederholung auf die Geburtstagsgrenze fällt.

Neuere Stromchiffrendesigns zielen darauf ab, die Ressourcenanforderungen des Hardwaremoduls zu minimieren, indem sie eine extern verfügbare Ressource einbinden, und dabei ein hohes Sicherheitsniveau beibehalten. Das Sicherheitsniveau wird typischerweise in Bezug auf die Größe des volatilen internen Zustands gemessen, d.h., die Zustandszellen innerhalb des Hardware-Moduls der Chiffre. Es wurden mehrere Designs vorgeschlagen, die während der Keystreamgenerierung kontinuierlich auf den extern verfügbaren nicht-volatilen gehemein Schlüssel zugreifen. Es gibt jedoch einen generischen Unterscheidungsangriff mit Birthday-Bound-Komplexität. Wir schlagen Konstruktionen vor, die kontinuierlich auf den extern verfügbaren nicht-volatilen *Initial Value* zugreifen. Für alle Konstruktionen, konventionelle oder neuere Designs, liefern wir Sicherheitsbeweise ge-

gen generische Angriffe im Random Oracle Model. Bemerkenswert ist, dass Stromchiffren, welche den nicht-volatilen Initial Value während der Keystreamgenerierung nutzen, eine Sicherheit über die Geburtstagsgrenze hinaus bieten. Basierend auf diesen Erkenntnissen schlagen wir ein neues Stromchiffrendesign namens DRACO vor.

Acknowledgements

First and foremost, I would like to express my profound gratitude to Matthias Krause for serving as my supervisor. The guidance, suggestions, and inspiration you provided were invaluable to the completion of this journey.

Special thanks to Eik List for assisting me with my first cryptography publication. Further acknowledgment goes to Matthias Hamann and Matthias Krause for their support during the subsequent publications.

My appreciation goes out to my colleagues: Christian Müller, Philipp Schaber, Vasily Mikhalev, Jasmin Zalonis, Jochen Schäfer, Linda Scheu-Hachtel, and Youzhe Heng. Their contributions, though often behind the scenes, haven't gone unnoticed.

I am also deeply appreciative of the behind-the-scenes contributions of Karin Teynor and Gabi Atkinson, allowing me to focus more intently on my research.

I extend my gratitude to Frederik Armknecht and Stefan Lucks for their review and assessment of my thesis. Likewise, I appreciate Andreas Neuenkirch for presiding over my defense and Colin Atkinson for stepping in on short notice during the disputation.

To my parents, thank you for your foundational support in my early years, which has led me to this academic milestone.

Contents

List of Figures

List of Tables

Outline

This thesis is concerned with provable security for lightweight message authentication and stream cipher encryption. Message authentication and stream cipher constructions will be proposed in this work and their security will be shown in the random oracle model. Also, a new stream cipher design called Draco will be proposed based on the proofs of security.

Probability theory. As it is essential to definitions of security, the modeling of our proposed construction and the proofs of security, the fundamentals of probability theory will be reviewed in this thesis. Of particular relevance is the *total variation distance*, occasionally referred to as the *statistical distance*. The total variation distance is used to establish an upper bound an attacker's probability of breaking the cryptosystem.

Security model. Throughout this work, we will consider asymptotic security with an information-theoretic adversary. This means that the scheme will be describes abstractly in dependence of some security parameters, i.e., the key length, the block size, or the internal state size. The adversary is not bounded in computational resources but only in the amount of *queries* it can ask to the cryptographic construction and its underlying building blocks.

The goal of the adversary is to distinguish the construction from a perfectly random counterpart. Intuitively, this corresponds to distinguishing the output of a cryptographic algorithm from random noise. The adversary will get access to an oracle in either the *real* world or the *ideal* world. The real world corresponds to the construction to be analyzed and the ideal world to its perfectly random counterpart. After a total of q queries, the adversary has to decide whether it was interacting with the real world or with the ideal world. Its task is therefore to *distinguish* the two worlds and the difference how much better the adversary is than a random guess is measured as the *distinguishing advantage*.

Birthday bound. Of particular relevance in this work, and for many cryptographic schemes in general, is the birthday bound. If we pick elements from a set of size N uniformly at random with repetition, we expect the first collision after $\sqrt{N}$ random draws. One typical goal when designing new schemes is overcoming the birthday bound. This allows a scheme to have higher security for given resources, or lower resource requirements for a given security level.

Proof technique. The proofs in this work make use of the H-coefficient technique which is due to Jacques Patarin [Pat08]. The setting is that of an information-theoretic adversary that has to distinguish the construction from a perfectly random counterpart. The main idea is that some events, that allow for easy distinguishing, occur with a small probability, while the remaining ones can only be distinguished with a small probability. We describe the H-coefficient technique in Chapter 4 and give an example using the Even-Mansour cipher. For MACs we use the Mirror Theory and also give a short example in Section 4.2.

Message authentication. The message authentication codes (MACs) we propose were inspired by the works [CS16] and [MN17]. [CS16] proposed a beyond the birthday bound-secure nonce-based MAC that requires two consecutive permutation calls. [MN17] showed a simple proof of security with n bits of security for the sum of permutation based on the Mirror Theory by Jacques Patarin [Pat10, Pat17] and the H-coefficients technique by Jacques Patarin [Pat08]. We based our MAC construction on the sum of permutations as it allows for parallel calls to the permutations. One of our schemes is nonce-based (stateful) and the other replaces the nonce with a second hash value and is thus stateless. [DNT19] studied the same nonce-based construction in parallel to us and further showed graceful degradation of security if nonces are repeated. For other schemes, security drops back to the birthday bound if nonces are repeated [CS16] or, even worse, the scheme is broken [WC81]. [DDNP18] studied the same stateless construction in parallel to us and showed the same security bound. For both constructions, the security bounds were later improved [CLLL20, KLL20] and their security was analyzed in a multi-user setting [CDN22, SWGW21, DDNT23].

Stream ciphers. Stream ciphers are vulnerable to an attack type called *time-memory-data tradeoff attacks*. These cap the security of stream ciphers to half the internal state size. This implies a large internal state to be used. Contemporary designs try to keep the cipher's hardware module small by accessing an externally available resource. The total state size, internal plus external state, must still be double the security level, but the cipher's hardware module uses fewer resources.

Our stream cipher constructions were inspired by earlier works that introduced the Sprout stream cipher [AM15] and the Plantlet stream cipher [MAM16]. These designs continuously use the secret key during keystream generation. The secret key is not stored within the cipher's hardware module. But the design makes use of the existing wiring to access the key and keep the cipher's hardware module small.

Extended-state stream ciphers. The problem with designs continuously using the secret key is that the security level is still capped at half the volatile internal state size when considering indistinguishability. That prompted us to design a random oracle model which

would allow to analyze the security of stream ciphers in a generic setting. Using this model, we show tight bounds on the security against generic attacks of stream ciphers. The model was initially presented in [HKM19], which also demonstrated that stream ciphers continuously using the IV do not suffer from a generic distinguishing attack. Based on this, the Draco stream cipher [HMKM22] was designed, which additionally uses a non-volatile key prefix to achieve a higher security level. The proof of security in Draco is based on a model equivalent to that of [HKM19], but makes use of the H-coefficients technique and provides much simpler proofs of security. The model was applied to all four constructions in [Moc23]. As the original version of Draco was attacked, we present a fix as a work in progress towards the end of this thesis.

Publications. This thesis is based on a total of four publications [ML19, HKM19, Moc23, HMKM22]. [ML19] presents two message authentication constructions and shows their security in the random oracle model. It is joint work with Eik List and was presented at *Applied Cryptography and Network Security 2019* (ACNS 2019). [Moc23] proves the security for four stream cipher constructions in the random oracle model. It was accepted for publication in the *Journal of Mathematical Cryptology – Volume 17, Issue 1* (JMC 17.1). [Moc23] extends the earlier work [HKM19], which is joint work with Matthias Hamann and Matthias Krause and was presented at *Selected Areas in Cryptography 2019* (SAC 2019). [HMKM22] presents the Draco stream cipher that builds upon one of the four stream cipher constructions proven secure in the random oracle model. It is joint work with Matthias Hamann, Matthias Krause, and Vasily Mikhalev and was published in *IACR Transactions on Symmetric Cryptology – Volume 2022, Issue 2* (ToSC 2022.2) and presented at the *Fast Software Encryption 2023* (FSE 2023) conference.

Structure. This thesis is structured into three parts. The first part presents a general introduction into the topic in Chapter 1, reviews the basics in probability theory in Chapter 2, and explains the notion of cryptographic security that we will be using as well as the proof technique in Chapter 3 and Chapter 4.

The second part presents the basics of message authentication codes and the Wegman-Carter MAC in particular in Chapter 5. Chapter 6 presents the two message authentication codes based on the sum of permutations from [ML19] and their proof of security as presented at ACNS 2019.

The third part presents an introduction into stream ciphers in Chapter 7. Chapter 8 and Chapter 9 are based on [Moc23] and present the enhanced-state stream cipher constructions as well as their proofs of security as accepted for JMC 17.1. Chapter 10 presents the Draco stream cipher from [HMKM22] in its original version as presented at FSE 2023. The original version of Draco contains a glitch in the key schedule, and Chapter 11 presents a work in progress on possible solutions on how to fix this.

Part I

Foundations

1 | Introduction

Secure communication matters in the digital age more than ever. From browsing the internet, composing emails and instant messages, to performing complex operations like exchanging medical records, conducting electronic banking, or managing access control to buildings, ensuring the security of these activities necessitates the utilization of cryptographic algorithms. These algorithms play a crucial role in safeguarding the integrity and confidentiality of data, preventing any unauthorized entity from gaining unauthorized access or tampering with the information. When communicating over the internet, in particular, each message (or data packet) travels through several routers before it reaches its destination. This allows an attacker to intercept, modify, or inject a message between two legitimate parties. This is to be avoided.

Goals. Although cryptography is primarily known for its role in preserving *confidentiality*, which involves keeping information secret, its scope extends beyond that aspect. It also encompasses ensuring the *integrity* of a message, confirming the *authenticity* of the communication partner, and preventing the sender from denying their involvement in sending a message, i.e., *plausible deniability*. These additional objectives are commonly recognized as important goals of cryptography.

Challenges. Everyday life depends more and more on electronic devices. Desktop computers, smartphones, and smart cards all have different levels of computing power, and efficiency is a critical factor in their operation. A smart card may only be powered by a magnetic field, so an algorithm with high power demands is out of the question. A smartphone may run out of battery early if the algorithm is too computationally intensive. For a desktop computer or generally a more powerful computer connected to a power supply, one wants to optimize the speed of the algorithm for high-performance scenarios. Furthermore, these systems may want to communicate with one another, so compatibility is another key factor. The Internet of Things aims to connect an increasing number of everyday devices, making it essential to strike a balance between circuit power, efficiency, speed, and hardware module size. This balance is critical to ensure seamless interoperability between devices.

Cryptographic security. The next challenge is to define the requirements needed for a cryptographic algorithm and, in particular, to also define an attacker's capabilities. Different goals and use cases have different requirements for their security. Considering the continuous advancement in computing power, it is imperative to anticipate potential future threats to cryptographic schemes. A system designed at present should ideally remain secure even after several decades. Consequently, it becomes essential to establish the requirements in advance and develop cryptographic algorithms that align with these specifications. By adopting this approach, we can ensure the long-term security of the system against potential advancements in computational capabilities. Under assumptions, i.e., that certain problems are *hard* or some building blocks of a cryptographic scheme are replaced with ideal counterparts, proofs of security may be possible.

Historically, ciphers were typically designed in an ad-hoc fashion, meaning they were not specifically created to adhere to predetermined security criteria or definitions. Today, security is defined in advance, and schemes are built to meet the requirements. Ad-hoc designs were sufficient for the time, as before the advent of computers, humans had to perform the task of analyzing ciphertexts and possibly decrypting them. Today's computers can handle several billions of computations per second. By today's standards, classical ciphers like Caesar's cipher or the Vigenere cipher are considered insecure.

Lightweight cryptography. Many usage scenarios require low-powered or resource-constrained devices. These may be smart cards, sensors, or embedded systems. For efficiency reasons, these devices often still use very small key lengths that are comparatively easy to break using exhaustive search, and thus, these devices are considered insecure by today's standards. The Internet of Things connects lots of devices with little computational power. To improve security, new cipher designs attempt to reach a higher security level while still requiring only low resources. One particular obstacle is the so-called birthday bound.

Birthday bound. Hash functions, stream ciphers, and message authentication codes all suffer from the birthday bound. That is, to reach a certain security level, the internal state, respectively the block size, needs to be twice as large as the desired security level. Conversely, given a bound on the resources, only a certain security level may be reached. Designing ciphers overcoming the birthday bound implies smaller resource requirements for a given security level. A common goal of many cryptographic schemes is to overcome the birthday bound and allow for a higher security level for lightweight devices.

Openness. Having a cryptographic scheme published is considered very important today. Cryptographers often cite *Kerckhoffs' Principle* in this context.

1.1 Kerckhoffs' Principle

In 1883, Auguste Kerckhoffs published an article [Pet83] in which he stated six design principles for military ciphers. The second one is still cited throughout the literature on cryptography and became known as *Kerckhoffs' Principle*.

> Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi.
>
> [It must not require secrecy, and it must be able to fall into the hands of the enemy without inconvenience.][1]

According to Kerckhoffs' Principle, the details of a cryptosystem may be publicly known, and the security should only depend on the secrecy of the key. Keeping the details of a cipher secure may become intractable over time. Details may leak to an adversary, and replacing the cipher itself is much more cumbersome than agreeing on a new secret key. Furthermore, this will allow the cryptosystem to be analyzed by a large group of experts instead of only those with access to the cipher. Weaknesses are more likely to be spotted, reported to the designers, and fixed.

Heuristic process. New schemes are introduced through research papers, which are openly shared with the public. These papers undergo a rigorous process that involves review, conference presentations, publication, and extensive scrutiny by experts. The analyses conducted on these schemes are subsequently presented and published as well. As vulnerabilities come to light, the original designers enhance the scheme to address these issues, initiating a cycle of iterative improvement.

Eventually, this process is heuristic is nature: If there has been no significant attack on a scheme after being analyzed for more than 20 years, it is probably considered to be very good.

Defining security. It directly follows that any cryptosystem must be analyzed for weaknesses prior to its usage. Kerckhoffs' Principle does not state what it means for a cipher to be secure. Also, it does not state what tools may be used to assess its security. There are multiple approaches to security; the three most well-known are perfect secrecy, computational indistinguishability, and information-theoretic indistinguishability. All of those approaches require some background in probability theory. Thus, the next chapter introduces the necessary basics in probability theory.

[1] Translated using DeepL: `https://www.deepl.com/translator`

2 | Probability Theory

Cryptographic schemes used in practice are rarely perfectly secure. They allow a *very* small chance to break the scheme. We will define perfect secrecy later as we need the foundations in probability theory first. Later, in Subsection 3.1.1, we will present a perfectly secure cipher, prove it to be perfectly secure, and argue why it is not a feasible scheme in practice.

Resource-bounded adversaries. In cryptographic proofs, it is common to consider a resource-bounded adversary that attempts to compromise a cryptographic algorithm. Taking into account the limited resources of the adversary, a proof of security typically provides an upper bound for the adversary's probability of successfully breaking the cryptographic scheme. To continue, we need to introduce the necessary concepts.

Necessary foundations. In this section, we will review the necessary concepts of probability theory that will be used throughout this work. We will follow the excellent book by Mittelbach and Fischlin [MF21] closely in this section.

2.1 Basic Definitions

Cryptographic proofs usually consist of some form of a game in which an adversary tries to 'win' against the cryptosystem. Usually, the adversary is given a challenge and has to distinguish between two cases. In the beginning of the game, some form of randomness is sampled. That may be a secret key or idealized primitives, i.e., random permutations or random functions.

The adversary will then ask questions to the cryptosystem and receive answers. These questions asked by the adversary are typically called *queries*. After querying the cryptosystem, the adversary has to output a decision bit. The course of the game, i.e., the query-answer pairs, as well as the random key or the random primitives, can be modeled by a probability distribution. A probability distribution assigns a probability to each outcome of an experiment. In this case, by 'outcome', we mean the course of the game. Note that we will only be concerned with discrete probability distributions.

Definition 2.1 (***Probability distribution***)

Let $S = \{s_1, \ldots, s_n\}$ be a finite set. We call $D_S = \{p_1, \ldots, p_n\}$ a *probability distribution*

on S if $|S| = |D_S|$, $p_i \geq 0$ for all $i \in \{1, \ldots, n\}$ and

$$\sum_{i=1}^{n} p_i = 1.$$

With each element $s_i \in S$ we associate the probability p_i, denoted by $\Pr[s_i]$.

There is one distribution that we are particularly interested in, the *uniform distribution*. The uniform distribution is a type of probability distribution where all outcomes are equally likely.

Definition 2.2 (*Uniform distribution*)

Let $S = \{s_1, \ldots, s_n\}$ be a finite set and let $U_S = \{p_1, \ldots, p_n\}$ be a probability distribution on S. U_S is called the *uniform distribution* on S if $p_i = p_j$ for all $i, j \in \{1, \ldots, n\}$.

If we sample an element s from S with regard to the uniform distribution this is denoted by $s \twoheadleftarrow S$.

A set of outcomes of a probability experiment is called an *event* and we can associate a probability with each event:

Definition 2.3 (*Event*)

A subset $E \subseteq D_S$ is called an *event* and its probability denoted by $\Pr[E]$ is defined by

$$\Pr[E] = \sum_{s \in E} \Pr[s].$$

A random variable is a function based on the outcomes of the random experiment. In this work, we will restrict ourselves to the discrete and finite case:

Definition 2.4 (*Random variable*)

A *random variable* X consists of a finite set $\mathcal{X} = \{x_1, \ldots, x_n\}$ together with an associated probability distribution $D_X = \{p_1, \ldots, p_n\}$. The probability that random variable X takes on value x_i is p_i and is denoted by $\Pr[X = x_i]$.

Often, it is interesting what the expected outcome of an experiment is. The *expected value*, or *mean*, represents the average value of a random variable. Each outcome of the random variable X is weighted by its probability:

Definition 2.5 (*Expectation*)

Let X be a random variable with outcomes $\mathcal{X} = \{x_i, \ldots, x_n\}$ and the associated probability distribution $D_X = \{p_1, \ldots, p_n\}$. Then the *expectation* of X is defined as

$$\mathbb{E}[X] = \sum_{i=1}^{n} x_i p_i.$$

Note that the expected value need not be a possible outcome of the random experiment. For example, the expected value of a roll of a die is 3.5.

We will now look at a couple of more notations and properties that will be necessary in this work. With more than one random variable, we can consider their *joint probability*.

Definition 2.6 (*Joint probability*)

Let X and Y be two random variables. The *joint probability* $\Pr[X = x \wedge Y = y]$ denotes the probability that X takes on value x *and* Y takes on value y. When considering two events A and B then $\Pr[A \cap B]$ is also called the *joint probability* of A and B.

Using the joint probability, we can formulate the probability of *either* X or Y occurring and further we can give a simple upper bound on the probability:

Lemma 2.1 (*Union bound*)

The probability that $X = x$ or $Y = y$ is given by

$$\Pr[X \vee Y] = \Pr[X] + \Pr[Y] - \Pr[X \wedge Y].$$

Similarly for two events A and B, we have that

$$\begin{aligned}\Pr[A \cup B] &= \Pr[A] + \Pr[B] - \Pr[A \cap B] \\ &\leq \Pr[A] + \Pr[B].\end{aligned}$$

The union bound generalized to n events E_i, $i \in \{1, \ldots, n\}$:

$$\Pr\left[\bigcup_{i=1}^{n} E_i\right] \leq \sum_{i=1}^{n} \Pr[E_i].$$

The union bound is also known as *Boole's Inequality*.

Given the joint probability of X and Y, summing over all outcomes of Y, we can obtain the *marginal probability*:

Definition 2.7 (*Marginal probability*)

Given the joint probability of X and Y we can obtain the *marginal probability* $\Pr[X = x]$ via summation:

$$\Pr[X = x] = \sum_{y \in \mathcal{Y}} \Pr[X = x \wedge Y = y].$$

We will often consider the probability of the outcome of one random variable given the

outcome of the other. This is called the *conditional probability*:

Definition 2.8 (***Conditional probability***)

Let X and Y be two random variables. Then the *conditional probability* $\Pr[X = x \mid Y = y]$ denotes the probability that X takes on value x given that Y takes on value y.

The *law of total probability* is a fundamental principle in probability theory. It allows to determine the probability of an outcome if only conditional probabilities and the probability of the conditioning random variable are known.

Lemma 2.2 (***Law of total probability***)

By combining marginalization and conditional probabilities we can state the *law of total probability*:

$$\Pr[X] = \sum_{y \in \mathcal{Y}} \Pr[X \mid Y = y] \cdot \Pr[Y = y].$$

The relationship between the conditional probabilities of 'X given Y' and 'Y given X' is given by *Bayes' Theorem*:

Theorem 2.1 (***Bayes' Theorem***)

Let X and Y be two random variables. Assuming $\Pr[Y = y]$, we have that

$$\Pr[X = x \mid Y = y] = \frac{\Pr[Y = y \mid X = x] \cdot \Pr[X = x]}{\Pr[Y = y]}.$$

An important property of random variables is that of *independence*. Two events are called independent in probability theory if the occurrence of one event does not affect the probability of the other event

Definition 2.9 (***Independence***)

Two events A and B are called *independent* if, and only if,

$$\Pr[A \cap B] = \Pr[A] \cdot \Pr[B].$$

Two random variables X and Y are called *independent* if for all pairs of events $A \subseteq \mathcal{X}$ and $B \subseteq \mathcal{Y}$, A and B are independent.

Later, in our proofs, we will upper bound the 'advantage' of an adversary. That is, how much better than a coin toss can an adversary distinguish two distributions. To do so, we will need the so-called *total variation distance*. Loosely speaking, it is the distance between two probability distributions, see Figure 2.1.

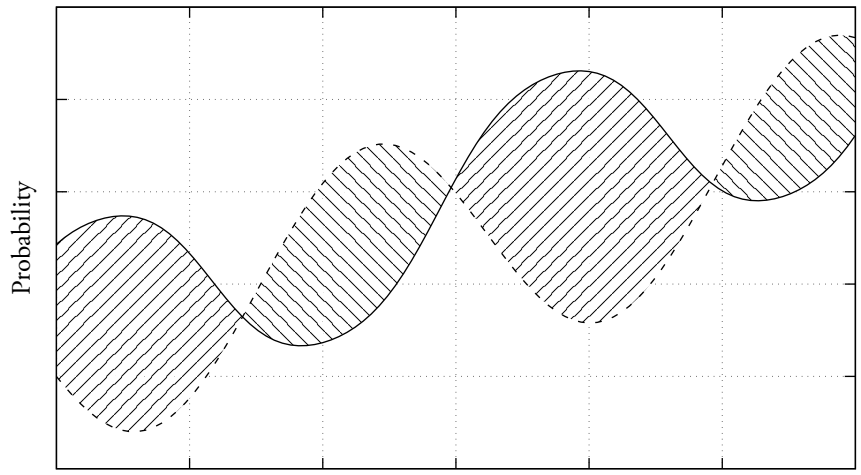

Figure 2.1: Visualization of the total variation distance. For two probability distributions (solid and dashed lines), the total variation distance describes the area between the lines scaled to a value between 0 and 1.

Definition 2.10 (***Total variation distance***)

Let X and Y be two random variables over the set $\mathscr{Z}$ and let D_X and D_Y be their respective probability distributions. The *total variation distance* is then defined to be

$$||X - Y||_{\mathrm{TV}} = \frac{1}{2} \sum_{z \in \mathscr{Z}} |\Pr[X = z] - \Pr[Y = z]| .$$

The total variation distance is sometimes just called *statistical distance*.

We will also need Markov's Inequality in our proofs.

Lemma 2.3 (***Markov's Inequality***)

Let $X \in \mathbb{R}_{\geq 0}$ be a nonnegative random variable and $a \in \mathbb{R}_{>0}$. Then, the probability that X is at least a is at most the expectation of X divided by a:

$$\Pr[X \geq a] \leq \frac{\mathbb{E}(X)}{a}.$$

Proof. Let $\mathscr{X}$ denote the underlying set of the random variable X. Let $\mathscr{X}_{\geq a}$ denote all $x \in \mathscr{X}$ where $x \geq a$. Then, we have that:

$$\mathbb{E}(X) = \sum_{x \in \mathscr{X}} \Pr[X = x] \cdot x$$

$$\geq \sum_{x \in \mathcal{X}_{\geq a}} \Pr[X = x] \cdot x$$
$$\geq \sum_{x \in \mathcal{X}_{\geq a}} \Pr[X = x] \cdot a$$
$$= a \cdot \Pr[X \geq a].$$

Rearranging the terms yields the claim. ■

2.2 The Birthday Paradox

Overcoming the birthday bound is of particular interest for lightweight cryptography. This would allow to reach a predefined security level with less resources, i.e., a smaller internal state or smaller block sized if block ciphers are used. Alternatively, with fixed resources, a higher security level may be achieved.

The birthday paradox is commonly used in introductory courses in probability theory to demonstrate that, when probabilities are intuitively estimated, the estimation is often wrong. Assuming that birthdays are uniformly distributed throughout a year of 365 days, it asks the following question:

> How many people do have to be in a room such that the probability of a shared birthday exceeds 50%?

The answer is 23 which seems counterintuitive to many and typical guesses tend to be way higher than that.[1] Later in this section, a simple proof for this phenomenon will be presented.

Use in cryptography. The birthday paradox has high relevance in cryptography, particularly for hashing, message authentication, and stream ciphers, and is thus of particular relevance for this work. A frequent goal of an attacker against a cryptographic scheme is to obtain a collision between two sets. These may be two sets of hash values in the case of a hash function or two sets of output keystream bits in the case of a stream cipher. Typically, one set was *observed* by the attacker, and the other set was *generated* by the attacker. If a collision was observed, the attacker can obtain, with high probability, the input value to the hash function or the internal state of the stream cipher that generated the relevant keystream.

2.2.1 Calculating the Probability

Despite being called the *birthday paradox*, it is not a true paradox, but rather a counterintuitive result that often surprises many. In this subsection, the probability of a birthday

[1] Birthdays are not uniformly distributed and thus, the actual number may be lower than 23.

collision in n people in a year of 365 days will be calculated. The following subsection generalizes it to years with d days.

Lemma 2.4 (***The birthday paradox***)

Assume that birthdays are uniformly distributed throughout a year of 365 days. There have to be at least 23 people in a room such that the probability of a shared birthday exceeds 0.5.

Proof. Consider the event that no collision occurs for i people in the room, denoted by NoColl_i. It is straightforward to see that $\mathsf{NoColl}_0 = \mathsf{NoColl}_1 = 1$. Further, we can calculate NoColl_n as follows:

$$\mathsf{NoColl}_n = \prod_{i=0}^{n-1} \Pr[\mathsf{NoColl}_{i+1} \mid \mathsf{NoColl}_i].$$

The conditional probabilities $\Pr[\mathsf{NoColl}_{i+1} \mid \mathsf{NoColl}_i]$ are easy to determine: The i people in the room all have distinct birthdays, therefore, the probability for the $(i+1)$-th person entering the room to cause a collision is $i/365$. Consequently, the probability for the i-th person to not cause a collision is $1 - i/365$. Thus, we obtain

$$\mathsf{NoColl}_n = \prod_{i=0}^{n-1} \left(1 - \frac{i}{365}\right).$$

In particular we obtain $1 - \Pr[\mathsf{NoColl}_{22}] \approx 0.476$ and $1 - \Pr\left[\mathsf{NoColl}_{23}\right] \approx 0.507$. The probabilities for up to 70 people can be obtained from Figure 2.2. ■

2.2.2 The Birthday Problem Generalized

The birthday problem is used throughout cryptography to calculate the collision probability of various values. These may be internal states for stream ciphers, tags for message authentication codes, or hash values. Hence, we need to consider larger values than 365 for our underlying set. We will later introduce the concept of asymptotic security that will be used throughout this work. Thus, we will parameterize the number of days d in a year and analyze the birthday problem asymptotically in d.

Arbitrary number of days. The birthday problem can be generalized to a year with an arbitrary number of days d. Typically, we are interested in collisions between elements drawn uniformly at random from a set of size d. In the following, we will present the lower and upper bounds on the number of people in a room. For a more detailed description, please refer to [KL20] and [Bri12].

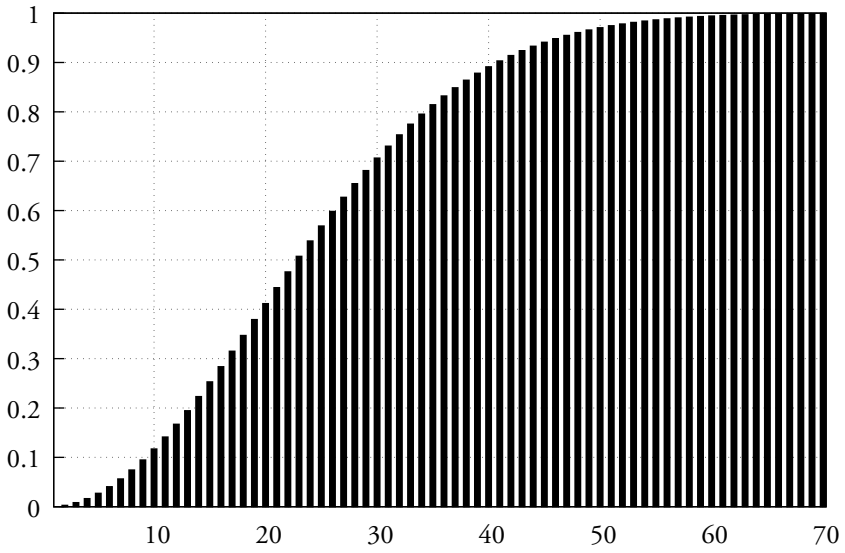

Figure 2.2: The probabilities of a collision for 0 up to 70 people in a year of 365 days.

Bounding the probability. We already explained how to calculate the probability of a collision for a year with 365 days in Section 2.2.1 to be

$$\mathsf{NoColl}_n = \prod_{i=0}^{n-1}\left(1-\frac{i}{365}\right).$$

Adjusting the term for a year with d days, we obtain:

$$\mathsf{NoColl}_n^d = \prod_{i=0}^{n-1}\left(1-\frac{i}{d}\right).$$

Following [Bri12], the term NoColl_n^d can be lower and upper bounded by:

$$\mathsf{NoColl}_n^d \geq 1-\sum_{i=1}^{n-1}\frac{i}{d} = 1-\left(\frac{1}{d}+\frac{2}{d}+\cdots+\frac{n-1}{d}\right) = 1-\frac{n^2-n}{d}$$

$$\mathsf{NoColl}_n^d \leq \prod_{i=1}^{n-1} e^{-\frac{i}{d}} = e^{-\left(\frac{1}{d}+\frac{2}{d}+\cdots+\frac{n-1}{d}\right)} = e^{-\frac{n^2-n}{d}}$$

Furthermore, [Bri12] shows that to obtain a collision probability of $1/2$, the number of people must always be either $\lceil\sqrt{2d\ln d}\rceil$ or $\lceil\sqrt{2d\ln d}\rceil+1$. Asymptotically, the number of people required to obtain a collision in a year with d days is typically approximated by $\sqrt{d}$.

3 | Cryptographic Security

In this chapter, we wish to introduce the notion of security that we will be working with. We will begin by introducing perfect secrecy as the strongest form of security. While desirable, perfect secrecy is not practical, as it requires secret keys to be exchanged that have the same size as the messages to be transmitted. Therefore, it is necessary to relax the notion of security. Typically, a computationally-bounded adversary is considered. This adversary has limited computational power and is allowed to succeed in breaking the scheme with a very small probability.

Information-theoretic adversaries. In our proofs, we will be concerned with an information-theoretic adversary instead of a computationally-bounded adversary. This type of adversary is not limited in its computational power but only in the amount of queries it asks to the cryptosystem. This type of adversary is clearly stronger than a computationally-bounded adversary, but assuming a stronger adversary makes the proofs of security easier.

Pseudorandomness. The adversary will get to query the cryptosystem through what is called an *oracle* and is limited in its number of queries to the oracles. Its task is to distinguish the cryptosystem from a perfectly random counterpart. Its ability to do so better than a random guess will be captured in what is called the *adversarial advantage*. If this advantage is *negligible*, the cryptosystem is called *pseudorandom*. These terms will be defined more rigorously later in this chapter.

Determinism. In this thesis, we will only consider deterministic algorithms. Any sources of randomness will be sampled in advance, such as the secret key, or the random permutations and random functions that will be required later. The algorithm will utilize the randomly sampled values either as inputs (e.g., secret key) or get oracle access to them (e.g., random permutations and random functions).

3.1 Perfect Secrecy

We wish to define what it means for a cryptographic scheme to be secure. Ideally, it is unbreakable. We will therefore first introduce the notion of *perfect secrecy* and show that

it is an impractical notion of security. This will then be used to motivate more practical definitions. Katz and Lindell [KL20] define perfect secrecy as follows:

Definition 3.1 (***Perfect secrecy***)

An encryption scheme with message space $\mathcal{M}$ is *perfectly secret* if for every probability distribution for M, every message $m \in \mathcal{M}$, and every ciphertext $c \in \mathcal{C}$ for which $\Pr[C = c] > 0$:

$$\Pr[M = m \mid C = c] = \Pr[M = m]$$

Loosely speaking, perfect secrecy means that the knowledge of the ciphertext reveals no information about the plaintext.

3.1.1 One-time Pad

The one-time pad is introduced here for two reasons: (1) it is perfectly secret and is the only encryption scheme to be perfectly secret, and (2) we will later mimic the one-time pad encryption by using stream ciphers that use a much shorter key.

Definition 3.2 (***One-time pad***)

Consider key space $\mathcal{K}$, message space $\mathcal{M}$, and ciphertext space $\mathcal{C}$, where $\mathcal{K} = \mathcal{M} = \mathcal{C} = \{0,1\}^\ell$ for some $\ell \in \mathbb{N}$. Let $k \twoheadleftarrow \mathcal{K}$ be the secret key drawn uniformly at random. A message $m \in \mathcal{M}$ is encrypted by adding the secret key k to m to obtain the ciphertext $c \in \mathcal{C}$:

$$c = m \oplus k.$$

Analogously, a ciphertext $c \in \mathcal{C}$ is decrypted by adding the secret key k to c:

$$m = c \oplus k.$$

Next, we will show that the one-time pad is perfectly secret.

Theorem 3.1

The one-time pad is perfectly secret.

Proof. Fix some $m \in \mathcal{M}$ and $c \in \mathcal{C}$. For the one-time pad, we have that

$$\begin{aligned}\Pr[C = c \mid M = m] &= \Pr[k \oplus m = c \mid M = m] \\ &= \Pr[k = m \oplus c \mid M = m] \\ &= 2^{-\ell}.\end{aligned}$$

Here, the first equality is by definition, and the last equality is due to the key k being drawn uniformly at random from the key space $\mathcal{K}$, independently of the message m.

Fix any distribution over $\mathcal{M}$. Using the above, we see that for any ciphertext $c \in \mathcal{C}$, we have that:

$$\begin{aligned}\Pr[C = c] &= \sum_{m \in \mathcal{M}} \Pr[C = c \mid M = m] \cdot \Pr[M = m] \\ &= 2^{-\ell} \cdot \sum_{m \in \mathcal{M}} \Pr[M = m] \\ &= 2^{-\ell}.\end{aligned}$$

Using Bayes' Theorem, we obtain

$$\begin{aligned}\Pr[M = m \mid C = c] &= \frac{\Pr[C = c \mid M = m] \cdot \Pr[M = m]}{\Pr[C = c]} \\ &= \frac{2^{-\ell} \cdot \Pr[M = m]}{2^{-\ell}} \\ &= \Pr[M = m]\end{aligned}$$

which concludes the proof. ■

3.1.2 Shannon's Theorem

We now know that the one-time pad is perfectly secret. One might wonder if there are other schemes that are perfectly secret, in particular ones with a smaller key size. For practical purposes, it is impractical to share a key that is as large as the message itself. In fact, this key size is required for achieving perfect secrecy.

Optimality of the OTP. Claude Shannon not only proved the one-time pad to be perfectly secure, he also proved it to be optimal [Sha49]. In particular, he showed that the key space and the message space must be equal in size, every key must be chosen uniformly at random and for every message-ciphertext pair (m, c) there exists exactly one key k such that the encryption algorithm $\mathsf{Enc}(k, m)$ outputs c. We show that the key space and the message space must be equal in size to obtain perfect secrecy [MF21].

Theorem 3.2

If an encryption scheme is perfectly secret, then $|\mathcal{K}| = |\mathcal{M}|$.

Proof. Assume that $|\mathcal{K}| < |\mathcal{M}|$. Fix a ciphertext $c \in \mathcal{C}$ with $\Pr[C = c] > 0$. Let M_c be the set of messages $m \in \mathcal{M}$ that c can be decrypted to:

$$M_c = \{m \in \mathcal{M} : \exists k \in \mathcal{K} \text{ such that } m = \mathsf{Dec}(k, c)\}.$$

As the decryption algorithm is deterministic, we have that $|M_c| \leq |\mathcal{K}| < |\mathcal{M}|$. This implies the existence of a message $m' \in \mathcal{M}$ such that $m' \notin M_c$. This means that m' cannot be

encrypted to c. Thus, given $C = c$, we know that

$$\Pr[M = m' | C = c] = 0.$$

For an arbitrary distribution over the messages such that $\Pr[M = m'] > 0$, we get

$$\Pr[M = m'] \neq \Pr[M = m' \mid C = c],$$

which concludes the proof. ■

3.2 Practical Security

It is apparent that the one-time pad is not a practical scheme and that perfect secrecy is not a viable goal in practice. We may still allow an adversary a small chance of breaking a scheme, even if that means sacrificing perfect secrecy. This will then allow for much smaller key sizes, typically around 128 to 256 bits, which is a length between 32 and 64 hexadecimal characters. It is much easier to exchange small keys using dedicated key exchange algorithms. These small keys will then be used to encrypt a much larger amount of data. A good cryptographic scheme has a security level that is not much smaller than its key length. Guessing a secret key with a length of 128 bits is successful with a probability of 2^{-128}. This is considered infeasible by today's standards.

Computational security. It may not be necessary to prevent the leakage of any information, except for the length of a message, to a computationally unbounded attacker. Typically, the literature on cryptography [KL20] considers the notion of *computational security*, which has two relaxations:

1. The computational power of the adversary is bounded so that the adversary only may take a feasible amount of time.

2. The adversary will succeed in breaking the scheme with very small probability.

There are two approaches to describe what is meant by 'feasible amount of time' and 'very small probability': the concrete approach and the asymptotic approach [KL20].

The concrete approach. The concrete approach would consider an adversary running for a specified amount of time, succeeding with a specified probability. An example would be an adversary using the fastest supercomputer to date and running it for 100 years, not being able to succeed in breaking the scheme with a probability higher than 2^{-80}. Alternate measures of time may be the CPU cycles needed. This type of security is important in practice but has its own shortcomings: Does the adversary have specialized hardware? How does future technology impact the success probability? How does the success probability

scale if the adversary runs for half the amount of time? How does the success probability scale if the adversary runs for double the amount of time?

The asymptotic approach. In this work, we will adopt the asymptotic approach. Schemes will be described abstractly and employ a security parameter n that all parties, good and bad alike, know. Throughout this work, the security parameter n will be synonymous with the key length or the internal state length. Rather than requiring fixed values, the computational approach limits the running time of adversaries to be a polynomial in n and the success probability to be negligible in n. In this context, *negligible* means smaller than any inverse polynomial in n.

Information-theoretic adversaries. In our proofs, we will not be concerned with computationally bounded adversaries and instead focus on information-theoretic adversaries. These adversaries are not bounded in their computational resources but rather the number of *queries* they can ask to a cryptosystem. An upper bound on the adversary's success probability will then be given depending on the number of queries the adversary has asked or is limited to. This type of adversary is stronger than the computationally bounded adversary. This makes proofs of security easier, as the bounds ultimately only depend on the number of queries q and the security parameter n. The bounds on the adversary's success probabilities will be of the form $q^\alpha/2^{\beta n}$, where α and β denote some constants, and we aim to do asymptotically better than $\alpha = 2$ and $\beta = 1$ (birthday bound). We will only focus on bounds that are better than the birthday bound, and these bounds are inherently negligible.

3.3 Turing Machines

The Turing machine is a theoretical computing device used to mimic the behavior of a general-purpose computer. It is used for the theoretical analysis of algorithms, particularly their running time. This will be of interest to us as we will model the adversary as a Turing machine, specifically an oracle Turing machine with access to a special oracle tape, or possibly multiple ones. This section aims to give an informal overview of Turing machines.

Oracles. We will analyze the schemes presented in this work in the random oracle model. The random oracle models for stream ciphers and message authentication codes will be presented later. In the random oracle model, an adversary is given access to a set of oracles it may ask questions to. Instead of bounding the computational capabilities of the adversary, the number of queries to the oracles is bounded. In this section, we wish to give a short overview of Turing machines and, in particular, oracle Turing machines. When we speak of an adversary or distinguisher, we mean an algorithm, which is abstracted by a Turing

machine. As adversaries are fundamentally algorithms or Turing machines, we refer to them by the pronoun 'it'.

Definition. A Turing machine consists of an infinitely long tape divided into cells and a head that performs read and write operations on the tape and can move to the left or right. Within the cells are symbols according to an alphabet Σ and a special blank character _. The machine is in a state $q \in \mathcal{Q}$ and changes its state and the content of the cell according to a transition function δ. A Turing machine can be formalized as follows:

Definition 3.3 (*Turing machine*)

A *Turing machine* is defined as a six tuple $M = (\Sigma, \_, \mathcal{Q}, \delta, q_{\mathsf{start}}, q_{\mathsf{halt}})$. Σ describes the alphabet, _ is a special blank symbol, $\mathcal{Q}$ is the finite set of states, $q_{\mathsf{start}} \in \mathcal{Q}$ is the starting state, $q_{\mathsf{halt}} \in \mathcal{Q}$ is the halting state. $\delta : \mathcal{Q} \times \Sigma \cup \{\_\} \to \mathcal{Q} \times \Sigma \cup \{\_\} \times \{\curvearrowleft, \curvearrowright\}$ is called the transition function. It maps the current state and the currently read symbol to the new state, written symbol and a head movement left ($\curvearrowleft$) or right ($\curvearrowright$).

3.3.1 Probabilistic Turing Machines

The Turing machine, as defined above, will always behave the same and deliver the same output when executed on a specific input. However, many applications are modeled as probabilistic, where the algorithm can make random choices, such as key generation or an adversary guessing values.

Random tape. To handle probabilistic behavior, an additional tape called the *random tape* is used, along with a read head corresponding to the random tape. The random tape contains randomly chosen values from the Turing machine's alphabet. The transition function δ is adjusted to consider the current cell content on the random tape when randomness is needed, and then the head of the random tape is moved by one cell.

3.3.2 Oracle Turing Machines

In our proofs, we will consider an information-theoretic distinguisher that asks a limited number of queries to the cryptosystem. The cryptosystem and some of its components will be exposed as oracles to the adversary. As the adversary is modeled as a Turing machine, we will define the *oracle Turing machine* in this section. An oracle Turing machine is enhanced with one or multiple oracle tapes to which the adversary writes its questions. The number of queries to these oracle tapes will be limited.

Oracles as black box. Many cryptographic proofs make use of *oracles*. An oracle receives an input and produces the corresponding output. It is not known how the oracle

works, and the oracle does not need to be implementable. The internals of the oracle are hidden from the algorithm that makes use of it.

A Turing machine M with access to an oracle $\mathcal{O}$ is written as $M^{\mathcal{O}}$. The Turing machine M is extended by an oracle tape for $\mathcal{O}$. $M^{\mathcal{O}}$ writes the input to $\mathcal{O}$ on the oracle tape, enters the oracle state, and after one time step, the answer is written on the oracle tape. $M^{\mathcal{O}}$ may then proceed with its calculations.

In our analyses, we will have a distinguisher (i.e., an algorithm or Turing machine) with access to a set of oracles. The running time of the distinguisher will not be bounded, but the number of queries to the oracles will be bounded by a parameter q. The success probability of the distinguisher will be expressed as a function of q.

Remark 3.1

Often, the adversary has oracle access to multiple building blocks. This can be described as a single oracle with multiple interfaces, where the adversary specifies in its input to the oracle which building block to query. Alternatively, the adversary may have access to multiple oracle tapes, each corresponding to one building block.

3.4 Adversarial Advantage

Intuitively, the adversarial advantage represents how much better the adversary's answer is compared to a purely random guess or a coin flip. One definition measures the difference of $1/2$ to the probability of a correct answer by the adversary. We will provide an equivalent definition in which an adversary distinguishes between two schemes.

Lower bound on security. Proofs of security typically aim to provide an upper bound on the adversarial advantage, i.e., how much better than chance the adversary is at distinguishing the scheme being proven secure from an idealized scheme. In our case, this will be parameterized by the number of queries. An upper bound on the adversary's success probability is equivalent to a lower bound on a scheme's security level.

A common goal of security proofs is to determine the *distinguishing advantage* of an adversary **A**, i.e., the success probability in distinguishing between two schemes $\mathcal{O}$ and $\mathscr{P}$ to which the adversary has oracle access. $\mathcal{O}$ and $\mathscr{P}$ have identical interfaces, and the adversary **A** interacts with one of the two sets of oracles but does not know which one. The adversary, in this setting also called the *distinguisher*, posts its queries to the oracle and eventually outputs a decision bit that represents its guess about which scheme it was interacting with, $\mathcal{O}$ or $\mathscr{P}$. The distinguishing advantage $\Delta_{\mathbf{A}}$ is formally defined as follows.

Definition 3.4 (*Distinguishing advantage*)

Let **A** be an adversary that is given access to two oracles $\mathcal{O}$ and $\mathscr{P}$ that have identical

interfaces. After the adversary's interaction with the oracles, **A** will output a decision bit. The *distinguishing advantage* is given by:

$$\Delta_{\mathbf{A}}(\mathscr{O},\mathscr{P}) = \left|\Pr[\mathbf{A}^{\mathscr{O}} = 1] - \Pr[\mathbf{A}^{\mathscr{P}} = 1]\right|.$$

Probability space and random variables. Randomness is typically sampled before the adversary's interaction with the oracles. The probability space thus consists of the secret key and the random primitives. The deterministic adversary will interact with these primitives through oracles, ask questions, and receive answers. The set of question-answer pairs can be viewed as a (deterministic) function of the underlying probability space, i.e., a random variable.

Total variation distance. The interaction of the adversary with the oracles can be captured in a transcript τ which contains the questions asked by **A** and the answers returned by the oracle. Denote by X the probability distribution induced by the adversary's interaction with $\mathscr{O}$ and denote by Y the probability distribution induced by the adversary's interaction with $\mathscr{P}$. We can show that the adversarial advantage $\Delta_{\mathbf{A}}(\mathscr{O},\mathscr{P})$ is upper bounded by the total variation distance between X and Y:

Theorem 3.3

The distinguishing advantage $\Delta_{\mathbf{A}}(\mathscr{O},\mathscr{P})$ is upper bounded by the total variation distance $||X - Y||_{\mathrm{TV}}$.

Proof. Note that the distinguishing advantage could be defined with the probability of the adversary outputting a 0 instead of 1:

$$\begin{aligned}
\Delta_{\mathbf{A}}(\mathscr{O},\mathscr{P}) &= \left|\Pr[\mathbf{A}^{\mathscr{O}} = 1] - \Pr[\mathbf{A}^{\mathscr{P}} = 1]\right| \\
&= \left|(1 - \Pr[\mathbf{A}^{\mathscr{O}} = 0]) - (1 - \Pr[\mathbf{A}^{\mathscr{P}} = 0])\right| \\
&= \left|\Pr[\mathbf{A}^{\mathscr{O}} = 0] - \Pr[\mathbf{A}^{\mathscr{P}} = 0]\right|.
\end{aligned}$$

Using this fact, for either bit $b \in \{0,1\}$, the following upper bound holds true:

$$\begin{aligned}
\Delta_{\mathbf{A}}(\mathscr{O},\mathscr{P}) &= \left|\Pr[\mathbf{A}^{\mathscr{O}} = b] - \Pr[\mathbf{A}^{\mathscr{P}} = b]\right| \\
&= |\Pr[\mathbf{A}(X) = b] - \Pr[\mathbf{A}(Y) = b]| \\
&= \left| \sum_{\tau \in \mathscr{T}} \Pr[\mathbf{A}(\tau) = b \mid X = \tau] \cdot \Pr[X = \tau] \right. \\
&\qquad \left. - \Pr[\mathbf{A}(\tau) = b \mid Y = \tau] \cdot \Pr[Y = \tau] \right|
\end{aligned}$$

$$= \left| \sum_{\tau \in \mathscr{T}} \Pr[\mathbf{A}(\tau) = b \mid X = \tau] \cdot (\Pr[X = \tau] - \Pr[Y = \tau]) \right|$$
$$\leq \sum_{\tau \in \mathscr{T}} \Pr[\mathbf{A}(\tau) = b \mid X = \tau] \cdot |\Pr[X = \tau] - \Pr[Y = \tau]|$$

The last equality is due to the adversary being indifferent to whether the transcript τ was generated by X or by Y. The upper bound is due to the triangle inequality. Thus, we can state:

$$\Delta_{\mathbf{A}}(\mathscr{O}, \mathscr{P}) \leq \frac{1}{2} \sum_{b \in \{0,1\}} \sum_{\tau \in \mathscr{T}} \Pr[\mathbf{A}(\tau) = b \mid X = \tau] \cdot |\Pr[X = \tau] - \Pr[Y = \tau]|$$
$$= \frac{1}{2} \sum_{\tau \in \mathscr{T}} |\Pr[X = \tau] - \Pr[Y = \tau]|,$$

which concludes the proof. ■

3.5 Pseudorandomness

Cryptographic schemes use as input a small random key and generate from this input a (much) larger output. The output of the cryptographic algorithm is a deterministic function of its input. We want this output to resemble a purely random output as much as possible. If no adversary can distinguish the cryptographic algorithm's output from a purely random output with nonnegligible probability, this cryptographic algorithm is called *pseudorandom*.

Pseudorandom functions and permutations. In this work, we will primarily look at message authentication codes and stream ciphers. MACs produce a pseudorandom *tag* while stream ciphers produce a pseudorandom *keystream*. Ideally both the tag and the keystream resemble random bits as much as possible. We will therefore define the *pseudorandom function (PRF) advantage* of an adversary and use this to measure the security of the scheme we propose in this work.

Let $\mathsf{Func}(\mathscr{X}, \mathscr{Y})$ describe the set of all functions between sets $\mathscr{X}$ and $\mathscr{Y}$ and similarly let $\mathsf{Perm}(\mathscr{X}, \mathscr{Y})$ describe the set of all permutations between sets $\mathscr{X}$ and $\mathscr{Y}$.

Definition 3.5 (*Pseudorandom function*)

Let $\mathscr{K}$, $\mathscr{X}$, and $\mathscr{Y}$ be non-empty sets and let $F : \mathscr{K} \times \mathscr{X} \to \mathscr{Y}$ and $\rho \twoheadleftarrow \mathsf{Func}(\mathscr{X}, \mathscr{Y})$ and $k \twoheadleftarrow \mathscr{K}$. Then, the PRF *advantage* of $\mathbf{A}$ w.r.t. F is defined as $\mathbf{Adv}^{\mathrm{PRF}}_{F}(\mathbf{A}) \stackrel{\mathrm{def}}{=} \Delta_{\mathbf{A}}(F_k, \rho)$, i.e.

$$\Delta_{\mathbf{A}}(F_k, \rho) = \frac{1}{2} \cdot \left(\left| \Pr[\mathbf{A}^{F_k} = 1] - \Pr[\mathbf{A}^{\rho} = 1] \right| \right).$$

A block cipher needs to be bijective, so it is possible to decrypt bits. It therefore resembles a pseudorandom permutation. We will need the *pseudorandom permutation (PRP) advantage* for an example and for our proposed MAC scheme later.

Definition 3.6 (***Pseudorandom permutation***)

Let $\mathcal{K}$ and $\mathcal{X}$ be non-empty sets, $E : \mathcal{K} \times \mathcal{X} \to \mathcal{X}$ be a keyed permutation, and let $\pi \twoheadleftarrow \mathsf{Perm}(\mathcal{X})$ and $k \twoheadleftarrow \mathcal{K}$. Then, the PRP *advantage* of $\mathbf{A}$ w.r.t. E is defined as $\mathbf{Adv}^{\mathrm{PRP}}_{E_k}(\mathbf{A}) \stackrel{\mathrm{def}}{=} \Delta_{\mathbf{A}}(E_k, \pi)$, i.e.

$$\Delta_{\mathbf{A}}(E_k, \pi) = \frac{1}{2} \cdot \left(\left| \Pr[\mathbf{A}^{E_k} = 1] - \Pr[\mathbf{A}^{\pi} = 1] \right| \right).$$

3.6 Distinguishing Attacks

Typically, we want an adversary to not be able to extract any information from a ciphertext, at least not with significant probability. One notion that tries to formalize that is *indistinguishability*. We give a brief example: Consider a sample space $\mathcal{S}$, and for concreteness' sake, $\mathcal{K}$ a key space and $\mathcal{M}$ a message space. Now consider two values $x \in \mathcal{S}$ and $y \in \mathcal{S}$. Let $x \twoheadleftarrow \mathcal{S}$ be sampled uniformly at random from $\mathcal{S}$. As it is drawn at random, it contains no information; there is no underlying plaintext. Let y be generated by some deterministic cryptographic algorithm $\mathtt{Alg}(k, m) = y$ that uses a key $k \twoheadleftarrow \mathcal{K}$ and an additional input $m \in \mathcal{M}$ as inputs, where k is not known by the adversary. There is obviously a relation between y, k, and m. The adversary provides m and is given either x or y and has to decide whether it is the random one or the one generated by $\mathtt{Alg}$. If the adversary cannot decide correctly with significant probability, x and y are said to be indistinguishable.

There are different definitions of security. We chose to restrict ourselves with security against distinguishing attacks. Resistance against distinguishing attacks is the strongest type of security and implies resistance against other attacks.

Indistinguishability from random. The goal is to design a cryptographic algorithm with outputs that are indistinguishable from random noise. In this case, the distinguisher gets access to one of two oracles $\mathcal{O}_{\mathsf{real}}$ or $\mathcal{O}_{\mathsf{ideal}}$ and has to decide which oracle it is interacting with. $\mathcal{O}_{\mathsf{real}}$ corresponds to the cryptographic algorithm to be analyzed, and $\mathcal{O}_{\mathsf{ideal}}$ represents an ideal counterpart. In the case of a block cipher, the ideal counterpart would be a random permutation. In the case of a stream cipher, the ideal counterpart would be a random bitstream. In the case of a message authentication code, the ideal counterpart would be a random function. The distinguishing advantage of a distinguisher $\mathbf{A}$ with access to $\mathcal{O}_{\mathsf{real}}$ or $\mathcal{O}_{\mathsf{ideal}}$ is measured by:

$$\underset{\mathbf{A}}{\Delta}(\mathcal{O}_{\mathsf{real}}, \mathcal{O}_{\mathsf{ideal}}) = \left| \Pr\left[\mathbf{A}^{\mathcal{O}_{\mathsf{real}}} = 1\right] - \Pr\left[\mathbf{A}^{\mathcal{O}_{\mathsf{ideal}}} = 1\right] \right|.$$

Typically, an information-theoretic distinguisher is considered, i.e., its computational resources are unbounded, and the only measure of complexity is the number of queries to the respective oracle. In our work, we will derive an upper bound on the distinguishing advantage and thus a lower bound on the MAC's or stream cipher's security.

3.7 Random Oracle Model

Proofs in the random oracle model were made popular by the work *Random Oracles are Practical* by Bellare and Rogaway [BR93]. The idea behind the random oracle model is to identify primitives, e.g., the round function in AES as a pseudorandom permutation or hash functions in cryptographic protocols as pseudorandom functions, in a cryptosystem and replace these primitives with an idealized version, i.e., a random permutation or a random function. All parties, good and bad alike, get access to the idealized primitives through an oracle. The oracle will receive a query to that primitive, sample an answer uniformly at random, and return the answer.

Heuristic in nature. The cryptosystem is then proved to be secure in this model. Finally, when instantiating the cryptosystem, the random oracles are replaced with real-life primitives. In practice, instantiations of the cryptosystem's primitives are significantly simpler than their random counterparts. Bellare and Rogaway 'stress that the proof is in the random oracle model and the last step is heuristic in nature' [BR93].

Soundness. The soundness of the random oracle model has been debated. A work by Canetti et al. [CGH04] shows that it is possible to design cryptosystems that are secure in the random oracle model but for which no secure instantiation exists. Yet, Canetti et al. [CGH04] also note that:

> [The random oracle model] may be useful as a test-bed (or as a sanity check).
>
> After all, a security proof in the Random Oracle Model eliminates a broad class of potential attacks (i.e., the ones that would work also in the Random Oracle Model), and in many cases it seems that attacks of this type are usually the ones that are easier to find.
>
> For now, however, I view the random oracle methodology as a very useful 'engineering tool' for devising schemes. As a practical matter, I would much rather see today's standards built around schemes that are proven secure in the Random Oracle Model, than around schemes for which no such proofs exist. If nothing else, it makes finding attacks on such schemes a whole lot harder.

Therefore, we see proofs in the random oracle model as a useful groundwork and a solid foundation to base message authentication codes and stream cipher designs upon.

Generic attacks. What is of particular interest is that generic attacks also work in the random oracle model. So, a bound shown in this model will also prove that any generic attack cannot succeed with a probability higher than that in the given bound. This will not imply that any instantiation is secure but rule out a broad class of attacks against a scheme.

4 | The H-coefficient Technique

The H-coefficients technique is a proof technique due to Jacques Patarin [Pat08]. The H-coefficients technique became increasingly popular by the work on the iterated Even-Mansour cipher [CS14, CLL+14, DKS12, LPS12, BKL+12, ABD+13, EM97]. In particular, the work by Chen and Steinberger [CLL+14, CS14] made it accessible to a broader audience. The H-coefficients technique will be used throughout our proofs to calculate upper bounds on the adversary's distinguishing advantage, i.e., lower bounds on security.

Transcripts. The setting is that of an information-theoretic adversary with access to one of two random oracles $\mathcal{O}_{\mathsf{real}}$ and $\mathcal{O}_{\mathsf{ideal}}$. Typically, these correspond to a *real* world and an *ideal* world, where the former corresponds to an abstraction of the construction to be analyzed and the latter corresponds to an idealized construction. The adversary may ask up to q queries to the oracle and, at the end of the interaction, has to decide which oracle it was interacting with. The up to q queries and its answers by the oracle are collected in a transcript τ.

Adversary. The adversary is assumed to be deterministic, and the randomness of the oracle is sampled at the beginning of the experiment. Hence, one can identify the transcripts generated by the adversary's interaction with the first oracle $\mathcal{O}_{\mathsf{real}}$ as the random variable Θ_{real}. Similarly, the transcript produced by the interaction with the second oracle $\mathcal{O}_{\mathsf{ideal}}$ can be identified as the random variable Θ_{ideal}. For a broader introduction to the topic, please refer to [CS14, CLL+14].

Alternatives. Another interesting technique called the χ^2*-Method* was introduced by Dai, Hoang, and Tessaro [DHT17]. The authors used it to improve bounds and simplify proofs that were done using the H-coefficients technique. We did not use it as for our purposes, the H-coefficients technique would suffice.

Overview. In this section, we wish to give an overview of the H-coefficients technique [Pat08]. This overview is based on the popular tutorial by Chen and Steinberger [CLL+14].

Two worlds. The setting is that of an information-theoretic adversary $\mathbf{A}$ that asks q queries to one of two sets of oracles. These two sets of oracles correspond to two 'worlds': the real world and the ideal world. The task of the distinguisher is to determine with which world it is interacting, i.e., to distinguish.

Typically, the oracles correspond to the building blocks of the cryptosystem under consideration. Furthermore, there is an oracle for the construction function that is often the only difference between the two worlds. In the real world, the construction function is composed of the oracles of the underlying building blocks and mimics their interaction in the cryptosystem. In the ideal world, it is independent of the other oracles and corresponds to the primitive we want to show our cryptosystem to be indistinguishable from. In the case of block ciphers, this would be a random permutation; for stream ciphers, a random bitstream; for MACs, a random function.

Distribution of transcripts. The interaction of the distinguisher with the oracles is recorded in a transcript τ. The transcript τ contains all the query-answer pairs of the interaction with the oracle. Without loss of generality, the adversary $\mathbf{A}$ is assumed to be deterministic and makes its final decision as a deterministic function of the transcript obtained.

The probability of obtaining a transcript is different in the two worlds. Let Θ_{real} and Θ_{ideal} denote the distribution of transcripts in the real and the ideal world, respectively. We call a transcript τ *attainable* if $\Pr[\Theta_{\mathsf{ideal}} = \tau] > 0$. Let $\mathscr{T}$ denote the set of all attainable transcripts, and let $\mathscr{T}^+$ denote all the transcripts $\tau \in \mathscr{T}$ where $\Pr[\Theta_{\mathsf{ideal}} = \tau] > \Pr[\Theta_{\mathsf{real}} = \tau]$. $\mathbf{A}$'s distinguishing advantage is upper bounded by the statistical distance (cf. Theorem 3.3):

$$\begin{aligned}
\underset{\mathbf{A}}{\Delta}(\Theta_{\mathsf{ideal}}, \Theta_{\mathsf{real}}) &\leq \frac{1}{2} \sum_{\tau \in \mathscr{T}} |\Pr[\Theta_{\mathsf{ideal}} = \tau] - \Pr[\Theta_{\mathsf{real}} = \tau]| \\
&= \sum_{\tau \in \mathscr{T}^+} (\Pr[\Theta_{\mathsf{ideal}} = \tau] - \Pr[\Theta_{\mathsf{real}} = \tau]) \\
&= \sum_{\tau \in \mathscr{T}^+} \Pr[\Theta_{\mathsf{ideal}} = \tau] \cdot \left(1 - \frac{\Pr[\Theta_{\mathsf{real}} = \tau]}{\Pr[\Theta_{\mathsf{ideal}} = \tau]}\right) \\
&= \sum_{\tau \in \mathscr{T}} \Pr[\Theta_{\mathsf{ideal}} = \tau] \left(1 - \min\left(1, \frac{\Pr[\Theta_{\mathsf{real}} = \tau]}{\Pr[\Theta_{\mathsf{ideal}} = \tau]}\right)\right).
\end{aligned}$$

Good and bad transcripts. Some transcripts are better than others, i.e., the ratio $\Pr[\Theta_{\mathsf{real}} = \tau] / \Pr[\Theta_{\mathsf{ideal}} = \tau]$ may be small (bad) for some transcripts and close to 1 (good) for other transcripts. A typical proof partitions the set of attainable transcripts $\mathscr{T}$ into a set of *good* transcripts GoodT and a set of *bad* transcripts BadT, i.e., $\mathscr{T} = \mathsf{GoodT} \sqcup \mathsf{BadT}$. A central idea of this method is that most transcripts are equally likely in both worlds, i.e., their ratio is close to 1, and the bad ones only occur with negligible probability. We will define values δ and ϵ to upper bound the probability of a bad transcript occurring and to

lower bound the ratio of the good transcripts, respectively. Define $\delta \in [0,1]$ such that:

$$\sum_{\tau \in \mathsf{BadT}} \Pr[\Theta_{\mathsf{ideal}} = \tau] \leq \delta$$

and define $\epsilon \in [0,1]$ such that for all $\tau \in \mathsf{GoodT}$:

$$\frac{\Pr[\Theta_{\mathsf{real}} = \tau]}{\Pr[\Theta_{\mathsf{ideal}} = \tau]} \geq 1 - \epsilon.$$

We then find **A**'s distinguishing advantage to be upper bounded by:

$$\underset{\mathbf{A}}{\Delta}(\Theta_{\mathsf{ideal}}, \Theta_{\mathsf{real}}) \leq \sum_{\tau \in \mathsf{BadT}} \Pr[\Theta_{\mathsf{ideal}} = \tau] + \sum_{\tau \in \mathsf{GoodT}} \Pr[\Theta_{\mathsf{ideal}} = \tau] \cdot \epsilon \leq \delta + \epsilon.$$

Lemma 4.1 summarizes the above and states the fundamental lemma of the H-coefficients technique.

Lemma 4.1 (***H-coefficients technique***)

Assume that the set of attainable transcripts can be partitioned into two disjoint sets GoodT and BadT. Further, assume that there exist $\delta, \epsilon \in [0,1]$ such that for any transcript $\tau \in \mathsf{GoodT}$, it holds that:

$$\Pr[\Theta_{\mathsf{ideal}} \in \mathsf{BadT}] \leq \delta \quad \text{and} \quad \frac{\Pr[\Theta_{\mathsf{real}} = \tau]}{\Pr[\Theta_{\mathsf{ideal}} = \tau]} \geq 1 - \epsilon.$$

Then for all adversaries **A** we have that the distinguishing advantage satisfies

$$\underset{\mathbf{A}}{\Delta}(\Theta_{\mathsf{ideal}}, \Theta_{\mathsf{real}}) \leq \delta + \epsilon.$$

4.1 An Example Using the Even-Mansour Cipher

In this subsection, we will consider the single-round Even-Mansour cipher as introduced in [EM97]. The Even-Mansour cipher is a simple block cipher construction based on a random permutation P. In particular, its iterated variant can be seen as an abstraction of block ciphers like AES and allows us to evaluate the security in a generic setting. Here, $\mathscr{P}_n$ denotes the set of all n-bit permutations, $\mathscr{M}$ denotes the message space, and $\mathscr{C}$ denotes the ciphertext space.

Definition 4.1 (***Even-Mansour cipher***)

Let $k_1, k_2 \twoheadleftarrow \{0,1\}^n$ be two n-bit keys and let $P \twoheadleftarrow \mathscr{P}_n$ be an n-bit random permutation. An encryption of a message block $m \in \mathscr{M}$ using the Even-Mansour construc-

tion is defined as follows:

$$E_{k_1,k_2,P}(m) = P(m \oplus k_1) \oplus k_2.$$

Similarly, the decryption of a ciphertext block $c \in \mathcal{C}$ is defined as follows:

$$E^{-1}_{k_1,k_2,P}(c) = P^{-1}(c \oplus k_2) \oplus k_1.$$

Adversarial capabilities. We will prove that the Even-Mansour construction is indistinguishable from a random permutation for up to $\mathcal{O}(2^{n/2})$ adversarial queries. There will be an oracle for the encryption (construction) function E and for the permutation P. The adversary may query the encryption oracle as well as the permutation oracle in both directions, i.e., it may query E^{-1} and P^{-1} as well. The keys k_1 and k_2 are not known to the adversary. The plaintext-ciphertext pairs (m, c) will be recorded in a construction transcript τ_E. The permutation queries (x, y) will be recorded in a permutation transcript τ_P. The transcripts will not distinguish whether a query was a forward or a backward query. In the ideal world, the encryption oracle will sample its answer according to a second random permutation $P' \twoheadleftarrow \mathcal{P}_n$ that is independent of P and the keys k_1 and k_2, i.e., in the ideal world $E = P'$. Furthermore, in the ideal world, two dummy keys will be sampled uniformly at random. To make matters easier, in either world, we will hand out the keys to the adversary after the interaction with the oracles is finished. When speaking of random primitives, we always refer to the uniform distribution. The full transcript can be written as $(\tau_E, \tau_P, k_1, k_2)$.

Lemma 4.2

The advantage of all adversaries making up to q_E construction queries and up to q_P permutation queries is upper bounded by

$$\underset{\mathrm{A}}{\Delta}(\Theta_{\mathsf{ideal}}, \Theta_{\mathsf{real}}) \leq \frac{2 \cdot q_E \cdot q_P}{2^n}.$$

Proof. Denote the number of construction queries by q_E and the number of permutation queries by q_P. We will first define two bad events that will trivially allow to distinguish the ideal world from the real world.

Bad Events

We consider two bad events in this example:

- bad_1: There exist $(m, c) \in \tau_E$ and $(x, y) \in \tau_P$ such that $m \oplus x = k_1$.
- bad_2: There exist $(m, c) \in \tau_E$ and $(x, y) \in \tau_P$ such that $c \oplus y = k_2$.

The bad events correspond to an adversary asking a construction query and a corresponding permutation query that together are consistent with the respective secret key.

Remark 4.1

Technically, we would have to add to bad_1 the condition 'and $c \oplus y \neq k_2$' and similarly 'and $m \oplus x \neq k_1$' to bad_2. For simplicity we chose not to. This will only marginally affect the bounds.

First, we will bound the bad events in the ideal world.

Lemma 4.3

In the ideal world, we have that:

$$\Pr[\mathsf{bad}_1] = \Pr[\mathsf{bad}_2] \leq \frac{q_E \cdot q_P}{2^n}.$$

Proof of Lemma 4.3. There are $q_E \cdot q_P$ pairs (m, x). The number of distinct $m \oplus x$ is therefore upper bounded by $q_E \cdot q_P$. The key k_1 is sampled uniformly at random from $\{0,1\}^n$ and the collision probability with each $m \oplus x$ is 2^{-n}. The same argument holds true for all pairs (c, y) and k_2. ∎

Good transcripts. Next, we need to lower bound the ratio of the probability of a good transcript occurring in the real world and in the ideal world. We will consider lazy sampling of the adversary's queries. Assume that the keys are sampled first, then the permutation queries are answered, and finally, the construction queries are answered. It is easy to see that the two worlds are indistinguishable without considering construction queries, as the keys and permutation queries are sampled identically in either world.

The construction queries differ. In the ideal world, construction queries are answered using a second random permutation P'. Note that before answering construction queries, no values of P' have been sampled, but q_P values of P have already been sampled. Also, note that for all (m, x) we have $m \oplus x \neq k_1$ and for all (c, y) we have $c \oplus y \neq k_2$. This has no effect in the ideal world but means in the real world that no $m \oplus k_1$ and no $c \oplus k_2$ occurs as input or output in a P-query, as these have been excluded through the bad transcripts. That means, all construction queries that evaluate P in the real world sample 'fresh', i.e., previously unqueries, values. We can see that for all $\tau \in \mathsf{GoodT}$:

$$\Pr_{\text{real}}[\forall (m,c) \in \tau_E : E(m) = c] = \frac{1}{2^n - q_P} \cdot \frac{1}{2^n - q_P - 1} \cdots \frac{1}{2^n - q_P - (q_E - 1)},$$

$$\Pr_{\text{ideal}}\left[\forall(m,c)\in\tau_E : P'(m)=c\right]=\frac{1}{2^n}\cdot\frac{1}{2^n-1}\cdots\frac{1}{2^n-(q_E-1)}.$$

In particular, we can see that the probability of obtaining a good transcript is higher in the real world, i.e., $\epsilon = 0$. The claim follows. ■

4.2 Mirror Theory

Mennink and Neves [MN17] used the Mirror Theory to provide simpler proofs for the sum of permutations and improve the bound of EWCDM [CS16] that inspired us to prove the security of a MAC that is based upon the sum of permutations.

Related work. The Mirror Theory was originally described by Patarin [Pat10, Pat17] and made more accessible by Mennink and Neves [MN17]. It gives a lower bound to the number of solutions to a given system of equations of the type $P_{a_i} \oplus P_{b_i} = \lambda_i$.

For message authentication codes, an adversary is typically allowed to ask for *verification queries*. With regard to the system of equations, these types of queries correspond to non-equations of the type $P_{a_i} \oplus P_{b_i} \neq \lambda_i$. Hence, [DDNY18] extended the Mirror Theory by these non-equations.

There has been some debate over the correctness of the Mirror Theory, in particular whether it holds true for proofs that show security greater than $\mathcal{O}(2^{2n/3})$ bits, as gaps were identified in the proof of Patarin. [DDD21] showed the extended Mirror Theory to hold true for up to $\mathcal{O}(2^{3n/4})$ adversarial queries.

More recently, [CP20] and [DNS22] showed that the Mirror Theory holds true for up to $\mathcal{O}(2^n)$ queries if the maximal block size (as will be defined later) is smaller or equal to two. Finally, [CDN+23] showed that the Mirror Theory holds true for up to $\mathcal{O}(2^n)$ queries if the maximal block size is within the order of $\mathcal{O}(2^{n/4}/\sqrt{n})$.

H-coefficient technique. We will combine the H-coefficient technique with Patarin's Mirror Theory, which allows us to lower bound the amount of good transcripts. The Mirror Theory will facilitate calculating the probability of obtaining a good transcript in the real world. In the following, we briefly recall the necessary definitions according to the Mirror Theory, as presented in [MN17], which follows Patarin [Pat10, Pat17].

Mirror Theory. The Mirror Theory evaluates the number of possible solutions to a system of affine equations of the form $P_{a_i} \oplus P_{b_i} = \lambda_i$ in a finite group. Let $q \geq 1$ denote the number of equations and $r \geq 1$ denote the number of unknowns. Let $\mathscr{P} = \{P_1, \ldots, P_r\}$ represent the set of r distinct unknowns, and consider an equation system

$$\mathscr{E} = \left\{P_{a_1} \oplus P_{b_1} = \lambda_1, \ldots, P_{a_q} \oplus P_{b_q} = \lambda_q\right\},$$

where a_i, b_i for $1 \leq i \leq q$ are mapped to $\{1, \ldots, r\}$ by a surjective index mapping $\varphi : \{a_1, b_1, \ldots, a_q, b_q\} \to \{1, \ldots, r\}$. Given a subset of equations $\mathscr{I} \subseteq \{1, \ldots, q\}$, the multiset $\mathscr{M}_{\mathscr{I}}$ is defined as $\mathscr{M}_{\mathscr{I}} = \bigcup_{i \in \mathscr{I}} \{\varphi(a_i), \varphi(b_i)\}$.

Definition 4.2 (***Circle-freeness***)

An equation system $\mathscr{E}$ is circle-free if there exists no subset of indices $\mathscr{I} \subseteq \{1, \ldots, q\}$ of equations s.t. $\mathscr{M}_{\mathscr{I}}$ has even multiplicity elements only.

So, no linear combination of equations is independent of the unknowns.

Definition 4.3 (***Block-maximality***)

Let $\mathscr{Q}_1, \ldots, \mathscr{Q}_s = \{1, \ldots, r\}$ be a partitioning of the r indices into s minimal so-called blocks s.t. for all equation indices $i \in \{1, \ldots, q\}$, there exists a single block index $\ell \in \{1, \ldots, s\}$ s.t. the unknowns of the i-th equation are contained in only this block: $\{\varphi(a_i), \varphi(b_i)\} \subseteq \mathscr{Q}_\ell$. Then, the system of equations $\mathscr{E}$ is called ξ-block-maximal for $\xi \geq 2$ if there exists no $i \in \{1, \ldots, s\}$ s.t. $|\mathscr{Q}_i| > \xi$.

So, the unknowns can be partitioned into blocks of size at most $\xi + 1$ if $\mathscr{E}$ is ξ-block-maximal.

Definition 4.4 (***Non-degeneracy***)

A system of equations $\mathscr{E}$ is non-degenerate iff there is no $\mathscr{I} \subseteq \{1, \ldots, q\}$ s.t. $\mathscr{M}_{\mathscr{I}}$ has exactly two odd multiplicity elements and $\bigoplus_{i \in \mathscr{I}} \lambda_i = 0$.

So, an equation system is non-degenerate if there is no linear combination of one or more equations that imply $P_i = P_j$ for distinct i, j and $P_i, P_j \in \mathscr{P}$. The central theorem of Patarin's mirror theorem is then Theorem 2 in [MN17], which itself is a brief form of Theorem 6 in [Pat10].

Theorem 4.1 (***Mirror Theorem [MN17]***)

Let $\xi \geq 2$. Let $\mathscr{E}$ be a system of equations over the unknowns $\mathscr{P}$ that is (i) circle-free, (ii) ξ-block-maximal, and (iii) non-degenerate. Then, as long as $(\xi - 1)^2 \cdot r \leq 2^n/67$, the number of solutions s.t. $P_i \neq P_j$ for all pairwise distinct $i, j \in \{1, \ldots, r\}$ is at least

$$\frac{(2^n)_r}{(2^n)^q}.$$

A proof sketch is given in [MN17, Appendix A], and the details in [Pat10]. An updated proof had been given in [NPV17].

Relaxed Mirror Theory. Mennink and Neves [MN17] describe a relaxation wherein the condition that two unknowns P_a and P_b must differ whenever a and b differ is released

to the degree that distinct unknowns must be pairwise distinct only inside their blocks. So, it must hold for $a \neq b$ that $P_a \neq P_b$ when $a, b \in \mathscr{R}_j$ for some $j \in \{1, \ldots, s\}$ for a given partitioning $\{1, \ldots, r\} = \bigcup_{i=1}^{s} \mathscr{R}_i$.

Definition 4.5 (*Relaxed Non-degeneracy*)

An equation system $\mathscr{E}$ is relaxed non-degenerate with respect to the partitioning $\{1, \ldots, r\} = \bigcup_{i=1}^{s} \mathscr{R}_i$ iff there is no $\mathscr{I} \subseteq \{1, \ldots, q\}$ s.t. $\mathscr{M}_{\mathscr{I}}$ has exactly two odd multiplicity elements and $\bigoplus_{i \in \mathscr{I}} \lambda_i = 0$.

In [MN17, Theorem 3], Mennink and Neves extend Theorem 4.1 to the following relaxed form:

Theorem 4.2 (*Relaxed Mirror Theorem [MN17]*)

Let $\xi \geq 2$ and let $\{1, \ldots, r\} = \bigcup_{i=1}^{s} \mathscr{R}_i$ be a partition of the r indices. Let $\mathscr{E}$ be a system of equations over the unknowns $\mathscr{P}$ that is (i) circle-free, (ii) ξ-block-maximal, and (iii) non-degenerate w.r.t. the partition $\{1, \ldots, r\}$. Then, as long as $(\xi - 1)^2 \cdot r \leq 2^n/67$, the number of solutions s.t. $P_i \neq P_j$ for all pairwise distinct $i, j \in \{1, \ldots, r\}$ is at least

$$\frac{\mathsf{NonEq}(\mathscr{R}_1, \ldots, \mathscr{R}_s; \mathscr{E})}{(2^n)^q},$$

where $\mathsf{NonEq}(\mathscr{R}_1, \ldots, \mathscr{R}_s; \mathscr{E})$ is the number of solutions to $\mathscr{P}$ that satisfy $P_a \neq P_b$ for all $a, b \in \mathscr{R}_j$ for all $1 \leq j \leq s$ as well as all inequalities (the equalities released) by $\mathscr{E}$.

Mennink and Neves stress that the relaxed Theorem 4.2 is equivalent to Theorem 4.1 for $s = 1$, i.e., when the equation system consists of a single block. Moreover, the number of solutions that are covered in the term $\mathsf{NonEq}(\mathscr{R}_1, \ldots, \mathscr{R}_s; \mathscr{E})$ can be lower bounded by $(2^n)_{|\mathscr{R}_1|} \cdot \prod_{i=2}^{s} (2^n - (\xi - 1))_{|\mathscr{R}_i|}$ since every variable is in exactly one block which imposes at most $\xi - 1$ additional inequalities to the other unknowns in its block.

4.3 An Example Using the Sum of Permutations

The MAC constructions we propose later in this work are based on the sum of permutations. We therefore see it as a useful example to present to proof of security using the Mirror Theory for the sum of permutations.

Definition. In this section, we will present an example application of the Mirror Theory showing that the sum of permutations is a secure pseudorandom function. We will follow the example by [MN17], based on [Pat10], closely.

Definition 4.6 (***Sum of permutations***)

Let $\pi_1, \pi_2 \twoheadleftarrow \mathscr{P}_n$ be n-bit random permutations and let $x \in \{0,1\}^n$ be a n-bit input. The sum of permutations is defined as follows:

$$F_{\pi_1,\pi_2}(x) = \pi_1(x) \oplus \pi_2(x).$$

Adversarial capabilities. We will show that the sum of permutations is indistinguishable from a random function with n-bit inputs and n-bit outputs for up to $\mathcal{O}(2^n)$ adversarial queries. The adversary may only query the construction oracle, i.e., the function F_{π_1,π_2}, in the forward direction. Queries to the underlying random permutations are not permitted. The input-output pairs (x, y) will be recorded in a transcript τ. In the ideal world, the construction oracle will sample its answer according to a random function $F' \twoheadleftarrow \mathscr{F}_n$ that is independent of π_1 and π_2.

Lemma 4.4

The advantage of all adversaries making up to $q \leq 2^n/67$ queries to the sum of permutations is upper bounded by:

$$\underset{\mathsf{A}}{\Delta}(\Theta_{\mathsf{ideal}}, \Theta_{\mathsf{real}}) \leq \frac{q}{2^n}.$$

Proof. Following [MN17], we will define for every input-output pair $(x_i, y_i) \in \tau$

$$P_{2i-1} := \pi_1(x_i) \text{ and } P_{2i} := \pi_2(x_i).$$

The transcript then defines the system of q equations in $2q$ unknowns $\mathscr{E}$:

$$\begin{aligned} P_1 \oplus P_2 &= y_1, \\ P_3 \oplus P_4 &= y_2, \\ &\vdots \\ P_{2q-1} \oplus P_{2q} &= y_q. \end{aligned}$$

Note that all unknowns are distinct as we use two independent permutations. The indices $\{1, \ldots, 2q\}$ can be partitioned into the indices corresponding to π_1, $\mathscr{R}_1 = \{1, 3, \ldots, 2q-1\}$, and the indices corresponding to π_2, $\mathscr{R}_2 = \{2, 4, \ldots, 2q\}$.

We do not need to consider bad transcripts in this example. We only need to compute

the ratio for all transcripts $\tau \in \mathscr{T}$:

$$\frac{\Pr[\Theta_{\mathsf{real}} = \tau]}{\Pr[\Theta_{\mathsf{ideal}} = \tau]} \geq 1 - \epsilon.$$

The probability to obtain τ in the ideal world $\Pr[\Theta_{\mathsf{ideal}} = \tau]$ is easily upper bounded by 2^{-nq}. In the real world, we will apply the Relaxed Mirror Theorem. Note that $\mathscr{E}$ is circle-free, has q-blocks of size 2 and is relaxed non-degenerate with respect to partition $\{1, \ldots, r\} = \mathscr{R}_1 \sqcup \mathscr{R}_2$. The number of solutions to the unknowns P_1 to P_q is at least $\frac{\mathsf{NonEq}(\mathscr{R}_1, \mathscr{R}_2; \mathscr{E})}{2^{nq}}$.

We need to lower bound the term $\mathsf{NonEq}(\mathscr{R}_1, \mathscr{R}_2; \mathscr{E})$. There are 2^n choices for P_1, $2^n - 1$ choices for P_3 (as $P_3 \neq P_1$), etc. Thus, for $P_1, P_3, \ldots, P_{2q-1}$ there are $(2^n)_q$ choices. There are $2^n - 1$ choices for P_2 (if $y_1 \neq 0$ then P_2 should be unequal to P_1), $2^n - 2$ choices for P_4 (as $P_4 \neq P_2$ and if $y_2 \neq 0$ it should be unequal to P_3). Thus, for $P_2, P_4, \ldots, P_{2q}$ there are $(2^n - 1)_q$ choices. We obtain

$$\mathsf{NonEq}(\mathscr{R}_1, \mathscr{R}_2; \mathscr{E}) \geq (2^n)_q (2^n - 1)_q.$$

For the remaining output values of π_1 and π_2 there are $(2^n - q)!$ choices respectively, thus

$$\Pr[\Theta_{\mathsf{real}} = \tau] \geq \frac{\frac{(2^n)_q (2^n - 1)_q}{2^{nq}} \cdot ((2^n - q)!)^2}{(2^n!)^2} = \frac{1}{2^{nq}} \left(1 - \frac{q}{2^n}\right).$$

Hence, for the ratio we obtain

$$\frac{\Pr[\Theta_{\mathsf{real}} = \tau]}{\Pr[\Theta_{\mathsf{ideal}} = \tau]} \geq 1 - \frac{q}{2^n}.$$

and therefore $\epsilon = q/2^n$ provided $q \leq 2^n/67$. $\delta = 0$ as we do not consider bad transcripts. The claim follows. ■

Part II

Message Authentication Codes

5 | Message Authentication Codes

In this chapter, we wish to present the fundamentals of *message authentication codes* (MACs). We explain their purpose, what distinguishes them from hash functions and digital signatures, and explain their typical vulnerabilities. There are stateless schemes and stateful schemes that use a nonce as an additional parameter. One important theme in MAC research is overcoming the birthday bound. Another theme is the repetition of nonces in stateful schemes. The Wegman-Carter scheme, which we will introduce later, breaks with a single nonce repetition. Therefore, research tries to identify schemes that do not break with a single nonce repetition and potentially still allow for beyond the birthday bound secure schemes with multiple nonce repetition. Furthermore, schemes should be as efficient as possible.

Message authentication. Block ciphers and stream ciphers typically do not ensure that the message arrives unmodified at the receiver. Ciphertexts may be modified in transit, and while the decryption of a modified ciphertext block will most likely be garbage, for stream ciphers, select plaintext bits can easily be flipped by flipping the corresponding ciphertext bits. To prevent a message from being modified in transit and preserve the message's integrity, message authentication codes are used. MACs aim to guarantee the authenticity and integrity of submitted messages. These typically compute a tag from the message. That tag is sent along with the message. So, a receiver can successfully determine with high probability whether a given pair (m, t) of message and tag has been generated by the legitimate sender and has been transmitted correctly or not. The receiver then computes the tag for the message received and compares it with the tag received. An adversary should not be able to compute the tag with significant probability unless the MAC's secret key is known. A tag that is computed by an illegitimate party is called a *forgery*.

Network protocols. Message authentication codes are typically used in network transfers to verify the integrity and authenticity of messages. Popular protocols that make use of MACs are Transport Layer Security (TLS), known for securing HTTPS traffic over the internet, Secure Shell (SSH) for remotely accessing computers over a network, and the VPN protocols IPsec and Wireguard.

Popular algorithms. Some of the most popular MAC algorithms are the following:

- **HMAC** [BCK96a, BCK96b] is used in IPsec [KA98a, MG98, KA98b], Secure Shell (SSH) [YL06, BB12], Transport Layer Security (TLS) [DR08, Res18], and OAuth 1.0 [HL10].
- **Poly1305** [Ber05c] is used in libsodium [lib23], NaCl [NaC23], Transport Layer Security (TLS) [NLR18], and Wireguard [Wir23].
- **SipHash** [AB12] is used in libsodium [lib23] and Wireguard [Wir23].

Types of MACs. There are various types of Message Authentication Codes (MACs), including stateless deterministic, randomized, and stateful constructions. Additionally, nonce-based constructions are distinguished, wherein the sender is tasked with providing a unique nonce for each message to be authenticated. Considering the potential cost of obtaining cryptographically secure randomness in different scenarios, our emphasis lies on stateless and nonce-based constructions.

MACs vs. digital signatures. A hash is an unkeyed operation used to verify the integrity of a message. It provides protection against messages inadvertently being modified in transit, but it does not protect against a malicious attacker, as the attacker can also compute the hash.

The fundamental difference between a MAC and a digital signature is that MACs are symmetric schemes, i.e., the sender and receiver share the same key, while digital signatures are asymmetric schemes, i.e., the sender and receiver have different keys: one for signing and one for verifying a message.

In addition to integrity, both MACs and digital signatures verify a message's authenticity. However, unlike a hash function, a malicious attacker cannot compute the tag or the digital signature without knowing the respective secret key.

Due to digital signatures being an asymmetric scheme, they further provide non-repudiation, i.e., a sender cannot deny having sent a message. This is due to the fact that for digital signatures, the secret key is only known to the sender, whereas for a MAC, the secret key is shared between the legitimate parties, and thus both parties can generate a valid tag.

Asymmetric schemes tend to be slower than symmetric schemes. Consequently, if non-repudiation is not necessary, a MAC (Message Authentication Code) is the preferable choice.

Challenges. We will later introduce the Wegman-Carter MAC, a MAC that combines a pseudorandom function with a universal hash function. Universal hash functions generally do not typically fulfill cryptographic requirements, making them more lightweight and faster to compute. One downside is that this makes finding a preimage or identifying the hashing key much easier for an adversary.

The Wegman-Carter MAC uses a nonce as a second input. It enjoys a high-security level if nonces are never repeated. If nonces are repeated, this can lead to a recovery of the hashing key [Jou06, HP08]. Therefore, one goal to be accomplished is to allow for nonce repetitions for Wegman-Carter-style MACs without catastrophic consequences. This would also allow the use of random nonces, and thus, keeping a state would not be required.

Moreover, it is worth noting that pseudorandom functions are not as readily accessible as pseudorandom permutations or block ciphers, respectively. Replacing the pseudorandom function in the Wegman-Carter MAC with a pseudorandom permutation causes security to drop back to the birthday bound [Sho96]. It is, therefore, desirable to design a Wegman-Carter-style scheme that uses a pseudorandom permutation with as few primitive calls as possible. With higher data rates for high-powered devices and more and more small electronic devices sending sensitive information, efficiency becomes more important for MACs as well. There exist some more generic attacks.

Generic attacks. Some MACs are vulnerable to length extension attacks. In this type of attack, an attacker will extend a previously intercepted message and can create a valid tag without knowing the secret key. In the case of Wegman-Carter MACs and the universal hash function not fulfilling cryptographic requirements, the observation of nontrivial properties of the hash value can lead to a recovery of the hashing key and thus forgeries [Jou06, HP08]. A replay attack involves sending a previously sent message and its tag again. The tag is obviously valid. The replay attack as such is not preventable but is typically handled by the communication protocol using message counters and thus identifying duplicate messages.

Definition. Usually, people assume the goal of cryptography to be secrecy, i.e., a third party eavesdropping on a communication channel not being able to decrypt the contents. However, we also want to ensure that the contents received come from the legitimate sender and were not modified in transit. An adversary may have acquired some information about a message that is sent encrypted using a stream cipher. The adversary could then flip the appropriate bits to change the contents of the message. Message authentication codes aim to preserve the message integrity. Formally, a message authentication code is defined as follows [KL20]:

Definition 5.1

A message authentication code (MAC) consists of two probabilistic polynomial-time algorithms $(\mathsf{Tag}, \mathsf{Ver})$ such that:

- Tag receives as input the secret key k and a variable-length message $m \in \{0,1\}^*$. Its output is a tag $t \in \mathcal{T}$, where $\mathcal{T}$ denotes the tagspace.
- Ver receives as input the secret key k, the variable-length message $m \in \{0,1\}^*$ and the tag $t \in \mathcal{T}$. Its output is a bit $b \in \{0,1\}$. b is 1 if t is a valid tag for m

and 0 otherwise.

Further, it is required that $\mathsf{Ver}(k, m, \mathsf{Tag}(k, m)) = 1$.

5.1 Security Requirements

A MAC is supposed to ensure that the message was sent by the legitimate sender and was not modified in transit. The tag is used to verify the authenticity and integrity of the message. As the tag is used to verify a message, no adversary should be able to create a valid tag for a yet unknown message. Creating a valid tag for an unknown message not created by a legitimate party is called a *forgery*. We wish to achieve *unforgeability* for MACs. That is, no valid tag can be computed by an illegitimate party without knowledge of the secret key.

The two primary targets of a message authentication code are to ensure a message's integrity and the sender's authenticity. It is, therefore, required that no forgery, as defined below, be possible with significant probability. Often, a stronger definition of security is used: indistinguishability from a random function. Showing that Tag is a pseudorandom function will imply the hardness of forgeries. In our proofs, the attacker will get access to a tagging oracle and a verification oracle and can thus generate tags for chosen messages and verify tags as desired.

Unforgeability. The security of a message authentication code is defined via forgeries. A forgery is a *new* tag t generated by the adversary for a message m that has not been communicated before by the legitimate parties. It is required that no adversary can generate a valid tag for a new message with significant probability. We will first define security for a stateless MAC scheme and later present the definition of security for a nonce-based (stateful) scheme.

Definition 5.2 (*MAC-advantage*)

Let $(\mathsf{Tag}, \mathsf{Ver})$ be a message authentication code as defined in Definition 5.1. Let $\mathsf{Tag}_k(m)$ be an oracle for the tagging algorithm $\mathsf{Tag}(k, m)$ and let $\mathsf{Ver}_k(m, t)$ be an oracle for the verification algorithm $\mathsf{Ver}(k, m, t)$. Consider an adversary $\mathbf{A}$ with oracle access to $\mathsf{Tag}_k(m)$ and $\mathsf{Ver}_k(m, t)$ that is allowed to make at most q_m queries to the tagging oracle (MAC queries) and at most q_v queries to the verification oracle (verification queries). We say that $\mathbf{A}$ forges if any of its verification queries returns one. The advantage of $\mathbf{A}$ against the MAC security of $(\mathsf{Tag}, \mathsf{Ver})$ is defined as

$$\mathbf{Adv}^{\mathrm{MAC}}_{(\mathsf{Tag},\mathsf{Ver})}(\mathbf{A}) = \Pr\left[k \twoheadleftarrow \mathscr{K} \ : \ A^{\mathsf{Tag}_k, \mathsf{Ver}_k} \text{ forges}\right].$$

The adversary is not allowed to ask a verification query (m, t) if a previous query to

$\mathsf{Tag}_k(m)$ returned t.

Remark 5.1

We do not consider probabilistic adversaries here. Also we consider information-theoretic security, hence the adversary is unbounded in time.

Nonce-based MACs. In the previous section, we defined a message authentication code without including nonces. In this work, we will prove the security of two constructions: one is nonce-based and therefore stateful, and the other is stateless. The main difference to Definition 5.1 is that Tag and Ver now both take an additional input $\nu \in \mathcal{N}$.

Definition 5.3 (*Nonce-based MAC*)

Let $\mathcal{K}$ be the keyspace, $\mathcal{N}$ be the noncespace and $\mathcal{T}$ be the tagspace. A message authentication code (MAC) consists of two probabilistic polynomial-time algorithms $(\mathsf{Tag}, \mathsf{Ver})$ such that:

- Tag receives as input the secret key k, a nonce ν, and a variable-length message $m \in \{0,1\}^*$. Its output is a tag $t \in \mathcal{T}$.
- Ver receives as input the secret key k, the nonce ν, the variable-length message $m \in \{0,1\}^*$, and the tag $t \in \mathcal{T}$. Its output is a bit $b \in \{0,1\}$. b is 1 if t is a valid tag for m and 0 otherwise.

Further it is required that $\mathsf{Ver}(k, \nu, m, \mathsf{Tag}(k, \nu, m)) = 1$.

Accordingly, we wish to update the definition of the MAC-advantage for a nonce-based message authentication code. Also, based on this definition, we introduce what it means that an adversary is *nonce-respecting*.

Definition 5.4 (*MAC-advantage*)

Let $(\mathsf{Tag}, \mathsf{Ver})$ be a nonce-based message authentication code as defined in Definition 5.3. Let $\mathsf{Tag}_k(\nu, m)$ be an oracle for the tagging algorithm $\mathsf{Tag}(k, \nu, m)$ and let $\mathsf{Ver}_k(\nu, m, t)$ be an oracle for the verification algorithm $\mathsf{Ver}(k, \nu, m, t)$. Consider an adversary $\mathbf{A}$ with oracle access to $\mathsf{Tag}_k(\nu, m)$ and $\mathsf{Ver}_k(\nu, m, t)$ that is allowed to make at most q_m queries to the tagging oracle (MAC queries) and at most q_v queries to the verification oracle (verification queries). We say that $\mathbf{A}$ forges if any of its verification queries returns 1. The advantage of $\mathbf{A}$ against the MAC security of $(\mathsf{Tag}, \mathsf{Ver})$ is defined as

$$\mathbf{Adv}^{\mathrm{MAC}}_{(\mathsf{Tag},\mathsf{Ver})}(\mathbf{A}) = \Pr\left[k \twoheadleftarrow \mathcal{K} \ : \ A^{\mathsf{Tag}_k, \mathsf{Ver}_k} \text{ forges}\right].$$

The adversary is not allowed to ask a verification query (ν, m, t) if a previous query to $\mathsf{Tag}_k(\nu, m)$ returned t. The adversary is said to be *nonce-respecting* if it never repeats a nonce $\nu \in \mathcal{N}$ in its queries to Tag_k.

While the primary goal of a MAC is unforgeability, indistinguishability from random bits can be a valuable replacement goal to evaluate the security. If tags are indistinguishable from random, they are also hard to forge.

PRF security. PRF security implies that the outputs of the tag generation algorithm Tag cannot be distinguished from a random bitstream of the same length. This is the stronger notion of security for message authentication codes and implies the hardness of forgeries.

Lemma 5.1

If an algorithm Tag is a secure PRF, then it also is a secure MAC. A secure MAC need not be a secure PRF.

Proof. Assume an algorithm Tag is a secure PRF but not a secure MAC and let Ver be the corresponding verification algorithm. In particular, assume that the MAC-advantage is lower then the PRF-advantage: $\mathbf{Adv}^{\mathrm{MAC}}_{(\mathsf{Tag},\mathsf{Ver})}(\mathbf{A}) < \mathbf{Adv}^{\mathrm{PRF}}_{\mathsf{Tag}}(\mathbf{A})$. Then, the adversary can produce a valid pair (m, t) on a previously unqueried message m. That is, the adversary can predict a value t of the pseudorandom function $\mathsf{Tag}(m)$. This is a contradiction to the PRF requirement of Tag.

PRF security is not necessary for a secure MAC though. Let f be a secure PRF. Define the tagging algorithm $\mathsf{Tag} := f(k, m) \,||\, 0$. Tag is a secure MAC but as it always ends in 0 it is easy to distinguish from a truly random function. ■

In the proofs later presented in this part of this thesis, we will show PRF security for the proposed MAC scheme.

Replay attacks. MACs are vulnerable to *replay attacks*. In this type of attack, an adversary uses an observed message-tag pair (m, t) and sends (m, t) to the respective receiver. Usually, a message counter is included in the message m to make these kinds of attacks detectable. The replayed message m is then discarded. Due to the correctness of the scheme, replay attacks are not preventable by the MAC algorithm. We will not consider replay attacks any further.

5.2 Wegman-Carter MACs

The Wegman-Carter approach [WC81] is a popular and efficient paradigm for constructing secure MACs. There, a given message is first compressed with a universal hash function

before the result is processed by a cryptographically secure random function.

Original description. Let $h \in \mathcal{H}$ be the hash function used. The original scheme generates, for every message m_i, a random bitstring b_i where $|h(m_i)| = |b_i|$ and the tag is computed via $t = h(m_i) \oplus b_i$. The secret key is the sequence $(h, b_1, b_2, \ldots)$ which has to be communicated between the legitimate parties over a secure channel.

Recent description. In more recent descriptions of the Wegman-Carter scheme, a pseudorandom function f is used instead of the random bitstrings. The secret key then becomes the tuple (h, f). Alternative descriptions use a keyed hash function with key k_h and a keyed function with key k_f. For every message m_i, a unique nonce ν_i is generated and sent with the tag t_i to the receiver. The tag is computed via $t_i = h(m_i) \oplus f(\nu_i)$.

Real world MACs. Some message authentication codes that use the Wegman-Carter construction are VMAC [Kro06], UMAC [BHK+99], and Poly1305-AES [Ber05c].

5.2.1 Universal Hashing

Universal, almost-universal, and almost-XOR-universal hash functions only make a statement about the collision probability of two values and are, in particular, not required to fulfill cryptographic requirements. Let $\mathcal{X}$ and $\mathcal{Y}$ denote two non-empty sets, and $\mathcal{H} = \{h : \mathcal{X} \to \mathcal{Y}\}$ be a family of hash functions h.

Definition 5.5 (*Almost-Universal Hash Function [CW79]*)

We say that $\mathcal{H}$ is ε-almost-universal (ε-AU) if, for all distinct $x, x' \in \mathcal{X}$, it holds that $\Pr_{h \leftarrow \mathcal{H}}[h(x) = h(x')] \leq \varepsilon$.

If the outputs of the hash functions $h \in \mathcal{H}$ have length n and the collision probability is 2^{-n}, then $\mathcal{H}$ is called universal. Almost-universality is a more relaxed definition. Almost-XOR-universal hash functions were introduced in [Kra94]; however, the term is due to Rogaway [Rog95].

Definition 5.6 (*Almost-XOR-Universal Hash Function [Kra94, Rog95]*)

Here, let $\mathcal{Y} \subseteq \{0,1\}^n$ for some positive integer n. We say that $\mathcal{H}$ is ε-almost-XOR-universal (ε-AXU) if, for all distinct $x, x' \in \mathcal{X}$ and arbitrary $y \in \mathcal{Y}$, it holds that $\Pr_{h \leftarrow \mathcal{H}}[h(x) \oplus h(x') = y] \leq \varepsilon$.

Polynomial hashing. Many almost-XOR-universal hash functions have been proposed in the literature, most of them based on polynomial hashing [Kra94, Rog95, Sho96, HK97, BHK+99, KR00, KVW04, MV04, Ber05c]. One popular example of a polynomial hash function is PolyHash [MI11]. However, in this work, it should be noted that PolyHash

is solely used as an illustrative example of a polynomial hash function, chosen for its simplicity and ease of comprehension. We do not elaborate any further on polynomial hashing but want to give the reader a basic idea. It needs to be highlighted that polynomial hash function families need not fulfill cryptographic requirements and, in particular, need not be *cryptographic* hash functions.

Definition 5.7 (PolyHash)

Let $k \in \{0,1\}^n$ be the n-bit hashing key and let $m \in \{0,1\}^*$ be a message. m is first padded with an injective padding of the form 01^* so its length is a multiple of n. Let $m^* = m_1||m_2||\ldots||m_\ell$ be the padded message, where $|m_i| = n$ for each $i \in \{1,\ldots,\ell\}$, and ℓ is the number of n-bit blocks of the padded message. Then, PolyHash is defined as follows:

$$\text{PolyHash}_k(m) = m_\ell k \oplus m_{\ell-1}k^2 \oplus \cdots \oplus m_1 k^\ell.$$

PolyHash is a $\ell/2^n$-XOR-universal hash function [DDNY18].

5.2.2 Attacks on Wegman-Carter MACs

Wegman-Carter MACs enjoy high security properties if nonces are never repeated. Assuming that f is a perfect random function, an adversary posting q_m MAC queries and q_v verification queries succeeds with a probability of $\varepsilon \cdot q_v$ [CS16]. A single nonce repetition, however, can lead to a hashing key recovery and will thus allow universal forgeries [Jou06, HP08].

Repeating nonces. Assume that an adversary obtains two message-tag-nonce triples (m,t,ν) and (m',t',ν) with a repeated nonce. The adversary then computes $t \oplus t' = h(m) \oplus f(\nu) \oplus h(m') \oplus f(\nu) = h(m) \oplus h(m')$, i.e., with $h(m) \oplus h(m') \oplus t \oplus t' = 0$ the adversary learns that the hashing key is the root of a known polynomial. As [Jou06] notes:

> One important problem for the adversary is that the degree of this polynomial can be high, since it is equal to the length in blocks of the longer of the two messages M(1) and M(2). Since the number of roots can potentially be as high as the degree, the adversary may hesitate between a large number of possible H. However, on average, we only expect a small number of roots. Moreover, if the adversary can obtain a second pair of messages with common IVs, he gets a second polynomial with root H. Then by computing the GCD of the two polynomials, he finds a polynomial of small degree with H as a root. Finally, he is left with a small number of candidates for H. If there is a single candidate, he is done and stops there. If there are a few, he can either collect additional

> pairs with common IVs or use a chosen ciphertext attack [...] to test each possible value.

Known hashing key. Assume that an adversary learns the secret hashing key after observing nonce-repetitions as shown in [Jou06, HP08]. This will allow the adversary to compute $h(m)$ for any message m. Note that for verification queries the adversary is allowed to repeat nonces. For an observed message-nonce-tag triple (m, ν, t), the adversary can compute $f(\nu) = t \oplus h(m)$. For a new message m', the adversary can simply compute a forgery via $t' = h(m') \oplus f(\nu)$ and obtains a valid message-nonce-tag triple (m', ν, t').

Random permutation. Pseudorandom functions often are not available, yet pseudorandom permutations are readily available as block ciphers. Therefore, one may ask if f can be instantiated as a (pseudo-)random permutation. This will cause the security of the Wegman-Carter scheme to drop to the birthday bound.

One can easily obtain a distinguisher with birthday bound complexity: For $2^{n/2}$ different nonces ν_i and a single message m, the adversary will ask for the tag t_i of the pair (m, ν_i). As f is a random permutation, all values of $f(\nu_i)$ are unique and no collision in the tags will be observed. If f were a random function, one would expect a collision after $2^{n/2}$ queries with high probability.

Stateless MAC. Getting rid of the nonce, one can define a universal hashing-based MAC as $t = f(h(m))$. This will result in security bounded by $\varepsilon \cdot q_m^2$ as a hash collision will necessarily cause a tag collision. Again, as described above, this allows the adversary to identify hash collisions and recover the secret hashing key, which will in turn allow them to create more hash collisions and thus universal forgeries.

5.2.3 Improving the Wegman-Carter MAC

In [Sho96], Shoup replaced the function F with a permutation, addressing the fact that there exist a number of standardized and well-analyzed block ciphers. Bernstein later proved the security of Shoup's construction, e.g., [Ber05c]. Bernstein's well-known bound still ensures that the advantage for any adversary that asks $2^{n/2}$ authentication queries [Ber05b] is bounded by $1.7 q_v \ell / 2^n$, where q_v is the number of verification queries and ℓ is the maximal message length, usually in terms of elements of a ring or field used in h. Throughout this work, we adopt the common way of referring to security bounds that are negligible up to $\mathcal{O}(2^{n/2})$ blocks or queries as $n/2$ bits of security.

Despite its simplicity, one can identify two interesting directions for extending the Wegman-Carter construction. The first concerns the nonce requirement, which is a well-known considerable risk: If a single nonce is repeated only once, the security of the construction may collapse completely since the hash-function key could leak. Secondly, even if nonces never repeat, its security is inherently limited by Bernstein's bound, which is of

birthday-bound type. Recent works showed that Bernstein's bound is tight [LP18, Nan18], which means that the original construction cannot provide higher security.

6 | Improving Wegman-Carter

EWCDM. An ongoing series of research aims to find constructions with higher security guarantees that retained some security also under nonce reuse. As one of the starting points, one could identify the proposal of the Encrypted Davies-Meyer (EDM) and the Encrypted Wegman-Carter Davies-Meyer (EWCDM) modes by Cogliati et al. [CS16]. While EDM is a PRP-to-PRF conversion method and therefore restricted to inputs of n bits length, EWCDM supports nonce-based authentication for variable-input-length messages as does the original Wegman-Carter construction. In EWCDM, a nonce ν is first processed by the Davies-Meyer construction under a permutation π_1; its result is XORed with the hash of a message m and the sum is encrypted under a second independent permutation: $\pi_2(\pi_1(\nu) \oplus \nu \oplus h_{k'}(m))$. EDM misses the hash and uses ν as the only message input. Its authors showed that EWCDM provides at least $2n/3$-bit security in a nonce-respecting setting and birthday-bound security in a nonce-misusing setting. Their proof also showed $2n/3$-bit security for EDM. Recently, Cogliati and Seurin [CS18] showed that one can use the same permutation twice in EDM while retaining $2n/3$-bit security. EWCDM [CS16] started a trend in search of new message authentication constructions.

Improving EWCDM. DWCDM [DDNY18] sought to improve upon EWCDM by incorporating only a single key while still retaining the same $\mathcal{O}(2^{2n/3})$ bits of security. [MN17] showed an improved bound on security of up to $2^n/67n$ adversarial queries for EWCDM and up to $2^n/67$ queries for a dual of EWCDM called EWCDMD. The latter proof contained an mistake as EWCDMD suffers from a birthday type attack [Nan17]. Either construction only claimed birthday bound security in a nonce-repeating setting. [DDD21] showed that DWCDM with $3n/4$-bit nonces achieves $\mathcal{O}(2^{3n/4})$ security and EWCDM also achieved $\mathcal{O}(2^{3n/4})$ security. At that time, it was debated whether the Mirror Theory holds true for security larger than $\mathcal{O}(2^{2n/3})$ and thus the result was [MN17] was questioned. [DDD21] showed the extended Mirror Theory to hold for up to $\mathcal{O}(2^{3n/4})$ queries.

EWCDM concatenates two permutation calls, i.e. one permutation call uses as its input the output of the other permutation call. Hence, the permutation calls cannot be parallelized. [MN17] also showed a very simple proof of security for the sum of permutations as presented in Section 4.3 with security in $\mathcal{O}(2^n)$. For this construction the permutations

calls can be performed in parallel. Naturally, the question arises whether one can modify the sum of permutations to obtain a MAC.

TBC-based MACs. An alternative approach has been taken by Cogliati et al. [CLS17] using tweakable block ciphers (TBC). They proposed four generic constructions based on the composition of universal hashing and a block cipher: Hash-as-Tweak (HaT), Nonce-as-Tweak (NaT), Hash-as-Key (HaK), and Nonce-as-Key (NaK). They proved n-bit security for all constructions in the ideal-permutation model (assuming a universal hash function). However, the former two constructions require a tweakable primitive, whereas the latter two require message-dependent rekeying.

Goals. We can identify four desiderata for interesting MACs based on permutations and universal hashing. In terms of security, the adversary's advantage should remain negligible for $\ell q \gg 2^{n/2}$. In terms of simplicity, the number of calls to the primitive(s) should be minimized. For efficiency, their calls should be parallelizable, and frequent rekeying should be avoided. Last but not least, they should support variable-length messages. So, in spite of recent advances, it remains an interesting question how one can generally achieve those aspects for stateless deterministic and/or nonce-based constructions.

Enhanced Hash-then-Mask. Minematsu [Min10] proposed a construction called the Enhanced Hash-then-Mask (EHtM) MAC, which uses a random salt and two keyed functions that are xored to obtain $\mathcal{O}(2^{2n/3})$ security. [DJN17] improved the bound to $\mathcal{O}(2^{3n/4})$ and showed tightness by providing a matching attack.

[DNT19] showed in parallel to us [ML19] that EHtM with two keyed permutations instead of two keyed function and a nonce instead of a random salt is secure to up to $\mathcal{O}(2^{2n/3})$ queries. They further showed graceful degradation of security when nonces are repeated, i.e. security degrades as nonces are repeated more often and does not drop to the birthday bound with a single repetition. [CLLL20] improved the bound to $\mathcal{O}(2^{3n/4})$. [DN20] investigated the construction with public permutations and [CDN22] extended this to a multi-user setting. We analyzed this construction under the name HPxNP.

Double-block Hash-then-Sum. A different construction uses two hash values as inputs to the two permutations and was studied in parallel by [DDNP18] and us [ML19]. Both works show a bound of security of $\mathcal{O}(2^{2n/3})$. This was bound was later improved by [KLL20] to show a security level of $\mathcal{O}(2^{3n/4})$. Its security in a multi-user setting was evaluated and first shown to be $\mathcal{O}(2^{2n/3})$ [SWGW21] and then improved to $\mathcal{O}(2^{3n/4})$ [DDNT23]. We analyzed this construction under the name HPxHP.

Contribution. This work analyzes two constructions based on the sum of permutations and universal hashing with the help of the Mirror Theory. This first construction is stateless deterministic whereas our second is nonce-based. They are named HPxNP and HPxHP,

according to the fact whether they employ a universal hash function (HP) or a nonce (NP) as inputs to the permutation. Figure 6.1 illustrates them schematically. Both modes are shown to provide $\mathcal{O}(2n/3)$ bits of security asymptotically.

Structure. Hereupon, we first cover briefly the necessary preliminaries used in this work, including a brief recap of Patarin's Mirror Theory. Thereupon, Section 6.2 proposes our two constructions whose security is then analyzed in the subsequent Sections 6.4 and 6.5. Section 6.6 concludes.

Remark 6.1

We note that the HPxHP construction is clearly not novel, but an abstraction of a variety of existing double-lane MACs, e.g., 3KF9 [ZWSW12], GCM-SIV-2 [IM16], or PMAC$^+$ [Yas11]. However, in its abstract form, it has been studied by Datta et al. [DDNP18] (the same authors already had studied the construction in [DDN$^+$15]) from a constructive view; in parallel to our work, Dutta et al. also analyzed a variant of HPxNP with a single bit for domain separation in [DNT19], where they also showed $O(q^3/2^{2n})$ bits of security. Recently, Leurent et al. [LNS18] also studied an attacking view. More precisely, Leurent et al. [LNS18] proposed a forgery attack with data complexity of $\mathcal{O}(2^{3n/4})$ for such constructions. We also take the constructive view, so that our derived security bound is also inherently limited by the result by Leurent et al.; moreover, at the end of each analysis section, we further discuss the effect of using 4-wise independent hash functions for our constructions, with the positive result that the then-obtained security bounds render their result inapplicable and lead to higher security.

6.1 Preliminaries

In this section, we will briefly recap the notation and definitions used throughout this chapter. We use calligraphic uppercase letters $\mathcal{X}, \mathcal{Y}$ for sets. We write $\{0,1\}^n$ for the set of bit strings of length n, and denote the concatenation of binary strings x and y by $x \,\|\, y$ and the result of their bitwise XOR by $x \oplus y$. We write $x \twoheadleftarrow \mathcal{X}$ to mean that x is chosen uniformly at random from the set $\mathcal{X}$. We consider $\mathsf{Func}(\mathcal{X}, \mathcal{Y})$ to be the set of all deterministic mappings $F : \mathcal{X} \to \mathcal{Y}$ and $\mathsf{Perm}(\mathcal{X})$ to be the set of all permutations over $\mathcal{X}$. Given an event E, we denote by $\Pr[E]$ the probability of E. For two integers n, k with $n \geq k \geq 1$, we denote the falling factorial as $(n)_k \stackrel{\text{def}}{=} \prod_{i=0}^{k-1}(n-i)$.

Distinguisher. A (complexity-theoretic) distinguisher $\mathbf{A}$ is an efficient adversary, i.e., an efficient Turing machine that is given access to a number of oracles which it can interact with. The task of $\mathbf{A}$ is to distinguish between two worlds of oracles, one of which is chosen at the beginning of the experiment uniformly at random. After its interaction,

A outputs a bit that represents a guess of the world that **A** interacted with. The distinguishing advantage between a real world $\mathcal{O}_{\mathsf{real}}$ and an ideal world $\mathcal{O}_{\mathsf{ideal}}$ is given by $\Delta_{\mathbf{A}}(\mathscr{P}, \mathcal{O}) \stackrel{\text{def}}{=} \left|\Pr\left[\mathbf{A}^{\mathcal{O}_{\mathsf{real}}} = 1\right] - \Pr\left[\mathbf{A}^{\mathcal{O}_{\mathsf{ideal}}} = 1\right]\right|$. We consider information-theoretic distinguishers, i.e., distinguishers that are computationally unbounded, and that are limited only by the number of queries they can ask to their available oracles. We assume that distinguishers do not ask duplicate queries or queries to which they already can compute the answer themselves from earlier queries, as is common. W.l.o.g., we limit our interest to deterministic distinguishers since for each probabilistic distinguisher, there exists a deterministic one with equal advantage that fixed a random tape beforehand (cf. [ADMA15, CS14]).

Pseudorandomness and hash functions. We briefly recall the definitions for the advantage of distinguishing a construction from a random function (PRF) and from a random permutation (PRP), respectively.

Definition 6.1 (PRF *Advantage*)

Let $\mathscr{K}$, $\mathscr{X}$, and $\mathscr{Y}$ be non-empty sets and let $F : \mathscr{K} \times \mathscr{X} \to \mathscr{Y}$ and $\rho \twoheadleftarrow \mathsf{Func}(\mathscr{X}, \mathscr{Y})$ and $k \twoheadleftarrow \mathscr{K}$. Then, the PRF advantage of **A** w.r.t. F is defined as $\mathbf{Adv}^{\text{PRF}}_{F}(\mathbf{A}) \stackrel{\text{def}}{=} \Delta_{\mathbf{A}}(F_k, \rho)$.

A keyed permutation $E : \mathscr{K} \times \mathscr{X} \to \mathscr{X}$ is a family of permutation over $\mathscr{X}$ indexed by a key $k \in \mathscr{K}$.

Definition 6.2 (PRP *Advantage*)

Let $\mathscr{K}$ and $\mathscr{X}$ be non-empty sets, $E : \mathscr{K} \times \mathscr{X} \to \mathscr{X}$ be a keyed permutation, and let $\pi \twoheadleftarrow \mathsf{Perm}(\mathscr{X})$ and $k \twoheadleftarrow \mathscr{K}$. Then, the PRP advantage of **A** w.r.t. E is defined as $\mathbf{Adv}^{\text{PRP}}_{E_k}(\mathbf{A}) \stackrel{\text{def}}{=} \Delta_{\mathbf{A}}(E_k, \pi)$.

We will also need the definition of a d-wise independent family of hash functions.

Definition 6.3 (*d-wise Independence [WC81]*)

We say that $\mathscr{H}$ is d-independent if, for all pair-wise distinct $x_1, \ldots x_d \in \mathscr{X}$ and all $y_1, \ldots, y_d \in \mathscr{Y}^d$, it holds that $\Pr_{h \twoheadleftarrow \mathscr{H}}[h(x_i) = y_i, \text{ for } 1 \le i \le d] = 1/|\mathscr{Y}|^d$.

H-coefficient technique. We will briefly recapture the H-coefficients technique as it is central to our proof. The H-coefficients technique is a proof method due to Patarin, where we consider the variant by Chen and Steinberger [CS14, Pat08]. The results of the interaction of an adversary **A** with its oracles are collected in a transcript τ. The oracles can sample randomness prior to the interaction (often a key or an ideal primitive that is sampled beforehand), and are then deterministic throughout the experiment [CS14]. The task of **A** is to distinguish the real world $\mathcal{O}_{\mathsf{real}}$ from the ideal world $\mathcal{O}_{\mathsf{ideal}}$. Let Θ_{real} and Θ_{ideal} denote the distribution of transcripts in the real and the ideal world, respectively. A

transcript τ is called *attainable* if the probability to obtain τ in the ideal world – i.e. over Θ_{ideal} – is non-zero. Then, the fundamental Lemma of the H-coefficients technique, the proof to which is given in [CS14, Pat08], states:

Lemma 6.1 (***Fundamental Lemma of the H-coefficient Technique [Pat08]***)

Assume that the set of attainable transcripts can be partitioned into two disjoint sets GoodT and BadT. Further, assume that there exist $\delta, \epsilon \in [0,1]$ such that for any transcript $\tau \in \mathsf{GoodT}$, it holds that:

$$\Pr[\Theta_{\mathsf{ideal}} \in \mathsf{BadT}] \leq \delta \quad \text{and} \quad \frac{\Pr[\Theta_{\mathsf{real}} = \tau]}{\Pr[\Theta_{\mathsf{ideal}} = \tau]} \geq 1 - \epsilon.$$

Then for all adversaries $\mathbf{A}$ we have that the distinguishing advantage satisfies

$$\underset{\mathbf{A}}{\Delta}(\Theta_{\mathsf{ideal}}, \Theta_{\mathsf{real}}) \leq \delta + \epsilon.$$

Mirror Theory. We will use this in conjunction with the Mirror Theory from Section 4.2. We will use the Bad Events of the H-coefficients technique to ensure that all properties of the Mirror Theory are fulfilled and then apply the Mirror Theorem to the Good Transcripts.

Remark 6.2

We consider PRF security in the information-theoretic setting, similar to Mennink and Neves [MN17]. The underlying permutations are secret and assumed to be drawn uniformly at random from $\mathsf{Perm}(\{0,1\}^n)$. Our results generalize to the complexity-theoretic setting. There, the permutations π_1 and π_2 are supposed to be instantiated with a block cipher E under independent random secret keys k_1 and k_2, E_{k_1} and E_{k_2}, respectively. The bounds from this paper can be easily adapted to the complexity-theoretic setting by adding a term of $2 \cdot \mathbf{Adv}^{\mathrm{PRP}}_{E_k}(q)$. The term refers to twice the maximal advantage for an adversary $\mathbf{A}'$ to distinguish $E : \mathscr{K} \times \{0,1\}^n \to \{0,1\}^n$ keyed with a random key $k \twoheadleftarrow \mathscr{K}$ from a random permutation π, where $\mathbf{A}$ asks at most q queries. Note that we only employ the forward direction of the permutation; therefore, PRP security suffices and we do not need to consider the strong variant.

6.2 Constructions

Let $n \geq 1$ be a positive integer, and let $\mathscr{K}$ denote a non-empty set. Let $\pi_1, \pi_2 \twoheadleftarrow \mathsf{Perm}(\{0,1\}^n)$ be independently uniformly at random sampled permutations over n-bit strings. Let $\mathscr{H} = \{h \mid h : \{0,1\}^* \to \{0,1\}^n\}$ be a family of ε_1-AXU hash functions; for

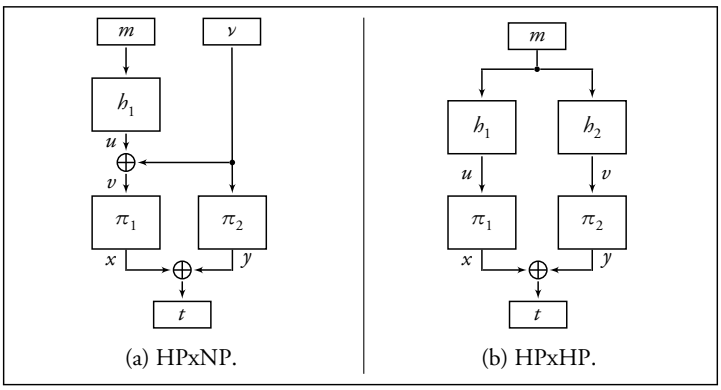

Figure 6.1: Our proposed constructions. π_1 and π_2 represent two permutations over $\{0,1\}^n$, h_1 and h_2 two universal hash functions, m a variable-length message, ν, ν^1, and ν^2 nonces of fixed length, and t the authentication tag.

HPxHP, we will define and use instead $\mathscr{H}_1 = \{h_1 \mid h_1 : \{0,1\}^* \to \{0,1\}^n\}$ be a family of ε_1-AU hash functions, and $\mathscr{H}_2 = \{h_2 \mid h_2 : \{0,1\}^* \to \{0,1\}^n\}$ be a family of ε_2-AU hash functions. We require the hash functions to be sampled independently uniformly at random. Usually, the hash function instances are determined by sampling a hash key independently uniformly at random for each instance.

Nonce-based MAC. Our first, nonce-based construction, HPxNP, is illustrated in Figure 6.1a. It shares similarities with Minematsu's Enhanced Hash-then-Mask construction [Min10] that had been analyzed further in [DJN16, DJN17]; however, Minematsu's construction used a function instead of a permutation and a per-message random IV. In this construction, the message is hashed to an n-bit value $h(m)$. For this construction, we need $\mathscr{H}$ to be an ε-almost-XOR-universal family of hash functions. An n-bit nonce ν is XORed to the hash u to obtain $v := h(m) \oplus \nu$; v and ν serve as inputs to the two calls to a permutation π_1 and π_2, respectively, and yield $x := \pi_1(v)$ and $y := \pi_2(\nu)$. Finally, the outputs of the permutation calls are XORed and released as authentication tag: $t := x \oplus y$.

Stateless MAC. Our second construction, HPxHP, is illustrated in Figure 6.1b. It consists of two parallel invocations of the hash functions on the input message $m \in \{0,1\}^*$ that are hashed using $h_1 \in \mathscr{H}_1$ and $h_2 \in \mathscr{H}_2$, respectively, to two n-bit values u and v. Those serve as inputs to the two calls to a permutation π_1 and π_2, respectively and yield $x := \pi_1(u)$ and $y := \pi_2(v)$. Finally, the outputs of the permutation calls are XORed and released as authentication tag: $t := x \oplus y$.

Block ciphers and hashing. In practice, the permutations π_1 and π_2 will be instantiated with a secure block cipher E under two independent keys k_1 and k_2. An intuitive choice for the hash function is, for example, polynomial hashing. Let $\mathbb{F}_{2^n}$ be the Galois

Field $GF(2^n)$ with a fixed primitive polynomial $p(\mathsf{x})$. For $n = 128$, the GCM polynomial $p(\mathsf{x}) = \mathsf{x}^{128} + \mathsf{x}^7 + \mathsf{x}^2 + \mathsf{x} + 1$ is a usual choice. The hash function is instantiated by sampling a hash key $k \twoheadleftarrow \mathbb{F}_{2^n}$. Given k and a message $m \in (\mathbb{F}_{2^n})^\ell$ of ℓ blocks, polynomial hashing is then defined as the sum of

$$h_k(m) \stackrel{\text{def}}{=} \sum_{i=1}^{\ell} k^{\ell+1-i} \cdot m_i,$$

where m_i denotes the i-th message block and additions as well as multiplications are in $\mathbb{F}_{2^n}$.

It is well-known that for maximal message lengths of ℓ blocks (after padding), polynomial hashing is ε-AXU for $\varepsilon = \ell/2^n$, and therefore also $\ell/2^n$-AU. Note that polynomial hashing requires an injective padding to prevent trivial hash collisions; a simple 10^*-padding works, but may extend messages by one block.

Parallelization. While the sum of a polynomial hash is sequential, computing the individual terms on a few cores in parallel is well-known at the cost of storing multiple powers of the hash key. For instance, optimized instances of GCM parallelize the computations of four (or eight) subsequent blocks $k^4 \cdot m_i$, $k^3 \cdot m_{i+1}$, $k^2 \cdot m_{i+2}$, and $k^4 \cdot m_{i+3}$, before their results are summed, reduced by the modulus, and summed to the sum of the previous blocks $\sum_{j=1}^{i-1} k^j m_j$ [GK14, GL15]. Thus, several hash multiplications, or two hash-function calls, or hashing and computing a permutation are efficiently parallelizable as long as the platform is not too resource-restricted. Note that a number of related hash functions exist with similar security properties; pseudo-dot-product hashing, BRW hashing, or combined approaches such as [CGS17] can half the number of necessary multiplications, and provide similar parallelizability. We refer the interested reader to an overview by Bernstein [Ber07].

6.3 Relation to the Attack by Leurent et al.

The attacks in [LNS18] exploit that 4-circles may occur after $2^{3n/4}$ queries if the hash functions are universal, and the messages are constructed in a dedicated manner. We briefly recall the attack by Leurent et al. [LNS18] here.

Attack description. Leurent et al. considered MACs with $2n$ bits of internal state that can be abstracted to HPxHP. They searched for four-tuples (x, y, z, t) such that they build a 4-circle as:

$$\begin{cases} h_1(x) = h_1(y) \\ h_2(x) = h_2(t) \\ h_1(t) = h_1(z) \\ h_2(y) = h_2(z). \end{cases}$$

Such a tuple can be efficiently verified since it must hold that their corresponding authentication tags sum to zero: $t_x \oplus t_y \oplus t_z \oplus t_t = 0^n$. Since practical instances of such MACs (e.g., PMAC$^+$, 3kf9, Sum-ECBC) hash the message block-wise, they further employ two distinct prefixes p_0 and p_1, such that $|p_0| = |p_1|$ and the prefixes end at the block boundary:

$$x = p_0 \,\|\, x_*, \quad y = p_1 \,\|\, y_*, \quad z = p_0 \,\|\, z_*, \quad t = p_1 \,\|\, t_*.$$

So, the prefixes lead to differences $\Delta = h_1(p_0) \oplus h_1(p_1)$ and $\nabla = h_2(p_0) \oplus h_2(p_1)$. Considering only four-tuples (x_*, y_*, z_*, t_*) with $x_* \oplus y_* \oplus z_* \oplus t_* = 0^n$, they could translate the problem of finding a solution to the rank-four equation system to the problem of finding a solution to the following rank-three system:

$$\begin{cases} h_1(x) = h_1(y) \\ h_2(x) = h_2(t) \\ h_1(t) = h_1(z) \\ h_2(y) = h_2(z). \end{cases} \Leftrightarrow \begin{cases} h_1(x) = h_1(y) \\ h_2(x) = h_2(t) \\ h_1(t) = h_1(z), \end{cases}$$

with $x_* \oplus y_* = z_* \oplus t_* = \Delta$ and $x_* \oplus t_* = y_* \oplus z_* = \nabla$, From $x_* \oplus y_* \oplus z_* \oplus t_* = 0^n$, it followed then that $h_2(x) = h_2(y)$ also holds. Leurent et al. propose data-efficient algorithms for this 4-sum problem, i.e., finding four-tuples with data complexity of $\mathcal{O}(2^{3n/4})$ queries.

6.4 Security Analysis of HPxNP

First, we consider the construction HPxNP. Patarin's approach [Pat10] allows us to obtain a bound of $\mathcal{O}(2n/3)$ bits of security. At the end of this section, we discuss the implications of considering ξ_{average} instead, as was also suggested ibidem.

Theorem 6.1

Let $n \geq 1, \xi \geq 2$ be integers, and $\mathscr{H} = \{h \,|\, h : \{0,1\}^* \to \{0,1\}^n\}$ be a family of ε-AXU hash functions with $h \twoheadleftarrow \mathscr{H}$. For any nonce-respecting PRF distinguisher $\mathbf{A}$ that asks at most $q \leq 2^n/(67\xi^2)$ queries, it holds that

$$\mathbf{Adv}^{\text{PRF}}_{\text{HPxNP}[h,\pi_1,\pi_2]}(\mathbf{A}) \leq \frac{2q^2 \cdot \varepsilon}{\xi^2} + \frac{\binom{q}{2} \cdot \varepsilon}{2^n} + \frac{q}{2^n}.$$

Note that in this case, the optimal choice of ξ to obtain the best bound is $2^{n/6}$, assuming that $\varepsilon \in \mathcal{O}(2^{-n})$. Then, the bound in Theorem 6.1 is dominated by the first term of $\mathcal{O}(q^2/2^{4n/3} + q^2/2^{2n} + q/2^n)$, while the number of queries is allowed to be $q \leq 2^{2n/3}$. Other values for ξ reduce either the security bound or the number of queries.

The remainder of this section is devoted to show Theorem 6.1. Here, $\mathbf{A}$ makes q con-

struction queries (ν_i, m_i), for $1 \leq i \leq q$, that are stored together with the query results t_i in a transcript $\tau = \{(\nu_i, m_1, t_1), \ldots, (\nu_q, m_q, t_q)\}$. In both worlds, the oracle samples h at the beginning uniformly at random from all hash instances. **A** sees the results t_i after each query. We employ a common method to alleviate the proof: after the adversary finished its interaction with the oracle, but before outputting its final decision bit, **A** is given the hash-function instance h so that it can compute the values $u_1, \ldots, u_q$ itself. Clearly, this only makes the adversary stronger, but spares the need to discuss security internals of the hash function.

Let $1 \leq r \leq 2q$ and consider the set $\mathscr{P} = \{P_1, \ldots, P_r\}$ of r unknowns. We consider a system of q equations

$$\mathscr{E} = \{P_{a_1} \oplus P_{b_1} = t_1, \quad P_{a_2} \oplus P_{b_2} = t_2, \quad \ldots, \quad P_{a_q} \oplus P_{b_q} = t_q\},$$

where $P_{a_i} := x_i = \pi_1(h(m_i) \oplus \nu_i)$ and $P_{b_i} := y_i = \pi_2(\nu_i)$. We further define an index mapping $\varphi : \{a_1, b_1, \ldots, a_q, b_q\} \to \{1, \ldots, r\}$. For all $i, j \in \{1, \ldots, q\}$:

- $\varphi(a_i) \neq \varphi(a_j) \Leftrightarrow h_1(m_i) \oplus \nu_i \neq h_1(m_j) \oplus \nu_j$.
- $\varphi(b_i) \neq \varphi(b_j)$ since $\nu_i \neq \nu_j$.
- $\varphi(a_i) \neq \varphi(b_j)$ since both permutations π_1 and π_2 are independent.

The index mapping φ has a range of size $q_x + q_y$, where $q_x = |\{x_i, \ldots, x_q\}| \leq q$ and $q_y = |\{\nu_1, \ldots, \nu_q\}| = q$.

6.4.1 Bad Transcripts

φ only exposes collisions of the form $\varphi(a_i) = \varphi(a_j)$ or equivalently $x_i = x_j$. We define the following bad events:

Bad Events

- bad_1: there exist ξ distinct equation indices $i_1, i_2, \ldots, i_\xi \in \{1, \ldots, q\}$ s.t. $x_{i_1} = x_{i_2} = \ldots = x_{i_\xi}$ where ξ is the threshold given in Theorem 6.1.
- bad_2: There exist query indices $i \neq j$, $i, j \in \{1, \ldots, q\}$ s.t. $(v_i, t_i) = (v_j, t_j)$.

Let us consider bad_1 first. Since h is ε-AXU, the expected amount of collisions is $q^2 \cdot \varepsilon$. Unfortunately ε-AXU is not strong enough to allow for statements regarding multicollisions, i.e. we cannot make a statement on the probability that three or more input values collide. Considering the maximal block size ξ, the worst case would be that all collisions occur in the same hash value. If there exists a block of size $(\xi + 1)$, this block contains ξ^2 collisions. Let $\#\mathsf{Colls}(q)$ be the random variable that counts the collisions in h. By

Markov's Inequality, the probability that there are more than $\binom{\xi}{2}$ collisions in h is at most:

$$\Pr\left[\#\mathsf{Colls}_1(q) \geq \binom{\xi}{2}\right] \leq \frac{\mathbb{E}(C)}{\binom{\xi}{2}} = \frac{\binom{q}{2}\cdot\varepsilon}{\binom{\xi}{2}} \leq \frac{2q^2\varepsilon}{\xi^2}.$$

For bad_2, recall that the ideal world samples the tags independently uniformly at random. Since h is ε-AXU, it follows for some distinct pair $i, j \in \{1,\ldots,q\}$:

$$\Pr\left[v_i = v_j \wedge t_i = t_j\right] \leq \frac{\binom{q}{2}\cdot\varepsilon}{2^n}.$$

It follows from the sum of both probability for bad_1 and bad_2 that

$$\Pr\left[\tau \in \mathsf{BadT} \,|\Theta_{\mathsf{ideal}} = \tau\right] \leq \frac{2q^2\cdot\varepsilon}{\xi^2} + \frac{\binom{q}{2}\cdot\varepsilon}{2^n}.$$

6.4.2 Ratio of Good Transcripts

Lemma 6.2

The system of equations is (i) circle-free, (ii) ξ-block-maximal and (iii) relaxed non-degenerate with respect to the partitioning into $\mathscr{R}_1 \sqcup \mathscr{R}_2$, where

$$\mathscr{R}_1 =^{\text{def}} \{\varphi(a_1),\ldots,\varphi(a_q)\} \text{ and } \mathscr{R}_2 =^{\text{def}} \{\varphi(b_1),\ldots,\varphi(b_q)\}.$$

Proof. The proof relies on the fact that $\varphi(b_i) \neq \varphi(b_j)$ and $\varphi(a_i) \neq \varphi(b_j)$ for any $i \neq j$. For any $\mathscr{I} \subseteq \{1,\ldots,q\}$ the corresponding multiset $M_{\mathscr{I}}$ has at least $|\mathscr{I}|$ odd multiplicity elements and therefore the system of equations $\mathscr{E}$ is (i) circle-free.

(ii) If $\mathscr{E}$ were not ξ-block-maximal, then there must be an ordering $\mathscr{I} = \{i_1,\ldots,i_\xi\}$ s.t. $\varphi(a_{i_1}) = \ldots = \varphi(a_{i_\xi})$. This is equivalent to a ξ-fold collision $x_{i_1} = \ldots = x_{i_\xi}$, which contradicts the assumption that τ is a good transcript.

(iii) Suppose that $\mathscr{E}$ would be relaxed degenerate. Then, there would exist a minimal subset $\mathscr{I} \subseteq 1,\ldots,q$ that has exactly two odd multiplicity elements corresponding to the same oracle and s.t. $\bigoplus_{i\in\mathscr{I}} t_i = 0$. If $|\mathscr{I}| = 1$, $M_{\mathscr{I}}$ would have two elements from different oracles. If $|\mathscr{I}| = 2$ and $t_{i_1} = t_{i_2}$, then we would know that $x_{i_1} \neq x_{i_2}$ since $\nu_{i_1} \neq \nu_{i_2}$, i.e. $y_{i_1} \neq y_{i_2}$. Therefore, we have four odd multiplicity elements. If $|\mathscr{I}| \geq 3$, there would exist at least three odd multiplicity elements. So, $\mathscr{E}$ cannot be relaxed degenerate, which concludes the proof. ∎

Lemma 6.3

Let $\tau \in \mathsf{GoodT}$ and $q \leq 2^n/(67\xi^2)$. Then, it holds that

$$\frac{\Pr[\Theta_{\mathsf{real}} = \tau]}{\Pr[\Theta_{\mathsf{ideal}} = \tau]} \geq 1 - \frac{q}{2^n}.$$

Proof. The probability to obtain a good transcript τ consists of that for obtaining the tags $t_1, \ldots, t_q$, and the hash-function outputs $h(m_i)$. The probability to obtain the latter is given in both worlds by $|\mathscr{H}|^{-1}$. The bound in Lemma 6.3 is determined by the ratio of the respective probabilities. This term appears in the real world as well as in the ideal world and cancels out eventually. Hence, we ignore it for the remainder of the analysis. The probability of obtaining the rest of the transcript, i.e., the tags t_i, in the ideal world is then given by

$$\Pr\left[t_1, \ldots, t_q \middle| \Theta_{\mathsf{ideal}} \right] = \frac{1}{(2^n)^q}$$

since the outputs t_i are sampled independently and uniformly at random from $\{0,1\}^n$ in the ideal world. In the real world, the probability is given by

$$\begin{aligned}\Pr[\Theta_{\mathsf{real}} = \tau] &\geq \frac{\frac{\mathsf{NonEQ}(\mathscr{R}_1, \mathscr{R}_2; \mathscr{E})}{2^{nq}} \cdot (2^n - q_x)! \cdot (2^n - q_y)!}{(2^n!)^2} \\ &= \frac{\mathsf{NonEQ}(\mathscr{R}_1, \mathscr{R}_2; \mathscr{E})}{2^{nq}(2^n)_{q_x}(2^n)_{q_y}}.\end{aligned}$$

Remember that $q_y = q$ since all ν_i are distinct. To lower bound $\mathsf{NonEQ}(\mathscr{R}_1, \mathscr{R}_2; \mathscr{E})$, note that we have $(2^n)_{q_x}$ choices for $\{P_j \,|\, j \in \mathscr{R}_1\}$ and at least $(2^n - 1)_q$ possible choices for $\{P_j \,|\, j \in \mathscr{R}_2\}$, as every index in $\mathscr{R}_2$ is in a block with exactly one unknown from $\mathscr{R}_1$. Thus

$$\Pr[\Theta_{\mathsf{real}} = \tau] \geq \frac{(2^n - 1)_q (2^n)_{q_x}}{2^{nq}(2^n)_q (2^n)_{q_x}} = \frac{1}{2^{nq}} \left(1 - \frac{q}{2^n}\right).$$

Hence, we obtain the ratio as in Lemma 6.3. ∎

6.4.3 Using ξ_{average}

In [Pat10], Patarin suggests that one can potentially consider the average instead of the maximal block size for the sum of permutations in the Mirror Theory. More precisely, Generalization 2 of [Pat10, Section 6] suggests that:

> The theorem $P_i \oplus P_j$ is still true if we change the condition $\xi_{\max}\alpha \ll 2^n$ by $\xi_{\mathsf{average}} \ll 2^n$.

The bottleneck in our bound is the event bad_1; bad_2 as well as the good transcripts do not consider ξ at all and the respective terms become significant only for q approaching 2^n. Upper bounding the block size is necessary to ensure the condition $q \leq 2^n/(67\xi_{\max}^2)$. Using a universal family of hash functions only allows for a very crude upper bound of the maximal block size that limits us at a security level of around $2^{2n/3}$ queries.

If we could use the average block size as suggested by Patarin, we are limited by the condition $q \leq 2^n/(67\xi_{\text{average}}^2)$; then, bad_1 would no longer be necessary and would significantly improve the bound. The following theorem would yield an upper bound on the expected average block size ξ_{average}.

Theorem 6.2

For any $q \leq 2^n$ and $\varepsilon \leq 1$, we expect that $\xi_{\text{average}} \leq (q-1)\varepsilon + 2$.

We will first briefly sketch the idea for $\varepsilon = 2^{-n}$ and explain the idea in more detail below: For $q \ll 2^n$, the expected amount of collisions $q^2/2^n$ is in $\mathcal{O}(q)$. For $q = 2^n$, the expected amount of collisions is 2^{n-1}. In the worst case (regarding the average), the collisions are uniformly distributed, i.e. $h(m_1) = h(m_2), h(m_3) = h(m_4), \ldots, h(m_{2^n-1}) = h(m_{2^n})$. This pattern corresponds to the case that every block were of size 3 and hence the average is 3 as well. Any other pattern would not increase the average block size. The proof will consider the more general case for ε. From Theorem 6.2, we obtain

$$q \leq \frac{2^n}{67((q-1)\varepsilon + 2)^2}.$$

We note that the use of ξ_{average} implies the need to employ the stronger form of the Mirror Theory, that assumes that the iterated proof suggested by Patarin holds. Both the stronger form of the Mirror Theory and the Generalization 2 [Pat10] are subject to their own analysis.

Proof Sketch. Note that we argue about an upper bound on the *expected* average block size. To use this argument in a proper security proof, we would also need to bound the amount of collisions such that after q queries no more than q collisions occur. This can be done quite comfortably using Markov's Inequality.

We recall that ξ_{average} is the average block size of non-empty blocks of equations. More formally, we define 2^n bins $i \in \{0, \ldots, 2^n - 1\}$, where each bin i represents the n-bit value i that the hash values $u = h(m)$ can take. Over q queries, we define the number of non-empty bins by $B =^{\text{def}} |\{i : \exists j \in \{1, \ldots, q\} \text{ s.t. } u_j = i\}|$. We denote by ℓ_i the load of the i-th bin, i.e., the number of queries $u = h(m)$ that were equal to i, all over q queries, for $1 \leq i \leq q$. The average bin load over all non-empty bins is given by

$$\ell_{\text{average}} \stackrel{\text{def}}{=} \frac{1}{B} \sum_{i=0}^{2^n-1} \ell_i = \frac{q}{B}$$

since the sum of all bin loads must yield q. We denote by b_i the block size that corresponds to bin i in our proof since a block contains all variables corresponding to tuples (u, v) with $u = i$ plus the ℓ_i disjoint nonces. So, $b_i =^{\text{def}} \ell_i + 1$ if $\ell_i > 0$ and $b_i =^{\text{def}} 0$ if $\ell_i = 0$, i.e., if bin i is empty. It follows from our definitions above that $\xi_{\text{average}} = \ell_{\text{average}} + 1$. In total, we expect $\binom{q}{2}\varepsilon$ collisions which are distributed over all bins, i.e.,

$$\#\mathsf{Colls} = \binom{q}{2}\varepsilon = \sum_{i=0}^{2^n-1}\binom{\ell_i}{2}. \tag{1}$$

To show the claim, our goal is to maximize ξ_{average}, which is equivalent to maximize ℓ_{average}, which again is equivalent to minimizing B, i.e., the number of non-empty bins.

We have to show two aspects that ℓ_{average} is largest if the distribution of balls is closest possible to uniform while maintaining the expected number of collisions. We can observe that the average block size decreases whenever we would move a ball from one bin to another so that the load of both diverges. Given two disjoint bin indices $i, j \in \{0, \ldots, 2^n - 1\}$ with loads ℓ_i and ℓ_j, respectively. W.l.o.g., we assume that $\ell_i \geq \ell_j$. We have that

$$\#\mathsf{Colls} = \binom{\ell_i}{2} + \binom{\ell_j}{2} + \left(\binom{q}{2}\varepsilon - \binom{\ell_i}{2} - \binom{\ell_j}{2}\right) \stackrel{\text{def}}{=} \binom{\ell_i}{2} + \binom{\ell_j}{2} + R.$$

We move a ball from bin j to bin i and obtain new loads $\ell_i' = \ell_i + 1$ and $\ell_j' = \ell_j - 1$. We obtain that

$$\#\mathsf{Colls}' = \binom{\ell_i'}{2} + \binom{\ell_j'}{2} + R = \#\mathsf{Colls} + \ell_i - \ell_j + 1 \geq \#\mathsf{Colls} + 2.$$

So, whenever we move a ball such that the resulting bin loads diverge more, the number of collisions used up by those bins increases. Hence, we have less collisions remaining for the remaining bins, which implies that the balls in the remaining bins have to be moved:

- either from some bin i' to j' such that $\ell_{i'} - \ell_{j'} \geq \ell_i - \ell_j$,
- or between multiple bins,
- or balls have to be moved to previously empty bins.

It is easy to see that this configuration is optimal when the individual non-empty bin loads diverge as little as possible. This is given by having B non-empty bins of the same load ℓ_i s. t.

$$\binom{\ell_i}{2} \cdot B = \binom{q}{2}\varepsilon.$$

Since $q = B \cdot \ell_i$, we obtain $\ell_i = (q-1)\varepsilon + 1$. It follows that the maximal average block size is $\xi_{\text{average}} = (q-1)\varepsilon + 2$. ∎

6.5 Security Analysis of HPxHP

The analysis of HPxHP shares many similarities with that of HPxNP, but differs in certain key points. Regarding the maximum block size, a hash collision (considering the hashes separately) may occur now on one of both sides, i.e., there may be a collision in $h_1(m) = h_1(m')$ or in $h_2(m) = h_2(m')$, which increases the block size and effectively doubles the probability of obtaining a hash collision.[1] Further, since collisions may occur on both sides, it is possible to obtain a circle.

Using a universal hash function, we can obtain security up to $\mathcal{O}(2^{2n/3})$ queries, matching the security bound of earlier analyses. Increasing the strength of the hash function and using a d-wise independent hash function, it is possible to obtain security up to $\mathcal{O}(2^{\frac{(n-1)d}{d+1}})$ queries. Putting stronger requirements on the family of hash functions increases its size and therefore the length of the key. We still find this result interesting since recent results [LNS18] provided attacks with a query complexity of $\mathcal{O}(2^{3n/4})$. If we demand stronger properties from the hash function, our security level exceeds the complexity by the known attacks. Again, we provide an analysis with a universal hash function and $\xi_{\max}$ first. Thereupon, we will argue about the necessary proof changes to adapt to stronger hash-function families.

Theorem 6.3

Let $n \geq 1, \xi \geq 2$ be integers and $\mathcal{H}_1$ and $\mathcal{H}_2$ be ε_1 and ε_2-AU families of hash functions, respectively, and let $h_1 \twoheadleftarrow \mathcal{H}_1$ and $h_2 \twoheadleftarrow \mathcal{H}_2$ be sampled independently uniformly at random. Let $\varepsilon =^{\text{def}} \max\{\varepsilon_1, \varepsilon_2\}$. For any PRF distinguisher $\mathbf{A}$ that asks at most $q \leq 2^n/(67\xi^2)$ queries, it holds that

$$\mathbf{Adv}^{\text{PRF}}_{\text{HPxHP}[h_1,h_2,\pi_1,\pi_2]}(\mathbf{A}) \leq \frac{4q^2\varepsilon}{\xi^2} + 3 \cdot (q\varepsilon)^2 + q^3\varepsilon^2 + \frac{\xi \cdot q}{2^n - \xi}.$$

For $\xi = 2^{n/6}$, and assuming an optimal $\varepsilon = \mathcal{O}(2^{-n})$, the bound in Theorem 6.3 has the form of $\mathcal{O}(q^2/2^{4n/3} + q^2/2^{2n} + q^3/2^{2n} + q/2^{5n/6})$ for $q \in \mathcal{O}(2^{2n/3})$ queries. So, it is dominated by the first term. The remainder of this section contains the proof of Theorem 6.3. Consider a deterministic distinguisher $\mathbf{A}$ that has access to either HPxHP$[h_1, h_2, \pi_1, \pi_2]$ or ρ, which chooses the outputs given to $\mathbf{A}$ uniformly at random. $\mathbf{A}$ makes q construction queries m_i that are stored together with the query results t_i in a transcript $\tau = \{(m_1, t_1), \ldots, (m_q, t_q)\}$. In both worlds, the oracle samples h_1 and h_2 at the beginning independently and uniformly at random from their hash families. $\mathbf{A}$ sees the results t_i after each query. Again, we make the adversary stronger by defining that the hash keys are revealed to the adversary after it finished its interaction with the oracle, but before outputting its final decision bit.

[1] Technically speaking, there is a total of $q(q-1)/2$ of input pairs. When bounding the probability of a collision we used q^2 instead, ignoring the factor $1/2$.

Let $1 \leq r \leq 2q$ and consider the set $\mathscr{P} = \{P_1, \ldots, P_r\}$ of r unknowns. Again, we consider a system of q equations

$$\mathscr{E} = \{P_{a_1} \oplus P_{b_1} = t_1, \quad P_{a_2} \oplus P_{b_2} = t_2, \quad \ldots, \quad P_{a_q} \oplus P_{b_q} = t_q\},$$

where $P_{a_i} := x_i = \pi_1(h_1(m_i))$ and $P_{b_i} := y_i = \pi_2(h_2(m_i))$. We further define an index mapping $\varphi : \{a_1, b_1, \ldots, a_q, b_q\} \to \{1, \ldots, r\}$; φ maps equal permutation outputs $x_i = x_j$ that occur for any $i \neq j$ (from equal hash values $u_i = u_j$) to the same unknown P_k; similarly, φ maps equal permutation outputs $y_i = y_j$ that occur for any $i \neq j$ (from equal hash values $v_i = v_j$) to the same unknown P_ℓ. For all $i, j \in \{1, \ldots, q\}$, it holds that

- $\varphi(a_i) \neq \varphi(a_j) \Leftrightarrow h_1(m_i) \neq h_1(m_j)$.
- $\varphi(b_i) \neq \varphi(b_j) \Leftrightarrow h_2(m_i) \neq h_2(m_j)$.
- $\varphi(a_i) \neq \varphi(b_j)$ since both permutations π_1 and π_2 are independent.

In the real world, the transcript has collisions in the values $x_i = x_j$ or $y_i = y_j$ for $i \neq j$, when the corresponding hash values $u_i = u_j$ or $v_i = v_j$ collide. A collision in x_i and x_j corresponds to a collision in $\varphi(a_i)$ and $\varphi(a_j)$ and a collision in y_i and y_j corresponds to a collision in $\varphi(b_i)$ and $\varphi(b_j)$. Multi-collisions in the range values of π_1 and π_2 correspond to blocks in the mirror theory. To upper bound the size of the largest block $\mathscr{Q}_k$, we need to consider a special type of collision between two queries i and j. In this setting, we say that two queries i and j collide if $h_1(m_i) = h_1(m_j)$ and/or[2] $h_2(m_i) = h_2(m_j)$. The probability for such a collision to happen is $\varepsilon_1 + \varepsilon_2 \leq 2\varepsilon$.

We define an event bad_1 if there exists a ξ-multi-collision in any subset of queries $\{i_1, \ldots, i_{\xi+1}\} \subseteq \{1, \ldots, q\}$, where ξ is the threshold in Theorem 6.3. We need to consider four more events that render a transcript to be bad:

Bad Events

- bad_1: There exists a subset $\mathscr{I} \subseteq \{1, \ldots, q\}$ of size $|\mathscr{I}| = \xi$, s.t. for each pair of distinct indices $i, j \in \mathscr{I}$, it holds that $\varphi(a_i) = \varphi(a_j)$ and/or $\varphi(b_i) = \varphi(b_j)$; ξ is the threshold in Theorem 6.3.
- bad_2: There exist $i \neq j$, $i, j \in \{1, \ldots, q\}$ s.t. $(u_i, v_i) = (u_j, v_j)$ and $t_i \neq t_j$.
- bad_3: There exist $i \neq j$, $i, j \in \{1, \ldots, q\}$ s.t. $(u_i, t_i) = (u_j, t_j)$ and $v_i \neq v_j$.
- bad_4: There exist $i \neq j$, $i, j \in \{1, \ldots, q\}$ s.t. $(v_i, t_i) = (v_j, t_j)$ and $u_i \neq u_j$.
- bad_5: There exists a subset $\mathscr{I} \subseteq \{1, \ldots, q\}$ s.t. $\mathscr{M}_{\mathscr{I}}$ contains only elements of even multiplicity.

[2] To avoid confusion, by 'and/or' we actually mean the logical 'or'.

If an attainable transcript τ is not bad, we define τ as good. We denote by GoodT and BadT the sets of good and bad transcripts, respectively. In the H-coefficient technique, the probability that a transcript is bad is analyzed solely for the ideal world. The bound in Theorem 6.3 follows then from Lemma 6.1 and Lemmas 6.4–6.6.

6.5.1 Bad Transcripts

Lemma 6.4

Let $\xi \geq 1$ denote the threshold from Theorem 6.3. It holds that

$$\Pr[\,\tau \in \mathsf{BadT} | \Theta_{\mathsf{ideal}} = \tau] \leq \frac{4q^2\varepsilon}{\xi^2} + 3 \cdot (q\varepsilon)^2 + q^3\varepsilon^2.$$

Proof. In the following, we upper bound the probability that a transcript is bad. Most of the time, we can upper bound the probabilities of the individual bad events to occur and will simply take the sum of their probabilities. We will postpone the discussion of the first bad event to the end and begin with the second bad event.

For the second bad event, it holds that h_1 and h_2 are both ε-AU and independent. We drop the condition $t_i \neq t_j$ since it only decreases the probability and an upper bound suffices for our purpose. The probability that both hash values collide simultaneously for two queries is at most

$$\Pr[\mathsf{bad}_2] \leq \binom{q}{2}\varepsilon^2 \leq \frac{q^2\varepsilon^2}{2}.$$

For the third and fourth bad events, the probabilities can be formulated similarly. To upper bound bad_3, the probability that $u_i = u_j$ is again at most ε for a fixed pair of distinct query indices $i \neq j$. Since the outputs t_i and t_j are sampled uniformly at random and independently from the hash values, we can again neglect the requirement $v_i \neq v_j$ and obtain the same upper bound for bad_3 as for bad_2 when we use $\varepsilon \geq 2^{-n}$. A similar argument holds for bad_4.

When upper bounding the probability of bad_5, we are limited by the hash function. We consider all 3-tuples (m_a, m_b, m_c) such that $h_1(m_a) = h_1(m_b)$ and $h_2(m_b) = h_2(m_c)$. This event can be bounded by $\binom{q}{3}\varepsilon^2$, which also excludes the occurrence of a circle. Thus, it holds that $\Pr[\mathsf{bad}_5] \leq q^3\varepsilon^2$. Double-collisions that are small circles by themselves are excluded by bad_2.

Now, we will consider bad_1. As in the analysis of HPxNP we will upper bound the maximal block size for the individual hash functions. We will then condition bad_1 on $\neg\mathsf{bad}_5$ to ensure that no collisions in h_1 are connected to collisions in h_2. The hash functions are both ε-almost-universal. Again, the worst case regarding block maximality would

be that all collisions occur in the same block of size $\xi + 1$. Such a block would have $\binom{\xi}{2}$ collisions. Let $\#\mathsf{Colls}_1(q)$ denote a random variable for the number of collisions between $h_1(m_i) = h_1(m_j)$ for $1 \le i, j \le q$ and $i \ne j$. Using Markov's Inequality, we obtain an upper bound for the probability that

$$\Pr\left[\#\mathsf{Colls}_1(q) \ge \binom{\xi}{2}\right] \le \frac{\mathbb{E}[\#\mathsf{Colls}_1(q)]}{\binom{\xi}{2}} \le \frac{2q^2\varepsilon}{\xi^2}.$$

We can derive a similar argument using a random variable $\#\mathsf{Colls}_2(q)$ for the number of collisions between collisions $h_2(m_i) = h_2(m_j)$, So, the probability to obtain a block of size ξ is upper bounded by

$$\Pr\left[\mathsf{bad}_1 | \neg\mathsf{bad}_5\right] \le \frac{4q^2\varepsilon}{\xi^2}.$$

Our bound in Lemma 6.4 follows from summing up the obtained terms. ■

6.5.2 Good Transcripts

It remains to upper bound the ratio of probabilities to obtain a good transcript in both worlds. To upper bound it in the real world, we will use the Relaxed Mirror Theory. We show that a good transcript fulfills all the properties needed by the Relaxed Mirror Theorem.

Lemma 6.5

Let $\tau \in \mathsf{GoodT}$ be a good transcript. Let $\mathscr{E}$ be the system of q equations corresponding to $(\varphi^\tau, m_1, \ldots, m_q)$. Then, $\mathscr{E}$ is (i) circle-free, (ii) ξ-block-maximal, and (iii) relaxed non-degenerate with regard to the partitioning $\{1, \ldots, r\} = \mathscr{R}_1 \cup \mathscr{R}_2$, where $\mathscr{R}_1 = \{\varphi(a_i), \ldots, \varphi(a_q)\}$ and $\mathscr{R}_2 = \{\varphi(b_i), \ldots, \varphi(b_q)\}$.

Proof. We defined τ to be a good transcript; hence, no bad event has occurred, which implies that the transcript is (i) circle-free since we excluded bad_5 here.

(ii) If $\mathscr{E}$ were not ξ-block-maximal, there would exist a minimal subset $\mathscr{Q} \subseteq \{1, \ldots, r\}$ with $|\mathscr{Q}| \ge \xi + 1$ so that there exists some $i \in \{1, \ldots, q\}$ for which either $\{\varphi(a_i), \varphi(b_i)\} \subseteq \mathscr{Q}$ or $\{\varphi(a_i), \varphi(b_i)\} \cap \mathscr{Q} = \emptyset$. The latter event does not violate the block-maximality, so we can focus on the former statement.

Assuming that $\mathscr{E}$ were not ξ-block-maximal, we can define a subset of indices $\mathscr{I} \subset \{1, \ldots, q\}$ for which it holds that $\{\varphi(a_i), \varphi(b_i)\} \subseteq \mathscr{Q}$ for all $i \in \mathscr{I}$. Then, we can define an ordered sequence of the indices in $\mathscr{I}$ to $i_1, \ldots, i_\xi$ s.t. it would have to hold for all pairs of subsequent indices i_j, i_{j+1}, for $1 \le j < \xi$ that $\varphi(a_i) = \varphi(a_j)$ and/or $\varphi(b_i) = \varphi(b_j)$. This is equivalent to our definition of bad_1 and would therefore violate our assumption that τ is good. Hence, every good transcript τ is ξ-block-maximal.

(iii) Assume that τ would be relaxed degenerate. This would imply there exists a subset $\mathscr{I} \subseteq \{1,\ldots,q\}$ such that the multiset $M_{\mathscr{I}}$ has exactly two odd multiplicity elements from a single set $\mathscr{R}_1$ or $\mathscr{R}_2$ and the tags of the elements corresponding to $\mathscr{I}$ sum up to zero, i.e.

$$\bigoplus_{i\in\mathscr{I}} t_i = \bigoplus_{i\in\mathscr{I}} \pi_1(h_1(m_i)) \oplus \pi_2(h_2(m_i)) = 0.$$

Recall that $\varphi(a_i) \neq \varphi(a_j)$ if and only if $h_1(m_i) \neq h_1(m_j)$, $\varphi(b_i) \neq \varphi(b_j)$ if and only if $h_2(m_i) \neq h_2(m_j)$ and $\varphi(a_i) \neq \varphi(b_j)$ for any choice of i and j. An element $\varphi(a_i)$ has even multiplicity in $M_{\mathscr{I}}$ if there is an even amount of inputs that collide in $h_1(m_i)$. And similarly an element $\varphi(b_i)$ has even multiplicity in $M_{\mathscr{I}}$ if there is an even amount of inputs that collide in $h_2(m_i)$. If there is an even amount of queries that collide in a hash value, one can easily see that these elements will cancel out in the above sum.

For simplicity, assume, there exists a subset $\mathscr{I} \subseteq \{1,\ldots,q\}$ with exactly two odd multiplicity elements from $\mathscr{R}_1$ and even multiplicity elements only from $\mathscr{R}_2$. All elements from $\mathscr{R}_2$ cancel out in the sum above. and all even multiplicity elements from $\mathscr{R}_1$ cancel out as well. Let the two odd multiplicity elements from $\mathscr{R}_1$ have multiplicity $2n_1+1$ and $2n_2+1$, where $n_1, n_2 \geq 0$. In total, $2n_1$ and $2n_2$ terms will cancel out and what remains is $\pi_1(h_1(m_i)) \oplus \pi_1(h_1(m_j)) = 0$ where $\varphi(a_i) \neq \varphi(a_j)$. However, this event cannot occur since $\varphi(a_i) \neq \varphi(a_j)$ implies that $h_1(m_i) \neq h_1(m_j)$; thus the system cannot be relaxed degenerate. ■

Lemma 6.6

Let $\tau \in \mathsf{GoodT}$ and $q \leq 2^n/(67\xi^2)$. Then, it holds that

$$\frac{\Pr[\Theta_{\mathsf{real}} = \tau]}{\Pr[\Theta_{\mathsf{ideal}} = \tau]} \geq 1 - \frac{\xi \cdot q}{2^n - \xi}.$$

Proof. The probability to obtain a good transcript τ consists of that for obtaining the tags $t_1,\ldots,t_q$, and the hash-function outputs u_i and v_i. The probability to obtain the latter is given in both worlds by $\Pr[(h_1,h_2) \,|\, (h_1,h_2) \twoheadleftarrow \mathscr{H}_1 \times \mathscr{H}_2]$. The bound in Lemma 6.6 is determined by the ratio of the respective probabilities. This term appears in the real world as well as in the ideal world and cancels out eventually. Hence, we ignore it for the remainder of the analysis. The probability for the tags t_i in the ideal world is then given by $\Pr[t_1,\ldots,t_q|\Theta_{\mathsf{ideal}}] = 1/(2^n)^q$ since the outputs t_i are sampled independently and uniformly at random from $\{0,1\}^n$ in the ideal world.

In the real world, the situation is more complex and a little more work is necessary. We denote by $q_x := |\{\pi_1(h_1(m_i)) \mid i \in \{1,\ldots,q\}\}|$ the amount of distinct values for π_1 and similarly we denote by $q_y := |\{\pi_2(h_2(m_i)) \mid i \in \{1,\ldots,q\}\}|$ the amount of distinct values for π_2. The number of solutions to the $q_x + q_y$ unknowns is at least

$\mathsf{NonEQ}(\mathscr{R}_1,\mathscr{R}_2;\mathscr{E})/2^{nq}$. There are $(2^n-q_x)!$ possible choices for the remaining output values of π_1 and $(2^n-q_y)!$ possible choices for the remaining output values of π_2. Thus, we can lower bound

$$\Pr[\Theta_{\mathsf{real}}=\tau] \geq \frac{\frac{\mathsf{NonEQ}(\mathscr{R}_1,\mathscr{R}_2;\mathscr{E})}{2^{nq}}\cdot(2^n-q_x)!\cdot(2^n-q_y)!}{(2^n!)^2} = \frac{\mathsf{NonEQ}(\mathscr{R}_1,\mathscr{R}_2;\mathscr{E})}{2^{nq}(2^n)_{q_x}(2^n)_{q_y}}.$$

We will use the obvious lower bound for $\mathsf{NonEQ}(\mathscr{R}_1,\mathscr{R}_2;\mathscr{E})$ and we obtain

$$\Pr[\Theta_{\mathsf{real}}=\tau] \geq \frac{(2^n)_{q_x}(2^n-\xi)_{q_y}}{2^{nq}(2^n)_{q_x}(2^n)_{q_y}} = \frac{1}{2^{nq}}\cdot\frac{(2^n-\xi)_{q_y}}{(2^n)_{q_y}}.$$

We can immediately see that

$$\frac{\Pr[\Theta_{\mathsf{real}}=\tau]}{\Pr[\Theta_{\mathsf{ideal}}=\tau]} \geq \frac{(2^n-\xi)_{q_y}}{(2^n)_{q_y}}.$$

We can further reformulate the expression $(2^n-\xi)_{q_y}/(2^n)_{q_y}$ to

$$\frac{(2^n-q_y)(2^n-q_y-1)\cdots(2^n-q_y-(\xi-1))}{(2^n)(2^n-1)(2^n-2)\cdots(2^n-(\xi-1))} = \prod_{i=0}^{\xi-1}\frac{2^n-i-q_y}{2^n-i}.$$

This can be reformed to and upper bounded by

$$\prod_{i=0}^{\xi-1}\left(1-\frac{q_y}{2^n-i}\right) \geq \left(1-\frac{q}{2^n-\xi}\right)^{\xi} \geq 1-\frac{\xi\cdot q}{2^n-\xi},$$

where the final inequality is Bernoulli's. ■

6.5.3 Using d-wise Independent Hash Functions

In contrast to the analysis of HPxNP, for HPxHP, we find ξ not only in the analysis of bad_1, but also in that of bad_5 plus in the bound for the good transcripts. For the same reasons as in HPxNP, bad_1 and bad_5 cap the bound at around $q=2^{2n/3}$. Using the average block size would not work here since it would not affect the bound of bad_5. However, we can increase the security bound of HPxHP with stronger, d-wise independent hash functions. For even d, this allows to obtain a bound of $q=2^{dn/(d+1)}$ since such hash functions yield better bounds for circles of sizes $\geq d$. Since circles always contain an even amount of queries, there would be no benefit of an uneven values d. Leurent et al. required a 4-circle that is expected after $2^{3n/4}$ queries for their attack. Using a 4-independent hash function, the first 4-circle occurs after 2^n queries on average. So, we can obtain a security bound that exceeds the complexity of Leurent et al.'s attack. For simplicity, we will consider 4-wise independent hash functions first and illustrate the changes to the security bound of

HPxHP. Thereupon, we extend our analysis to larger values of d.

Lemma 6.7

Let $\mathscr{H}_1$ and $\mathscr{H}_2$ be independent 4-wise independent hash functions. Let $\xi \geq 7$. Then

$$\Pr[\mathsf{bad}_1 | \neg \mathsf{bad}_2] \leq \frac{2\binom{q}{4}}{2^{3n}\binom{\xi}{4}} + \frac{16q^5}{2^{4n}}.$$

Proof. The analysis of the maximal block size for HPxHP is a little more delicate than that of HPxNP, because we can have collisions on either side, i.e. in the inputs of π_1 or in the inputs of π_2. We will aim to bound the probability of blocks of size 7 among the queries, i.e., $\{(u_{i_1}, v_{i_1}), \ldots, (u_{i_7}, v_{i_7})\}$, for pairwise distinct $i_1, \ldots, i_7 \in \{1, \ldots, q\}$. For simplicity, we reindex them as $\{(u_1, v_1), \ldots, (u_7, v_7)\}$, hereafter. W.l.o.g., we consider them in an order s. t. $u_i = u_{i+1}$ or $v_i = v_{i+1}$ holds for each $1 \leq i < 7$. We exclude collisions of the form $(u_i, v_i) = (u_j, v_j)$ since those are already covered by bad_2.

For such blocks, we consider sub-blocks of 5 queries (our actual interest) and upper bound their probability. However, not in all cases, we can obtain a satisfying bound; therefore, we will consider 7-blocks at some points. We identify all possible collision patterns and bound their probability accordingly before we can make a final statement on the maximal block size.

The left side of Figure 6.2 illustrates the possible patterns of 5-chains. We can encode the possible hash-collisions patterns by four-bit strings (a_1, a_2, a_3, a_4), where $a_i = 0$ if $u_i = u_{i+1}$ and $a_i = 1$ if $v_i = v_{i+1}$. It is easy to see that we can obtain at most 16 such patterns indexed from $(0000) = 0$ through $(1111) = 15$. Moreover, the Variants (8) through (15) are symmetric to their counterparts (0) through (7). So, it suffices to bound the probability of the latter. Our claim follows.

Variant (0): $u_1 = u_2 = u_3 = u_4 = u_5$. 4-wise independence unfortunately does not allow a better bound than $q^4/2^{3n}$ for this case. Instead, we allow large collisions in one hash function as long as they are not connected to collisions in the other hash function. This will allow us to bound the probability of large blocks as we did in the analyses before.

For a single hash function, assume that the largest block has size of ξ. This block contains $\binom{\xi}{4}$ 4-collisions. Let $\#\mathsf{4Colls}_1(q)$ denote a random variable for the number of 4-collisions in the outputs of h_1. Again, Markov's Inequality allows us to upper bound the probability that there are more than $\binom{\xi}{4}$ 4-collisions in one hash function by:

$$\Pr\left[\#\mathsf{4Colls}_1(q) \geq \binom{\xi}{4}\right] \leq \frac{\mathbb{E}[\#\mathsf{4Colls}_1(q)]}{\binom{\xi}{4}} = \frac{\binom{q}{4}}{2^{3n} \cdot \binom{\xi}{4}} \approx \frac{q^4}{2^{3n} \cdot \xi^4}.$$

For $\xi = 2^{n/10}$, this term allows for up to $2^{17n/20}$ queries while the condition $q \cdot \xi^2 = 2^n/67$

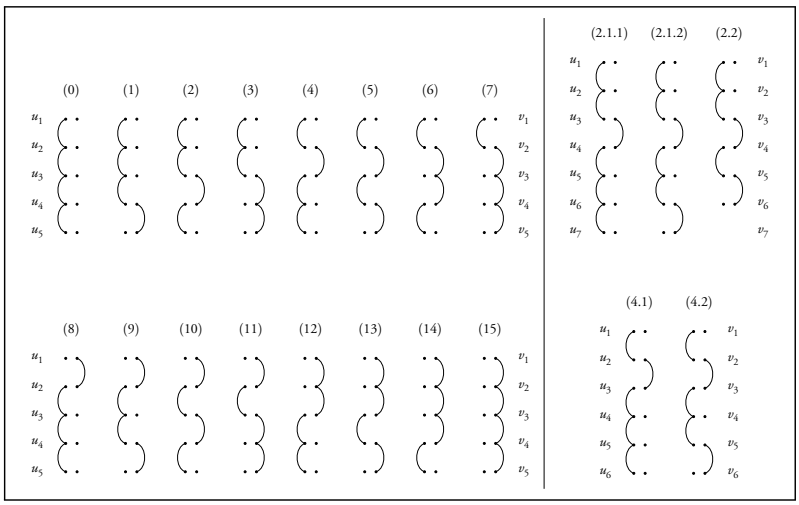

Figure 6.2: Structure graphs of hash-value pairs (u_i, v_i) in blocks of size 5 – 7. Each pair of horizontal dots denotes a pair (u_i, v_i). An edge describes that two hash values are equal, e.g., Variant (2) represents the case that $u_1 = u_2 = u_3$, $v_3 = v_4$, and $u_4 = u_5$.

is fulfilled for up to $\mathcal{O}(2^{4n/5})$ queries. We can derive a similar argument using a random variable $\#\mathsf{4Colls}_2(q)$ for the number of 4-collisions in the outputs of h_2, So, the probability to obtain a block of size ξ in this case is also approximately at most $2q^4/2^{3n}\xi^4$. In the remainder, we will show that we can upper bound the probability of blocks to a size of $\xi \geq 7$ if they connect collisions in h_1 with collisions in h_2 with a probability of $q^5/2^{4n}$.

Variant (1): $u_1 = u_2 = u_3 = u_4$ and $v_4 = u_5$. From 4-wise independence, it holds that the probability for $u_1 = \ldots = u_4$ is at most

$$\sum_{u_1 \in \{0,1\}^n} \Pr\left[h_1(m_1) = h_1(m_2) = h_1(m_3) = h_1(m_4) = u_1\right] \cdot \Pr\left[v_4 = v_5\right]$$
$$\leq (2^n \cdot 2^{-4n}) \cdot 2^{-n} = 2^{-4n}.$$

Since there are at $\binom{q}{5}$ such 5-tuples, this variant has probability at most $q^5/2^{4n}$. An analogous argument can be formulated for Variant (7). For their complexity, we will consider variants (2) and (4) at the end, and proceed with Variant (3) next.

Variant (3): $u_1 = u_2 = u_3$, $v_3 = v_4 = v_5$. From 4-wise independence, it holds

$$\sum_{u_1 \in \{0,1\}^n} \sum_{u_4 \in \{0,1\}^n} \Pr\left[h_1(m_1) = h_1(m_2) = h_1(m_3) = u_1, h_1(m_4) = u_4\right]$$
$$\leq 2^{2n} \cdot 2^{-4n} = 2^{-2n}.$$

Since the outputs of h_2 are independent from h_1, it holds independently

$$\sum_{v_3\in\{0,1\}^n}\sum_{v_2\in\{0,1\}^n}\Pr\left[h_2(m_3)=h_2(m_4)=h_2(m_5)=v_3, h_2(m_2)=v_2\right]$$
$$\leq 2^{2n}\cdot 2^{-4n}=2^{-2n}.$$

We obtain that the upper bound on the probability of this variant is 2^{-4n} for a fixed 5-tuple, and at most $q^5/2^{4n}$ over all such 5-tuples.

Variant (5): $u_1=u_2,\ v_2=v_3,\ u_3=u_4,\ v_4=v_5$. From our assumption that bad_2 is not set, it holds that $u_1\neq u_4$ and $v_2\neq v_5$. Since h_1 and h_2 are independent, it holds that the probability for this constellation is at most

$$\sum_{u_1\in\{0,1\}^n}\sum_{u_4\in\{0,1\}^n}\Pr\left[h_1(m_1)=h_1(m_2)=h_1(m_3)=u_1, h_1(m_4)=u_4\right]$$
$$\cdot\sum_{v_2\in\{0,1\}^n}\sum_{v_4\in\{0,1\}^n}\Pr\left[h_2(m_2)=h_2(m_3)=v_2, h_2(m_4)=h_2(m_5)=v_4\right]$$
$$\leq(2^{2n}\cdot 2^{-4n})\cdot(2^{2n}\cdot 2^{-4n}),$$

and therefore at most $q^5/2^{4n}$ over all 5-tuples.

Variant (6): $u_1=u_2,\ v_2=v_3=v_4,\ u_4=u_5$. Again, from our assumption that bad_2 is not set, it holds that $u_1\neq\{u_3,u_4\}$ and $v_2\notin\{v_1,v_5\}$. Since h_1 and h_2 are independent, it holds that the probability for this constellation is at most

$$\sum_{v_1\in\{0,1\}^n}\sum_{v_2\in\{0,1\}^n}\Pr\left[h_2(m_1)=v_1, h_2(m_2)=h_2(m_3)=h_2(m_3)=v_2\right]$$
$$\cdot\sum_{u_1\in\{0,1\}^n}\sum_{u_4\in\{0,1\}^n}\Pr\left[h_1(m_1)=h_1(m_2)=u_1, h_1(m_4)=h_1(m_5)=u_4\right]$$
$$\leq(2^{2n}\cdot 2^{-4n})\cdot(2^{2n}\cdot 2^{-4n}),$$

and therefore at most $q^5/2^{4n}$ over all 5-tuples.

Variant (2): $u_1=u_2=u_3,\ v_3=v_4,\ u_4=u_5$. While we could upper bound

$$\Pr\left[u_2=u_3, v_3=v_4, u_4=u_5\right]\leq q^4\cdot 2^{-3n}$$

in a straight-forward manner for this constellation, it would be inferior to our desired bound. Hence, we extend it further to six-query variants (2.1), where we add the condition $h_1(m_5)=h_1(m_6)=u_5=u_6$; and (2.2), where we add $h_2(m_5)=h_2(m_6)=v_5=v_6$. One can observe that constellation (2.2) contains Variant (5). So, the probability for the subset of queries $(m_2,\ldots,m_6)$ to form the collisions as shown can be derived from there to be at

most $q^5/2^{4n}$ over all such 5-tuples.

Since we cannot find a good bound for (2.1) yet, we extend it further. We define Variant (2.1.1) to add a seventh query to the block such that $h_1(m_6) = h_1(m_7)$. From 4-wise independence, it holds that the probability for $u_4 = \ldots = u_7$ is at most

$$\begin{aligned}&\sum_{u_4 \in \{0,1\}^n} \Pr\left[h_1(m_3) = h_1(m_4) = h_1(m_5) = h_1(m_6) = u_4\right] \cdot \Pr\left[v_3 = v_4\right] \\ &\leq (2^n \cdot 2^{-4n}) \cdot 2^{-n} = 2^{-4n}.\end{aligned}$$

So, the probability for Variant (2.1.1) is at most $q^5/2^{4n}$. For Variant (2.1.2), we can observe that it contains Variant 9, which is axially symmetric to Variant 6. Thus

$$\begin{aligned}&\sum_{u_3 \in \{0,1\}^n} \sum_{u_4 \in \{0,1\}^n} \Pr\left[h_1(m_3) = u_3, h_1(m_4) = h_1(m_5) = h_1(m_6) = u_4\right] \\ &\cdot \sum_{v_3 \in \{0,1\}^n} \sum_{v_6 \in \{0,1\}^n} \Pr\left[h_2(m_3) = h_2(m_4) = v_3, h_2(m_6) = h_2(m_7) = m_6\right] \\ &\leq (2^{2n} \cdot 2^{-4n}) \cdot (2^{2n} \cdot 2^{-4n})\end{aligned}$$

and therefore at most $q^5/2^{4n}$ over all 5-tuples $(m_3, \ldots, m_7)$. So, for all extensions, the probability of a 7-query block from Variant (2) is upper bounded by $q^5/2^{4n}$.

Variant (4): $u_1 = u_2$, $v_2 = v_3$, $u_3 = u_4 = u_5$. By relabeling the indices of the queries we can see that this variant is the same as Variant (2). ■

We find two interesting points here: (1) Raising the requirement of the hash functions to 4-wise independence yields a 4-circle after 2^n queries on average instead of after $2^{3n/4}$ queries as in the attack by Leurent et al.. Thus, a security level of $2^{4n/5}$ can be obtained. (2) We cannot show yet if it is possible to consider ξ_{average} instead of $\xi_{\max}$. If we can consider the average block size instead of the maximum block size, the upper bound of circles is the bottleneck. Vice versa, it seems that attacks on the HPxHP-type of MACs must exploit the occurrence of circles. We can formulate the following lemma to bound the probability of bad_5.

Lemma 6.8

Let $\mathscr{H}_1$ and $\mathscr{H}_2$ be independent 4-wise independent hash functions. Then

$$\Pr\left[\mathsf{bad}_5 | \neg\mathsf{bad}_2 \wedge \neg\mathsf{bad}_1\right] \leq q^4/2^{4n}.$$

Proof. The analysis of bad_5 can then be conducted as follows, where we restrict our attention to bad_5 conditioned on $\neg\mathsf{bad}_2$ and $\neg\mathsf{bad}_1$. So, we concern chains of even lengths, such that no collisions $(u_i, v_i) = (u_j, v_j)$ has occurred. Hence, bad_2 already covers the

probability of 2-chains. A 4-chain is a 4-tuple of pairwise disjoint query indices (i_1, i_2, i_3, i_4) such that there exists an ordering of the indices s. t. $h_1(m_{i_1}) = h_1(m_{i_2})$, $h_2(m_{i_2}) = h_2(m_{i_3})$, $h_1(m_{i_3}) = h_1(m_{i_4})$, and $h_2(m_{i_4}) = h_2(m_{i_1})$ hold. For simplicity, we reindex those queries to $(1, 2, 3, 4)$ and reindex their corresponding hash values. It holds that

$$\begin{aligned}
&\sum_{u_1 \in \{0,1\}^n} \sum_{u_3 \in \{0,1\}^n} \Pr\left[h_1(m_1) = h_1(m_2) = u_1, h_1(m_3) = h_1(m_4) = u_3\right] \\
&\cdot \sum_{v_1 \in \{0,1\}^n} \sum_{v_2 \in \{0,1\}^n} \Pr\left[h_2(m_1) = h_2(m_4) = v_1, h_2(m_2) = h_2(m_3) = v_2\right] \\
&\leq \left(2^{2n} \cdot 2^{-4n}\right)^2 = 2^{-4n},
\end{aligned}$$

and over $\binom{q}{4}$ such tuples, we obtain an upper bound of $q^4/2^{4n}$.

A 6-chain is a 6-tuple of pairwise disjoint query indices $(i_1, \ldots, i_6)$ such that there exists an ordering of the indices s. t. $h_1(m_{i_1}) = h_1(m_{i_2})$, $h_2(m_{i_2}) = h_2(m_{i_3})$, $h_1(m_{i_3}) = h_1(m_{i_4})$, $h_2(m_{i_4}) = h_2(m_{i_5})$, $h_1(m_{i_5}) = h_1(m_{i_6})$, and $h_2(m_{i_6}) = h_2(m_{i_1})$. Again, we simply reindex them to $(1, \ldots, 6)$. One can observe that there is a sub-structure that corresponds to Variant (5) in our proof of Lemma 6.7. By conditioning on $\neg\mathsf{bad}_1$ we do not need to add this term to the above bound. It is easy to see that every chain of eight or more queries must contain at least one of those sub-structures and can be bounded accordingly. Our claim in Lemma 6.8 follows. ■

6.5.4 Extension to d-independence for Even d

Lemma 6.9

Let $\mathscr{H}_1$ and $\mathscr{H}_2$ be independent d-wise independent hash functions. Then

$$\Pr[\mathsf{bad}_1 | \neg\mathsf{bad}_2] \leq \frac{2\binom{q}{d}}{2^{(d-1)n} \cdot \binom{\xi}{d}} + \frac{q^{d+1}}{2^{(n-1)d}}.$$

We can extend the argument above to obtain a security of up to $\mathcal{O}(2^{\frac{(n-1)d}{d+1}})$ queries with a d-independent family of hash function. Instead of considering 5-collisions as the base in the proof of Lemma 6.7, we consider d-collisions in the following. We can index all such $(d+1)$-collision patterns by a d-bit string $(x_1, \ldots, x_d)$ of d variables. Again, each string denotes a collision pattern between a d-tuple of disjoint queries with indices $(i_1, \ldots, i_d)$; So, each bit $x_i = 0$ represents that $h_1(m_i) = h_1(m_{i+1})$ and $x_i = 1$ indicates that $h_2(m_i) = h_2(m_{i+1})$ holds. We denote $h_1(m_i) = u_i$ and $h_2(m_i) = v_i$, for $1 \leq i \leq d$. Again, we exclude cases where $(u_i, v_i) = (u_j, v_j)$ for $i \neq j$.

The probability of almost all collision patterns from such d-bit strings can be easily upper bounded by $q^{d+1}/2^{nd}$. Since we have at most 2^d such patterns, we can upper bound the probability of their union by $2^d q^{d+1}/2^{nd} = q^{d+1}/2^{(n-1)d}$.

The only exceptions are represented by the patterns

– $(010\cdots0), (0010\cdots0), \ldots, (0\cdots010)$,

– and their counterparts $(101\cdots1), (1101\cdots1), \ldots, (1\cdots101)$.

Hence, we will allow these ***bad*** collision patterns and consider extensions thereof. We focus on those bit strings with hamming weight one since an analog argument holds for their counterparts.

Extending one of those weight-one patterns by a 1-bit on either side will produce a subpattern that has already been excluded. Moreover, we can extend any of the weight-one patterns above to a string of $d-2$ zeros followed by a 1 followed by another $d-2$ zeros. This extended $2(d-1)$-bit string encodes a collision pattern between $2d-1$ queries and is still allowed. However, beyond this point, any further extension will yield an excluded subpattern. Hence, the maximal block size for blocks connecting collisions on the left side with collisions on the right side is $2d$. We define the 0-1-variable $\#\mathsf{dColls}_1(q)$ to be 1 if there exists a chains of d-collisions of hashes from a single hash function, that are not connected to collisions in the second hash functions. Clearly, it can be upper bounded from a similar argument as before for 4-wise independent hash functions:

$$\Pr\left[\#\mathsf{dColls}_1(q) \geq \binom{\xi}{d}\right] \leq \frac{\mathbb{E}[\#\mathsf{dColls}_1(q)]}{\binom{\xi}{d}} = \frac{\binom{q}{d}}{2^{(d-1)n}\cdot\binom{\xi}{d}} \approx \frac{q^d}{2^{(d-1)n}\cdot\xi^d}.$$

Lemma 6.10

Let $\mathscr{H}_1$ and $\mathscr{H}_2$ be independent d-wise independent hash functions. Then

$$\Pr\left[\mathsf{bad}_5 | \neg\mathsf{bad}_2 \wedge \neg\mathsf{bad}_1\right] \leq \sum_{i=1}^{d/2}\left(\frac{q}{2^n}\right)^{2i}.$$

It remains to exclude circles up to a size of d. Larger circles are excluded by the pattern $(010\ldots010)$. Circles up to a size of d can be excluded by $\sum_{i=1}^{d/2}\left(\frac{q}{2^n}\right)^{2i}$.

6.6 Conclusion

We presented two MAC constructions that are provably secure to up to $\mathcal{O}(2^{2n/3})$ queries; HPxHP avoids nonces at the price of two independent hash-function evaluations; HPxNP trades one hash-function call for the use of a nonce.

Our results add to the works that demonstrate the usefulness of Patarin's Mirror Theory for such constructions. We indicated that considering the average instead of the maximal block size in the Mirror Theory would greatly increase the security of one of our construc-

tions. Though, a deeper study of Patarin's theory is required to derive the consequences of this replacement, which is out of the scope of this work.

Leurent et al.'s generic distinguisher on constructions similar to HPxHP with a data complexity of $\mathcal{O}(2^{3n/4})$ queries exploited the occurrence of circles in the underlying hash functions. We studied that stronger, d-wise independent hash functions decreased the probability of circles where we indicate that it can raise the security level above the bound of $\mathcal{O}(2^{3n/4})$.

For HPxNP, Choi et al. [CLLL20] later established an improved bound of $\mathcal{O}(2^{3n/4})$. Dutta and Nandi [DN20] examined the construction involving public permutations. This was further extended to a multi-user setting by Chen et al. [CDN22].

Fo HPxHP the lower bound was improved to $\mathcal{O}(2^{3n/4})$ by Kim et al. [KLL20]. The construction's security was first evaluated in a multi-user environment and a bound of $\mathcal{O}(2^{2n/3})$ was shown by Shen et al. [SWGW21], then later enhanced to $\mathcal{O}(2^{3n/4})$ by Datta et al. [DDNT23]

Part III

Stream Ciphers

7 | Stream Ciphers

In this chapter, we will first give a general overview of stream ciphers, present different types, their security requirements, and vulnerabilities. As we will mostly be concerned with hardware-based stream ciphers, we give a brief overview of feedback shift registers as these are commonly used in stream cipher designs.

Block ciphers vs. stream ciphers. The symmetric encryption of a plaintext is usually performed using block ciphers or stream ciphers. The most commonly used block cipher, AES, operates on blocks of 128 bits in size. The inputs to a block cipher are a plaintext block and a secret key that has been communicated between the legitimate parties using a key exchange protocol. If the plaintext consists of more than one block, a mode of operation defines how to repeatedly apply the block cipher to the plaintext.

Stream ciphers, on the other hand, generate a stream of key bits from a secret key and a public variable called the initial value, initialization vector, or just nonce. This pseudorandom keystream is then combined, usually via the bitwise exclusive-or operation, with the plaintext to produce the ciphertext.

Stateful vs. counter-based. Counter-based stream cipher designs produce a block of keystream bits from a counter value. As an example, one may use the counter mode of operation for block ciphers to encrypt the counter and use the block cipher's output as a keystream block, increment the counter, and repeat the process. Stateful stream ciphers, on the other hand, maintain an internal state from which the key bits are generated. That internal state is updated before the next key bits are output.

Hardware-based stream ciphers. Hardware-based stream ciphers commonly rely on feedback shift registers (FSRs). These FSRs maintain and update an internal state within their register cells [HJM06, CP05, AGH18, BCI+21, AM15, MAM16]. In this work, we are not concerned with FSRs in detail and refer the interested reader to [Aum17] and [Kle13]. One objective in designing secure hardware-based stream ciphers is to create an efficient design. In particular, it is important to keep the register size small and have a low power demand. This will allow reaching a higher security level for resource-constrained devices and thus ultimately allow more devices to communicate securely.

Ongoing research is currently centered on exploring designs that leverage available external resources outside of a cipher's hardware module. These resources may be the secret key or the secret IV. The wiring is typically already available to allow the loading of these values into the cipher's hardware module before keystream generation. In contemporary designs, the available wiring is utilized not only during the initial loading of values into the cipher's hardware module but also throughout the process of keystream generation.

Software-based stream ciphers. There are software-based stream ciphers as well. Their goal is to achieve high throughput rates by efficiently using the instructions provided by modern-day processors. In this work, we will not be dealing with software-based stream ciphers and focus instead on hardware-based stream ciphers and means to create efficient designs.

Contribution. The contribution of this work is twofold. First, a random oracle model for stream ciphers is proposed in which various designs are analyzed for their security against generic attacks. Second, based on the analysis in the random oracle model, a new lightweight stream cipher design called Draco is proposed. In this work, we present Draco in the original version as accepted at FSE 2023 [HMKM22]. A glitch in the key schedule of Draco was found by Banik [Ban22]. In a later chapter, we will present current work-in-progress describing how to fix the key schedule.

The eSTREAM project. The eSTREAM project, initiated by the European Union, aims to identify stream ciphers that are well-suited for widespread adoption. It distinguishes between software-based and hardware-based stream ciphers within two profiles [ECR08]:

Profile 1. Stream ciphers for software applications with high throughput requirements.

Profile 2. Stream ciphers for hardware applications with restricted resources such as limited storage, gate count, or power consumption.

The project's goal for hardware-based ciphers, in particular, is suitability for resource-constrained devices while still maintaining a high security level. They further note that for 'any profile it is likely that the stream cipher must be demonstrably superior to the AES in at least one significant aspect'.

The eSTREAM project does not standardize stream ciphers but pools information and makes them available to implementors.

Vulnerabilities. One goal of an adversary typically is to recover some unknown plaintext bits. For stream ciphers, this is often done by recovering an internal state, as the internal state ultimately determines the keystream. A known internal state allows one to generate further key bits and thus decrypt ciphertext and recover plaintext. A different adversarial goal is to distinguish the cipher from a truly random bitstream. Proofs of security

often show that an adversary cannot distinguish a cipher's output from random noise, as *indistinguishability* implies resistance against recovery of plaintext bits. Apart from being a stronger notion of security, indistinguishability also is often easier to prove.

Stream ciphers are vulnerable to an attack type called *time-memory-data tradeoff* (TMDTO) attacks, introduced by Babbage [Bab95] and Golić [Gol96]. These attacks exploit the birthday paradox to recover an internal state in data D and time T, satisfying $TD = 2^{\ell_s}$, where ℓ_s denotes the internal state length. The TMDTO attacks introduced by Babbage and Golić are generic, meaning they need not exploit any internals of the cipher. Thus, the conventional design approach for stream ciphers was to design them with an internal state length of at least twice the desired security level.

Challenges. Hardware-based stream ciphers are meant to be efficient, i.e., for the given security level, their power consumption and area requirements are supposed to be low. Due to their vulnerability to TMDTO attacks, stream ciphers need a large internal state. Keeping and updating a large amount of register cells is costly, and thus the number of state cells needs to be carefully adjusted.

One of the common concerns is the linearity of the update functions. Nonlinear functions are considered more secure, but they often come with increased complexity and resource requirements. Additionally, as the complexity of the functions increases, ensuring the absence of bias becomes more challenging. Furthermore, weaknesses in the key schedule may lead to a break of the cipher as happened with Draco in its original version as presented at FSE 2023 [HMKM22, Ban22].

Popular designs. Salsa20 [Ber05a] or its variant ChaCha20 [Ber08] are popular software-based stream ciphers. They are used in libsodium [lib23], OpenSSL [Ope16], Shadowsocks [Sha15], SSH [Ope14], and WireGuard [Wir23], respectively.

The use of hardware-based stream ciphers is harder to identify. Hardware manufacturers often keep the specification of their used ciphers secret. [Aum17] notes that:

> Grain-128a is used in some low-end embedded systems that need a compact and fast stream cipher – typically industrial proprietary systems – which is why Grain-128a is little known the the open-source software community.

Hardware-based vs. software-based designs. Hardware-based stream ciphers run, as the name implies, on dedicated hardware. Requirements are to use resources as efficiently as possible to be implementable on a wide range of devices. This includes power consumption, which matters in particular for passively powered devices, as no keystream will be produced if the magnetic field is too weak for the device. The energy consumption plays an important role for battery-powered devices and will thus have a direct impact on the battery life. Area also impacts the above, as more circuitry will have more demand for

power and energy. Also, less area will make designs easier to integrate into existing hardware designs, and it will also incur fewer monetary costs.

Software-based stream ciphers run on CPUs and make use of their word size and available instructions to run as efficiently as possible. The goal is typically to maximize the throughput while using as little CPU cycles as possible. These are especially interesting for general-purpose CPUs.

Lightweight designs. Small low-powered devices, such as RFIDs, only have very limited resources. It is important to use those resources as efficiently as possible to reach a high security level. In particular, TMDTO attacks limit the security of conventional stream ciphers, as their internal state needs to be at least twice the size of the desired security level. Conversely, if the state size is limited, so is the security level. In the upcoming chapters, we wish to overcome these limitations by analyzing novel paradigms for stream cipher designs.

As an introductory motivation to stream ciphers, we begin this chapter by explaining the one-time pad.

7.1 One-time Pad

We already gave an overview of the one-time pad in Subsection 3.1.1. We will briefly recap the one-time pad here, as it is particularly relevant for stream ciphers. Stream ciphers mimic the one-time pad. Instead of a large key, a stream cipher uses a small key to generate a large pseudorandom bitstream. This bitstream, in this context called the *keystream*, is then added to the plaintext as is done with the one-time pad.

Encryption. The one-time pad is a cipher that is known to be perfectly secure. To encrypt a plaintext message m of length ℓ, a secret key k of the same length ℓ is needed. First, the secret key is generated and exchanged between the legitimate parties via a secure channel. The ciphertext c is generated by adding the secret key k to the plaintext message m:

$$c := \mathsf{Enc}(k, m) = m \oplus k.$$

Similarly, a decryption can be performed by adding the secret key k to the ciphertext c:

$$m := \mathsf{Dec}(k, c) = c \oplus k.$$

The one-time pad has one major disadvantage: The key length is the same size as the message length and has to be communicated between the two parties in advance of their interaction via a secure channel. This may not always be feasible. Instead of using a perfectly random key (stream), stream ciphers generate a pseudorandom keystream from a small seed.

7.2 High-level Stream Cipher Encryption

A keystream generator is a fundamental part of a stream cipher. The keystream generator generates a pseudorandom keystream s from a seed value, which typically consists of a secret key k and an initial value x. The stream cipher then adds, usually via the bitwise exclusive-or operation, this pseudorandom keystream s to the plaintext to obtain the ciphertext. Here, the secret key and the initial value are small ($\approx$ 100 bits each) and can easily be communicated using one of the popular key exchange protocols. The pseudorandom keystream s that results as the output of the stream cipher is much larger than its inputs and is typically in the order of up to 2^{100} bits.

Encryption. Consider a keystream generator $\mathsf{KSG}(k, x)$ with inputs k and x. For simplicity, consider a message m of length ℓ. KSG generates the keystream $\mathsf{KSG}(k, x) = s = s_0 s_1 \dots s_{\ell-1}$ from the secret key k and the initial value x. Encryption is performed as follows:

$$c := \mathsf{Enc}(k, x, m) = m \oplus \mathsf{KSG}(k, x) = m \oplus s.$$

Similarly, a decryption can be performed by adding the pseudorandom keystream s to the ciphertext c:

$$m := \mathsf{Dec}(k, x, c) = c \oplus \mathsf{KSG}(k, x) = c \oplus s.$$

Note that, once the keystream was generated, encryption and decryption are performed identically to the one-time pad.

7.2.1 Keystream Generation Using Stateful Stream Ciphers

Throughout this work we will only consider stateful stream ciphers. We recognize five steps in generating the pseudorandom keystream using a stateful stream cipher:

1. **Key agreement.** A secret key k has to be generated and exchanged between the two legitimate parties. We will ignore this part as it is up to a key exchange protocol and we will assume the legitimate parties have securely exchanged their keys in advance.

2. **Loading.** The secret key, the IV and possibly some constants are loaded into the register, resp. internal state, before any keystream is generated. We will typically ignore loading and start with the loading state.

3. **Initialization.** An initialization function the computes the initial state, i.e. the state from which the first bit is output, from the loading state. Its task is to apply confusion and diffusion the the loading state. This is also referred to as *mixing*.

4. **State update.** An update function updates one internal state to the next internal state.

5. **Output.** An output function computes an output bit from an internal state.

After the inputs are loaded into the register cells and the initialization function has been executed, the output function will compute an output bit. The state is then updated, and the output function will compute the next bit, and so on.

We will formalize this in Section 8.1.1 when introducing the different cipher paradigms.

7.3 Feedback Shift Registers

Feedback shift registers are easy to implement in hardware and very efficient. They need only a small number of logical gates and registers. This makes them a good choice for low-power and resource-constrained environments, while ensuring high-speed operations. All three members of the eSTREAM hardware portfolio are based on feedback shift registers.

Feedback Shift Registers are typically used to store the internal state of a stream cipher. A feedback shift register R of degree $|R| = n$ consists of n register cells, each containing one bit. By R_i^t, we denote the i-th cell of the FSR at step t. At each clock tick, the FSR is

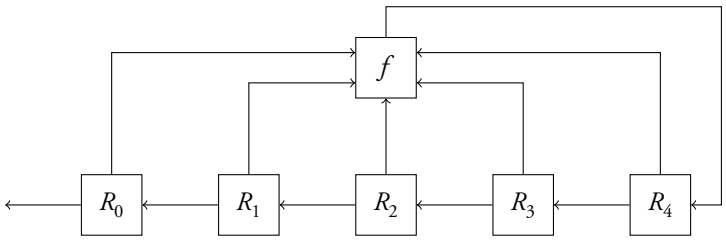

Figure 7.1: A feedback shift register with feedback function f.

updated using a feedback function $f : \{0,1\}^n \to \{0,1\}$ as follows:

$$R_i^{t+1} = \begin{cases} R_{i+1}^t & \text{for } i \in \{1,\ldots,n-1\}, \\ f(R^t) & \text{for } i = n. \end{cases}$$

One typically distinguishes linear from nonlinear feedback functions. A general form of an FSR can be found in Figure 7.1.

Linear feedback shift registers. *Linear feedback shift registers* (LFSRs) are feedback shift registers with a linear feedback function f. LFSRs can be implemented very efficiently in hardware but are not by themselves good pseudorandom generators. Nonetheless, they can be used as a building block in stream ciphers, particularly because it is known how to construct LFSRs with a maximal period of $2^n - 1$. Note that the all-zero state must not occur as the LFSR is then forever stuck in this state since the feedback bit will always be zero. The FSR in Figure 7.2 shows the feedback function $f(R^t) = R_0^t \oplus R_2^t \oplus R_4^t$. For

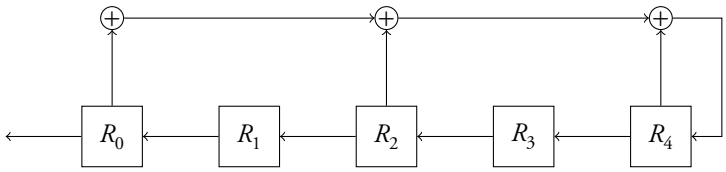

Figure 7.2: A linear feedback shift register.

coefficients $a_i \in \{0,1\}$, the feedback function f of an LFSR of degree n can be written as:

$$f(R^t) = \bigoplus_{i=0}^{n-1} a_i \cdot R_i^t.$$

The coefficients of the FSR depicted in Figure 7.2 are $a_0 = a_2 = a_4 = 1$ and $a_1 = a_3 = 0$.

An LFSR of degree n can be described by a polynomial of degree n. Consider the linear feedback function as described above. The feedback polynomial $p(X)$ is defined as follows:

$$p(X) = X^n + a_{n-1}X^{n-1} + \ldots + a_1 X + a_0.$$

Note that FSRs are often indexed in reverse, i.e., the feedback bit replaces the lowest indexed bit, and the output bit is the one with the highest index. We chose the variant by [PP10] as it is consistent with the notation published in [HMKM22]. It is known that the period of the LFSR is maximal *if and only if* the feedback polynomial $p(X)$ is primitive. A primitive polynomial is a special type of irreducible polynomial, i.e., a polynomial that cannot be factored into lower degree polynomials. More information on them and their relationship to FSRs can be found in [Kle13]. The identification of maximum length LFSRs with primitive polynomials makes it easy to find maximum length LFSRs. Given an output sequence, the Berlekamp-Massey algorithm [Ber68, Mas69] finds the shortest length LFSR that generates this sequence.

Nonlinear feedback shift registers. *Nonlinear Feedback Shift Registers* (NFSRs) are feedback shift registers with a nonlinear feedback function f. The advantage over LFSRs is that, due to nonlinear terms in the feedback function f, it is harder to recover the internal state from a sequence of output bits. The major disadvantage is that it is not known how to construct NFSRs with a large period. However, there exist NFSRs with an experimentally verified period. In particular, for small NFSRs, one can compute the period by clocking for up to 2^n cycles and checking the register cells for duplicates.

Filtered feedback shift registers. In the common description of FSRs, the leftmost bit is output. A filtered FSR will use a filter function $g : \{0,1\}^n \to \{0,1\}$ that computes the output based on the current content of the register cells. They are particularly useful for linear FSRs as a nonlinear output function will thwart straightforward attacks to recover the

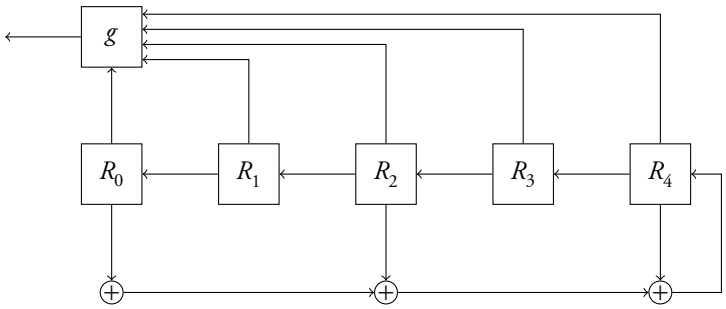

Figure 7.3: A filtered linear feedback shift register.

LFSR's internal state. The output function should be balanced so that the output bitstream will not have any obvious bias. For example, if the output bitstream contains more ones than zeros, it will be easy to distinguish this bitstream from a random bitstream.

Combining feedback shift registers. Another option is to use multiple FSRs, as with Grain-128a [ÅHJM11]. While NFSRs are cryptographically stronger, their properties are also harder to analyze, particularly their period. Combining an NFSR with an LFSR will yield a period at least as long as that of the LFSR. Alternatively, one may also combine multiple NFSRs to take advantage of their nonlinearity and further use one with a guaranteed period to prevent an early repetition of keystream bits.

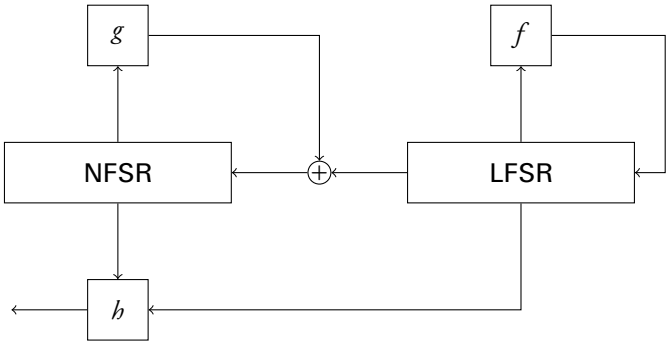

Figure 7.4: A stream cipher with a Grain-like structure.

7.4 Security Requirements

If an attacker manages to generate some bits of the keystream without knowledge of the key, this will allow them to decrypt those bits of the ciphertext. By bitwise addition of those bits to the ciphertext, the plaintext can be obtained.

Attacker's capabilities. The attacker does not know the secret key but is typically allowed to generate keystream for chosen IVs. Chosen-IV attackers imply chosen-plaintext

and chosen-ciphertext attackers since, given the keystream, the attacker can easily XOR the keystream with the plaintext or ciphertext.

Notions of security. As a stream cipher is a deterministic algorithm based on its inputs, knowing the internal state allows the generation of the keystream and thus trivially distinguishes it from randomness. There are typically three different notions of security that are considered regarding stream ciphers:

1. **Key recovery** refers to the adversary's ability to recover the secret session key. This will allow the adversary to decrypt any ciphertext that is encrypted using the secret session key. Security from key recovery attacks is the weakest notion considered for stream cipher security.

2. **Internal state recovery** refers to the adversary's ability to recover an internal state of the stream cipher. Many stream ciphers allow the decryption of an entire packet if an internal state of the stream cipher is known to the adversary. Here, a packet refers to the keystream that is generated for one key-IV pair. Some stream ciphers have an invertible state transition function that will further allow the recovery of the secret key by clocking the cipher back to its loading state.

3. **PRF security** refers to the adversary's ability to distinguish the bitstream generated by a stream cipher from a bitstream generated by a random function. This is the strongest security notion considered. It implies security from key recovery and internal state recovery attacks.

PRF security. PRF security implies the security from key recoveries or internal state recoveries. Of those, it is the strongest notion of security and typically the easiest to prove. Given the key, the adversary can generate any internal state. In conventional stream ciphers, from a given internal state, the adversary can compute the keystream bits, which then allows to distinguish. In our proofs, we will show that the output of the stream cipher resembles that of a random function, and we will show that an adversary is only negligibly better than chance at predicting the keystream bits.

7.5 Time-memory-data Tradeoff Attacks

In this section, we will give a short overview of time-memory-data tradeoff attacks against conventional stream ciphers. By conventional stream ciphers, we refer to stream ciphers without external non-volatile resources. A more thorough presentation will be given in Section 8.2.

Relevance. We are particularly concerned with lightweight cryptography. In particular, we want stream ciphers to be cheaply implementable in hardware. For conventional stream

ciphers, time-memory-data tradeoff attacks imply that security is limited to half of the internal state size. For resource-constrained devices with a small internal state, these attacks present a bottleneck to achieving a higher security level.

The Babbage-Golić attack. Time-memory-data tradeoff attacks are due to Babbage [Bab95] and Golić [Gol97]. The three dimensions, time T, memory M, and data D, are considered. Usually, these types of attacks are further divided into an online and an offline phase. Let ℓ_s be the internal state size.

1. For m randomly and independently chosen internal states, generate a corresponding keystream of length ℓ_s. Store the tuple in an efficiently searchable data structure.
2. Observe d keystream bits and search for a ℓ_s-bit-collision with the keystream generated in the first step.

If $m \cdot d = 2^{\ell_s}$, a collision in the internal state and thus the keystream bits will occur with high probability. In particular, for $m = d = 2^{\ell_s/2}$, the attack has an overall complexity of $\mathcal{O}(2^{\ell_s/2})$. Thus, the security level of conventional stream ciphers is capped at the so-called birthday bound, providing only $\ell_s/2$ bits of security for a state of length ℓ_s.

8 | Enhanced State Stream Ciphers

Conventional stream cipher designs typically have a large volatile internal state. New research directions seek to enhance the volatile state by a non-volatile internal state that may include resources outside of the cipher's hardware module. This will allow shrinking the volatile internal state kept inside the cipher's hardware module, thus allowing for more efficient resource utilization. Furthermore, even if the non-volatile bits are kept within the cipher's hardware module, this will result in lower power consumption as these bits are not updated during the keystream generation.

Stream ciphers. Throughout this work, we consider individual bit outputs. The output bit stream is then combined with the plaintext, usually using the XOR operation. One advantage of stream ciphers is that their resource requirements are lower than those of block ciphers in many application scenarios. This makes them particularly useful in lightweight cryptography. Instances of stream ciphers are used in the GSM cellular phone standard (A5/1), Bluetooth (E0), and wireless networking (RC4).

Vulnerabilities. Stream ciphers are vulnerable to time-memory-data tradeoff attacks [Bab95, Gol96, BS00]. These types of attacks exploit the birthday paradox to recover an internal state. This internal state can then be used to decrypt the remaining ciphertext. Due to the birthday paradox, the security of such ciphers is typically capped at half the size of the internal state. Accordingly, this has influenced the design of stream ciphers in such a way that the internal state size is at least twice the size of the desired security level. This is in stark contrast to the lightweight principle of stream ciphers, as a larger state necessarily increases resource requirements. Stream ciphers that employ a large internal state are the eSTREAM portfolio members Grain [HJM06] and Trivium [CP05]. We refer to these ciphers as the large-state-small-key construction, or LSSK for short.

Conventional versus enhanced designs. Conventional designs keep the entire internal state within the cipher's hardware module. We refer to these as ciphers with a large state and small key, abbreviated as LSSK. Examples of this include the eSTREAM hardware portfolio members Trivium [CP05] and Grain [HJM06]. More recent designs reduce the register size within the cipher's hardware module by accessing an externally available

resource that will *not* be updated during internal state updates. This externally available resource may be the secret key, the IV, or a combination of both stored in EEPROM. Therefore, we make the distinction between a volatile and a non-volatile internal state. We also note that the non-volatile state may also be kept inside the cipher's hardware module. This will not decrease the register size within the cipher's hardware module, but it may decrease power consumption as those register bits do not need to be updated.

Sprout [AM15], Plantlet [MAM16], Fruit [AGH18], and Atom [BCI+21] are stream ciphers that continuously use the non-volatile secret key during keystream generation. We refer to these as CKEY. The work [HKMZ18] describes a distinguishing attack on ciphers that use the non-volatile secret key during state updates, which has a complexity of $2^{\ell_v/2}$, where ℓ_v denotes the size of the *volatile* internal state only. The relevance of this distinguishing attack in practice may be debated, as no internal state recovery or key recovery attack is known. Instead of using the secret key as a non-volatile external resource, one may use the IV [HKM17b], a construction we call CIV. These ciphers use an additional parameter ℓ_p called the packet length, which specifies how many keystream bits may be generated per key-IV pair. It has been shown that these ciphers achieve a security level of $\ell_v - \log(\ell_p)$ bits, i.e., any successful (generic) attack has a complexity of at least $2^{\ell_v}/\ell_p$ [HKM19]. The CIVK construction uses the non-volatile IV with a key prefix of length at least $\log(\ell_p)$ to compensate for the factor ℓ_p^{-1}. CIVK achieves a security level against generic attacks of ℓ_v bits, i.e., any successful (generic) attack has a complexity of at least 2^{ℓ_v}.

Random oracle model. In this work, we propose a random oracle model (ROM) for the stream cipher constructions mentioned above. A random oracle model identifies building blocks within a cryptographic algorithm and replaces those with idealized, i.e., random, primitives. An adversary will have oracle access to these building blocks as well as the construction function in either of two worlds. In the real world, the construction function makes use of the underlying idealized building blocks. In the ideal world, the construction function is an idealized version of the cryptographic algorithm under consideration. In the case of stream ciphers, the idealized version would output a perfectly random keystream as opposed to a pseudorandom one. The adversary's task is to distinguish between these two worlds. We will present upper bounds on the adversary's success probability for all the stream cipher constructions that have been proposed so far.

Similar work was done for block ciphers, initiated by Even and Mansour [EM97]. The Even-Mansour construction and its iterated variant can be seen as an abstraction of block ciphers, particularly AES. The round function of the respective block cipher is replaced by an ideal variant, namely a random permutation. Before every round and after the last round, the secret round key is added. A computationally unbounded, i.e., information-theoretic, distinguisher may then query the construction and the underlying permutations. The distinguisher then has to decide whether it was interacting with the Even-Mansour construction or a truly random permutation. In this model, for r rounds, a lower bound

of security of $\Theta\left(2^{\frac{rn}{r+1}}\right)$ has been shown [CS14], where n denotes the block length. There is a huge body of work done on the iterated Even-Mansour cipher and its iterated variant [CS14, CLL+14, DKS12, LPS12, BKL+12, ABD+13, EM97].

The model is a strong one, as the internals of a cryptographic algorithm are clearly not random. Therefore, proofs in this model will *not* imply the absence of attacks against a concrete scheme but rather show that the interaction of these building blocks is secure. This is particularly interesting for stream ciphers, as time-memory-data tradeoff attacks are generic and do not necessarily exploit the internals of the stream cipher. While these proofs do not demonstrate the security of any specific scheme, they provide a solid foundation on which to base a stream cipher, as opposed to ad-hoc designs without such proofs. From a different perspective, these types of proofs show that any successful attack *must* exploit the internals of the respective construction. Hence, a generic time-memory-data tradeoff attack cannot have a complexity lower than the bounds we provide in this work. We discuss the validity of proofs in the random oracle model more thoroughly in Section 3.7.

We consider the CIVK construction, the constructions considered in [HKM19] (i.e., the conventional construction LSSK and stream ciphers continuously using the CIV), as well as stream ciphers continuously using the secret key CKEY, for which, to the best of our knowledge, a proof of security has not yet been presented. We use an equivalent model to that of [HKM19], and by making use of the H-coefficients technique [Pat08], our proofs of security are significantly simpler than those presented in [HKM19].

Contribution. We propose a random oracle model for the four stream cipher designs mentioned above. We then derive an upper bound on a distinguisher's advantage, respectively a lower bound on the security of the scheme, and thus show the absence of generic attacks with lower complexity than the given bounds. In particular, we confirm the lower bound of half the internal state for conventional stream cipher designs and prove increased security for the enhanced state stream cipher that incorporates the initial value during keystream generation. We note that designs continuously using the secret key during keystream generation offer no benefit over the conventional design with regard to security against distinguishing attacks. Furthermore, we present matching time-memory-data tradeoff attacks on each of the constructions presented in this work, thus showing that the presented bounds are tight.

This work enhances the works [HKM19] and [HMKM22]. [HKM19] provides lower bounds for LSSK and CIV. In their model, the adversary receives a block of keystream per query. This is similar to time-memory-data tradeoff attacks, where one typically searches for collisions between keystream blocks. This makes the analysis more cumbersome, as one has to consider overlapping blocks. In our model, the adversary receives individual bits per query, which more closely models the *stream* of bits. Both models are equivalent. Furthermore, we utilize the H-coefficients technique [Pat08], which allows us to significantly simplify the proof of security for LSSK and CIV.

We show the same bound for CIVK as [HMKM22], yet we distinguish the key length ℓ_k and the volatile state length ℓ_v, whereas in [HMKM22], it is assumed that $\ell_k = \ell_v$. Furthermore, we present a proof of security for CKEY.

One interesting aspect to note is that the enhanced state stream ciphers show that not all bits of the internal state need to be updated. Our proofs consider the state to be split into a non-volatile and a volatile part. However, there is no requirement that the non-volatile internal state may not be kept inside the hardware module.

The DRACO stream cipher. Furthermore, we will present a new stream cipher proposal called Draco. Draco uses a 128-bit volatile internal state and a 128-bit non-volatile internal state. The non-volatile state consists of the initial value with a length of 96 bits and a key prefix with a length of 32 bits.

Draco builds upon the generic CIVK construction. In this case, a time complexity of 2^{128} steps is needed for a successful distinguishing attack. The corresponding attack on CIVK can be found in Subsection 8.2.4, and therefore the bound shown in Section 9.5 is tight.

To the best of our knowledge, it is the first small-state stream cipher that achieves a full 128-bit security level against key-recovery *and* distinguishing attacks. Our main variant of Draco stores the key prefix and the IV externally. In an ultra-lightweight scenario, like RFIDs where the secret key is burned into the device or stored in an EEPROM and the frame counter is used as the IV, Draco needs 23 % less area and 31 % less power than Grain-128a at 10 MHz. The saving in power stems from reduced area requirements but particularly also from the fact that unlike previous ciphers such as Grain-128a, only half of the state bits are constantly updated, thus significantly reducing costly dynamic power consumption.

For high-performance environments, we also present the variant $\text{Draco}_{[\text{KI}]}$ where the secret key and the initial value are stored inside the Draco hardware module while still only using 128 bits during the state update. At a clock speed of 1 GHz, Draco needs about 34 % less energy than Grain-128a. This demonstrates that not all internal state bits need to be constantly updated to achieve a security level of 128 bits. For more details, please refer to Section 10.4.

Draco is a stream cipher that operates in packet mode, meaning there may be up to 2^{32} bits, i.e., 512 MiB, of output keystream per key-IV pair. After reaching this limit, a new IV has to be used. No IV may be used twice. In Subsection 8.1.2, we argue that most transmission protocols use a packet size much lower than 512 MiB, and therefore we consider the packet length as a valid constraint for keystream generation.

Fixing DRACO. [Ban22] presented an attack on the original version of Draco that was presented at FSE 2023 [HMKM22]. We will also present the version from FSE 2023 and discuss possible fixes in Chapter 11, as this is currently a work in progress.

Structure of this part. In this chapter, we provide an overview of stateful stream ciphers and present the enhanced state stream ciphers in Section 8.1. We also present the known TMDTO attacks on the four constructions in Section 8.2.

In Chapter 9, we introduce the random oracle model and the proof technique in Section 9.1. Then, in Sections 9.2 to 9.6, we present the proofs of security.

Chapter 10 presents the Draco stream cipher, and Chapter 11 describes our ongoing work on how to address the key schedule vulnerability identified by Banik [Ban22].

8.1 Enhanced State Stream Ciphers

We recognized loading, initialization, state update, and output as fundamental parts of a hardware-based stream cipher in Section 7.2.1. In this section, we will provide a quick overview of the enhanced state ciphers that we analyze in this work before giving a more detailed description in the next section. Initialization and state update are particularly affected by enhanced state ciphers.

The CKEY, CIV, and CIVK constructions divide the internal state into a volatile part of length ℓ_v and a non-volatile part of length $\ell_{nv} := \ell_s - \ell_v$. The non-volatile memory remains unchanged during state update and state initialization. The structure of the volatile and non-volatile state depends on the construction.

The secret key and the IV may already be available in the hardware, outside of the cipher's hardware module. It may therefore be possible to use these external resources from the cipher's hardware module. In practice, this allows for a reduction in the number of costly volatile register cells. Furthermore, even if kept in the cipher module, not all bits need to be updated, resulting in power consumption savings.

Continuous Key. The first example is the CKEY construction. It uses the non-volatile secret key not only for initialization, as is common, but also during state initialization and keystream generation. This principle underlies the stream cipher proposals Sprout, Plantlet, Fruit, and Atom. However, it was shown in [HKM19] that the resistance of the CKEY construction against generic TMDTO distinguishing attacks is at most $\ell_v/2$.

Continuous IV. The CIV construction was first proposed in [HKM17b]. Contrary to CKEY, it does not use the non-volatile key, but it uses the initial values as the non-volatile part of the internal state. This construction provides a provable security level of $\ell_v - \log_2(\ell_p)$ [HKM19]. It is worth noting that there are currently no stream cipher instantiations based on the CIV construction.

Continuous IV & Key. Our new construction CIVK uses the initial values as part of the non-volatile state, as well as a prefix of length $\log_2(\ell_p)$ of the secret key. Specifically, by using the initial values and a key prefix from non-volatile memory, the volatile memory

is initialized with the key only, with $\ell_v = \ell_k$, where ℓ_k denotes the key length. We also define the non-volatile internal state length ℓ_{nv} to be equal to the volatile state length, i.e., we have $\ell_{nv} = \ell_v = \ell_k$. The IV length ℓ_{IV} is determined by the key and packet length: $\ell_{\mathrm{IV}} = \ell_{nv} - \log_2(\ell_p) = \ell_k - \log_2(\ell_p)$.

As we will prove in this work, this allows us to achieve a security level of the entire volatile internal state length ℓ_v. In the next subsection, we will provide a detailed specification of the CIVK construction. The proof can be found in Section 9.5, and the resulting bound can be found in Corollary 9.4.

8.1.1 Description of the Cipher Constructions

We denote by $\mathcal{Q}_{nv}$ the non-volatile internal state space and by $\mathcal{Q}_v$ the volatile internal state space. An internal state is denoted as $\langle a \mid b \rangle$, where $a \in \mathcal{Q}_{nv}$ represents the non-volatile internal state and $b \in \mathcal{Q}_v$ represents the volatile internal state. This notation is chosen to distinguish internal states from arbitrary tuples. If the non-volatile state a consists of two parts a_1 and a_2, we denote the state as $\langle a_1, a_2 \mid b \rangle$. We hide constants in the notation of internal states as they are mostly irrelevant to the analysis. In our model, they only increase the state size and do not negatively affect security. The key space is denoted by $\mathcal{K}$ and the IV space is denoted by $\mathcal{IV}$. The lengths of the non-volatile internal state space, the volatile internal state space, the secret key, and the IV are denoted by ℓ_{nv}, ℓ_v, ℓ_k, and ℓ_{IV} respectively. The following description defines the keystream generation for the four constructions.

Packet length. LSSK and CKEY do not define a packet length. CIV and CIVK, on the other hand, define a packet length ℓ_p as an additional parameter that limits the number of output bits per key-IV pair. For each key-IV pair $(k, x) \in \mathcal{K} \times \mathcal{IV}$, CIV and CIVK can output up to ℓ_p keystream bits, after which a new IV must be exchanged. For simplicity, we define the packet length for LSSK and CKEY to be the total number of keystream bits that can be generated from a key-IV pair, i.e., $\ell_p = 2^{\ell_v}$.

Enhanced state. Conventional stream cipher designs according to LSSK only use a volatile internal state, i.e., the entire internal state is contained in the cipher's hardware module. For CKEY, CIV, and CIVK, the internal state is divided into a volatile part that is updated during state updates and a non-volatile part that is *not* updated during state updates. The non-volatile state of the enhanced state ciphers is composed as follows:

- CKEY: The non-volatile internal state consists of the secret key k only.
- CIV: The non-volatile internal state consists of the initial value x only.
- CIVK: Same as CIV, but the non-volatile internal state further contains a key prefix k^{pre} of length at least $\log(\ell_p)$.

Loading function. The loading function `load` takes a key-IV pair $(k, x) \in \mathscr{K} \times \mathscr{IV}$ as input and loads it into the registers to produce the *loading state*. We will omit the loading process for the most part of our analysis and start the keystream generation process with the loading state, but including it in our notation is useful.

$$\begin{aligned}
\mathsf{LSSK}\colon & \texttt{load} : \mathscr{K} \times \mathscr{IV} \to \mathscr{Q}_v, (k, x) \mapsto |x, k\rangle, \\
\mathsf{CKEY}\colon & \texttt{load} : \mathscr{K} \times \mathscr{IV} \to \mathscr{Q}_{nv} \times \mathscr{Q}_v, (k, x) \mapsto \langle k \,|\, x\rangle, \\
\mathsf{CIV}\colon & \texttt{load} : \mathscr{K} \times \mathscr{IV} \to \mathscr{Q}_{nv} \times \mathscr{Q}_v, (k, x) \mapsto \langle x \,|\, k\rangle, \\
\mathsf{CIVK}\colon & \texttt{load} : \mathscr{K} \times \mathscr{IV} \to \mathscr{Q}_{nv} \times \mathscr{Q}_v, (k, x) \mapsto \langle x, k^{\mathsf{pre}} \,|\, k\rangle.
\end{aligned}$$

Mixing function. The loading state is used as input to the mixing function p. Its task is to provide the initial state with enough confusion and diffusion for further operation and corresponds to clocking the cipher without producing output bits.

$$\begin{aligned}
\mathsf{LSSK}\colon & p : \mathscr{Q}_v \to \mathscr{Q}_v, |x, k\rangle \mapsto |y\rangle, \\
\mathsf{CKEY}\colon & p : \mathscr{Q}_{nv} \times \mathscr{Q}_v \to \mathscr{Q}_{nv} \times \mathscr{Q}_v, \langle k \,|\, x\rangle \mapsto \langle k \,|\, y\rangle, \\
\mathsf{CIV}\colon & p : \mathscr{Q}_{nv} \times \mathscr{Q}_v \to \mathscr{Q}_{nv} \times \mathscr{Q}_v, \langle x \,|\, k\rangle \mapsto \langle x \,|\, y\rangle, \\
\mathsf{CIVK}\colon & p : \mathscr{Q}_{nv} \times \mathscr{Q}_v \to \mathscr{Q}_{nv} \times \mathscr{Q}_v, \langle x, k^{\mathsf{pre}} \,|\, k\rangle \mapsto \langle x, k^{\mathsf{pre}} \,|\, y\rangle.
\end{aligned}$$

State update. The state update function π updates a (volatile) internal state to the next (volatile) internal state.

$$\begin{aligned}
\mathsf{LSSK}\colon & \pi : \mathscr{Q}_v \to \mathscr{Q}_v, |y\rangle \mapsto |y'\rangle, \\
\mathsf{CKEY}\colon & \pi : \mathscr{Q}_{nv} \times \mathscr{Q}_v \to \mathscr{Q}_{nv} \times \mathscr{Q}_v, \langle k \,|\, y\rangle \mapsto \langle k \,|\, y'\rangle, \\
\mathsf{CIV}\colon & \pi : \mathscr{Q}_{nv} \times \mathscr{Q}_v \to \mathscr{Q}_{nv} \times \mathscr{Q}_v, \langle x \,|\, y\rangle \mapsto \langle x \,|\, y'\rangle, \\
\mathsf{CIVK}\colon & \pi : \mathscr{Q}_{nv} \times \mathscr{Q}_v \to \mathscr{Q}_{nv} \times \mathscr{Q}_v, \langle x, k^{\mathsf{pre}} \,|\, y\rangle \mapsto \langle x, k^{\mathsf{pre}} \,|\, y'\rangle.
\end{aligned}$$

r successive invocations of the state update function π on an internal state $\langle a \,|\, b\rangle$ are denoted by $\pi^r(\langle a \,|\, b\rangle)$, e.g., for three successive invocations we write:

$$\pi^3(\langle a \,|\, b\rangle) = \pi(\pi(\pi(\langle a \,|\, b\rangle))).$$

It is needed that the period of the state update function π is larger than or equal to ℓ_p for the entire internal state space. This means that for any internal state $\langle a \,|\, b\rangle$, the set $\{\langle a \,|\, b\rangle, \pi^1(\langle a \,|\, b\rangle), \ldots, \pi^{\ell_p - 1}(\langle a \,|\, b\rangle)\}$ contains ℓ_p distinct elements.

Output function. The output function f maps an internal state $\langle a \,|\, b\rangle$ to an output bit $z \in \{0, 1\}$.

$$\mathsf{LSSK}\colon f : \mathscr{Q}_v \to \{0, 1\}, |y\rangle \mapsto z,$$

$$\mathsf{CKEY}\colon f : \mathcal{Q}_{nv} \times \mathcal{Q}_{v} \to \{0,1\}, \langle k \,|\, y \rangle \mapsto z,$$
$$\mathsf{CIV}\colon f : \mathcal{Q}_{nv} \times \mathcal{Q}_{v} \to \{0,1\}, \langle x \,|\, y \rangle \mapsto z,$$
$$\mathsf{CIVK}\colon f : \mathcal{Q}_{nv} \times \mathcal{Q}_{v} \to \{0,1\}, \langle x, k^{\mathsf{pre}} \,|\, y \rangle \mapsto z.$$

Keystream generation. Let (k, x) be an arbitrary key-IV pair. Using the functions defined above, we can define the construction function e that corresponds to the keystream generation using the above constructions:

$$e : \mathcal{K} \times \mathcal{IV} \times \{0, \ldots, \ell_p - 1\} \to \{0,1\}, (k, x, r) \mapsto f(\pi^r(p(\mathtt{load}(k, x)))).$$

We consider individual output bits, and e outputs the r-th keystream bit. The entire keystream packet of length ℓ_p for a key-IV pair (k, x) looks as follows:

$$e(k, x, 0) \,||\, e(k, x, 1) \,||\, \cdots \,||\, e(k, x, \ell_p - 1).$$

Note that $\ell_p = 2^{\ell_s}$ for LSSK and CKEY.

8.1.2 Discussion of the Packet Length

The eSTREAM hardware portfolio's three ciphers are all designed to handle potentially very large keystream sequences per key-IV pair. Grain v1 [HJM06], MICKEY 2.0 [BD06], and Trivium [CP05] all use 80-bit keys, and they have different IV lengths of 64 bits, up to 80 bits, and 80 bits, respectively. The authors of Grain v1 do not give an explicit limit on the number of keystream bits that should be generated for each key-IV pair. MICKEY 2.0 limits the amount of keystream bits to 2^{40} per key-IV pair, and Trivium has a limit of 2^{64} keystream bits per key-IV pair. Much smaller packet sizes are used by transmission standards like A5/1, Bluetooth, TLS, and IEEE 802.11 for wireless local area networks.

A5/1. Per key-IV pair, A5/1 can only generate 228 keystream bits. The session key is 64 bits long, and the IV corresponds to 22 bits of the publicly known frame number.

Bluetooth. For the so-called basic rate, Bluetooth packets can only include a maximum of 2790 bits. The Bluetooth cipher E_0 takes a 128-bit session key and uses 26 bits of the master's clock as the packet-specific IV. The master's clock is assumed to be publicly known.

Wireless LAN. In the IEEE 802.11 technical standards, wireless LAN communication is standardized. At most 11454 bytes (i.e., $< 2^{17}$ bits) are encrypted under the same key-IV pair using CCMP as specified by the currently active IEEE 802.11-2020 standard [Ins21].

SSL/TLS. Because SSL/TLS is the foundation of HTTPS, it is crucial for protecting the World Wide Web. In the most recent version, TLS 1.3 [Res18], the maximum amount of

data encrypted under the same key-IV pair is $2^{14} + 2^8$ bytes (i.e., $2^{17} + 2^{11} < 2^{18}$ bits), as long as RC4, which is now forbidden for all TLS versions by RFC 7465 [Pop15], is not used.

No rekeying. We view the packet length as a valid additional parameter in stream cipher design since all of the popular transmission standards mentioned above impose a limit on the number of keystream bits generated per key-IV pair. We also want to emphasize that, once the ℓ_p keystream bit limit is reached, only the IV needs to be changed. *Rekeying* is not necessary. As we argue in the following subsection, a full 2^{ℓ_k} bits can be generated per key if the IV length and the packet length are properly selected.

8.1.3 Discussion of the State Lengths

It is up to the designer of the corresponding cipher to specify the lengths of the volatile and non-volatile states, respectively. The proofs of the four constructions will show how the lengths affect the security. However, we would like to provide a brief discussion in this subsection.

Total state size. As the attack by Babbage [Bab95] and Golić [Gol96] is still applicable without modification, the combined total state length $\ell_s = \ell_{nv} + \ell_v$ must still be twice as large as the desired security level.

IV & packet length. It is important to maintain a balance between the IV length ℓ_{IV} and the packet length ℓ_p such that $2^{\ell_{\text{IV}}} \cdot \ell_p \geq 2^{\ell_k}$. If this condition is not met, the codebook would be smaller than 2^{ℓ_k}, which would allow for a trivial attack with a lower complexity than exhaustive search.

Key prefix size. Usually, the size of the key prefix, k^{pre}, is chosen to be at least $\log(\ell_p)$. This compensates for the factor $\log(\ell_p)$ in the bound of CIV, as a guess of the key prefix is correct with a probability of ℓ_p^{-1}. Minor modifications to the key prefix's size will not significantly impact the security level.

Constants. Constants that may be added for a variety of reasons will not be taken into account in our notation or analysis. They will not have a negative effect on the security level in our analysis because they will only increase the size of the state. In our proofs, we will also distinguish the key length ℓ_k and the IV length ℓ_{IV} from the state lengths ℓ_s, ℓ_v, and ℓ_{nv}. The difference between them may only be due to constants.

8.1.4 Hardware Implications of Continuous IV Access

Now one may argue from a hardware perspective that while the secret key has to be stored anyhow (e.g., also for LSSK stream ciphers such as Trivium, Grain etc.) in order to be

reused with other IVs, this would not be the case for the IV. Hence, at first sight, assuming that the IV is still accessible after state initialization might be considered cheating. However, we do not think that this is the case for many application scenarios. For example, in A5/1 of the GSM standard, the IV used for encrypting a data packet is the respective (sequentially incremented) 22-bit frame number. Hence, any A5/1 device needs some memory containing this frame number anyhow. Similarly, the DECT standard for cordless telephone systems relies on frame numbers as IVs for encryption.

Packet counters. In general, especially for ciphers with small IV spaces, there always has to be a mechanism like a stepwise incremented IV storage to make sure that the same IV is not accidentally used twice under the same secret key. Similarly, in all communication scenarios like A5/1 or DECT, where the packet number serves at the same time as an IV source, one will always have this information. Note that such packet counters are not only prevalent for standard network transmission protocols such as TCP/IP, but are also a common component of lightweight wireless devices such as RFID tags, e.g., for synchronization purposes and in order to protect against replay attacks.

Non-volatile EEPROM. The majority of RFID tags are based on ASICs (application-specific integrated circuits), whose primary types of writable storage are non-volatile EEPROMs and volatile flip-flops. In [MAM16], the designers of Plantlet extensively studied the effects of continuously reading the secret key from an EEPROM during keystream generation. Despite certain drawbacks of this approach, such as an increased design complexity and a potential reduction of the maximal achievable throughput, they concluded that this is in fact feasible. This naturally holds for any other kind of data stored in an EEPROM, too, such as a packet counter serving as the IV source and being accessed in the same way. In particular, the key-IV-schedule of Draco (cf.Section 10.1) accesses the 96 IV bits and the bits of the 32-bit key prefix in sequential order, just like the key schedule of Plantlet accesses the cipher's 80-bit key in sequential order.

The designers of Plantlet found this to be beneficial in terms of limiting the negative performance impact of continuous EEPROM access on the maximal achievable throughput. If the storage location of the IV source (such as the aforementioned network packet counter) is an array of flip-flops instead, the feasibility of continuous IV access is straightforward, because the ASIC's cryptographic logic can be connected to those flip-flops through wires at practically no cost.

Internal non-volatile memory. In the previous paragraphs, we have explained that there are various scenarios where the cipher's key and/or IV are actually already present in some storage location on the device, allowing to *reuse* this when realizing continuous key-IV access. In fact, the resulting savings on flip-flops (and, thus, chip area) inside the cipher module have long been the major motivation for such designs. In Section 10.4, we

show that due to this focus on the *number* of flip-flops, ignoring their actual *usage*, a great potential for reducing power consumption has been missed, so far.

More precisely, with our implementation variant $\text{DRACO}_{[\text{KI}]}$ we demonstrate that even if a $2n$-bit storage is required inside the cipher hardware module to achieve n-bit security against TMDTO attacks, algorithmically keeping half of this state constant is much more efficient (cf. Tab. 10.1 in Section 10.4) than and equally secure (see Section 10.3 and Section 9.5) as constantly updating the whole of it. That is, we show that even if the key and the IV are stored locally inside of the cipher hardware module (thus eliminating all the scenario assumptions / usage restrictions described above) in order to implement continuous key-IV access, our new small-state stream cipher DRACO still allows to save up to 34 % of energy as compared to Grain-128a when producing 10 kbit of keystream (including state initialization) at a clock speed of 1 GHz.

8.2 Time-memory-data Tradeoff Attacks

Stream ciphers are vulnerable to a type of attack called time-memory data tradeoff (TMDTO) attacks, as described by Babbage [Bab95] and Golić [Gol96]. We already gave a quick overview of TMDTO attacks in Section 7.5. In this section, we wish to elaborate further on TMDTO attacks and also present TMDTO attacks against enhanced state stream ciphers.

TMDTO attacks consider three cost dimensions: time T, memory M, and data D. Data D refers to the amount of data, usually observed keystream, that the adversary has obtained. Memory M measures the memory consumption of the attack, and time T measures the required computation time. Often, time T is divided into an online phase with T_{on} and an offline phase T_{off}. The offline phase is also known as precomputation and involves computing some auxiliary data structures before carrying out the actual attack in the online phase. TMDTO attacks typically specify a relation between the cost dimensions that allows for a successful attack. In other words, an adversary may spend more on one resource and, in return, less on another. Hence, these attacks are referred to as *tradeoff* attacks.

Babbage and Golić. The TMDTO attacks by Babbage and Golić are generic in the sense that they do not need to exploit the internal structure of the stream cipher. In these attacks, an adversary generates T keystream prefixes of length ℓ_s from randomly chosen internal states and observes D keystream blocks of length ℓ_s. A collision between these two is highly likely if $T \cdot D = 2^{\ell_s}$, and it is often caused by a collision of the underlying internal states. This collision allows for the recovery of the internal state used to generate the keystream. The relation $T \cdot D = 2^{\ell_s}$ has its optimal point at $T = D = 2^{\ell_s/2}$, i.e., the birthday bound, named after the birthday paradox.

Advanced attacks. An attack with a tradeoff $T \cdot M^2 \cdot D^2 = 2^{2 \cdot \ell_s}$ and $T_{\mathsf{off}} = 2^{\ell_s}/D$ was developed by Biryukov and Shamir [BS00]. They combined the attacks of Babbage and Golić with Hellman's attack on block ciphers [Hel80]. The GSM cipher A5/1 was targeted by an attack from Biryukov, Shamir, and Wagner using a method known as BSW-sampling [BSW01], which Biryukov and Shamir [BS00] also discuss. The tradeoff curve $T \cdot M^2 \cdot D^2 = 2^{2 \cdot \ell_s}$ and the relation $T_{\mathsf{off}} = 2^{\ell_s}/D$ remain the same, even though BSW-sampling allows for the relaxation of the restriction $T \geq D^2$ in the aforementioned attack. Therefore, even the use of BSW-sampling does not result in an attack with overall complexity lower than $2^{\ell_s/2}$, considering precomputation as part of the overall attack complexity. Additionally, because BSW-sampling is highly cipher-specific (see [BS00] for further details), the corresponding TMDTO attacks are not entirely generic.

Instead of sampling from the space of internal states, Hong and Sarkar [HS05] consider the TMDTO case of sampling pairs of keys and IVs. The overall complexity of this approach (including precomputation) in a single-key scenario, as analyzed by us, is at least as high as that of an exhaustive key search. A lower overall complexity can only be achieved in scenarios with multiple keys, where the attacker's objective is to discover *one* of these keys. Therefore, neither the results of [BS00] nor [HS05] conflict with our security bounds.

Key recovery. In conventional stream ciphers, a recovered internal state allows for the generation of further keystream bits and thus the decryption of additional ciphertext. Furthermore, the state update, respectively the mixing function, is often bijective. From a recovered internal state, this allows for the recovery of the loading state and thus the secret key k.

Attacks on enhanced states. In this section, we will present TMDTO attacks on all the constructions considered in this work. For simplicity, we focus on the cost dimensions time T and data D. These dimensions correspond to the queries made by an adversary, where T corresponds to queries made to the underlying primitives, and D corresponds to queries made to the construction function. Hence, focusing on T and D only will suffice to demonstrate that our bounds are tight.

We will first recap TMDTO attacks on conventional stream ciphers and then present tight TMDTO attacks for enhanced state ciphers.

8.2.1 The Conventional TMDTO Attack

The conventional time-memory-data tradeoff attack is due to Babbage [Bab95] and Golić [Gol96]. It is targeted against conventional stream ciphers according to LSSK. Its goal is to recover an internal state from some observed keystream. An adversary generates several internal states uniformly at random and computes a corresponding keystream prefix of at least length ℓ_v. A collision of this keystream prefix with the observed keystream is highly

likely due to a collision in the corresponding internal states. We will now explain the basic attack.

Attack description. The attacker proceeds as follows to attack the LSSK construction with an overall complexity in the order of $2^{\ell_v/2}$.

1. Generate a internal states s_i uniformly at random and compute a keystream prefix Z_i of length ℓ_v for every s_i. Store all (s_i, Z_i) in an efficiently searchable data structure, indexed by Z_i.
2. Obtain b keystream prefixes Y_j of length slightly larger than ℓ_v. This may be done by sliding a window of length ℓ_v over a b-bit keystream.
3. Search for a collision between Z_i and Y_j. Obtain s_i.
4. Verify if s_i is the corresponding internal state by generating some more keystream bits and comparing them to the observed ones.

Complexity. If $a \cdot b = 2^{\ell_v}$, a collision is highly likely to occur due to the birthday paradox. For $a = b = 2^{\ell_v/2}$, one obtains the optimal point with regard to overall complexity, and hence the security of the cipher is capped by a level of $\ell_v/2$ bits.

8.2.2 TMDTO Attacks Against CKEY

Mounting the attack in its original form over the entire (volatile + non-volatile) state would entail guessing the key and is therefore moot. However, there exists a distinguishing attack that works in time $2^{\ell_v/2}$. Contrary to the conventional TMDTO attack, it requires a chosen-IV attacker. The idea is that within a session, the IV changes but the key does not. In other words, one can provoke a collision within the volatile internal state for two IVs, as the non-volatile state remains fixed throughout the session.

Attack description. The attacker proceeds as follows to attack the CKEY construction with an overall complexity in the order of $2^{\ell_v/2}$.

1. Generate an initial value x' and obtain a keystream prefix Z' of length a. Let $\ell' > \ell_v$. Slide an ℓ'-bit window over Z' to obtain approximately a keystream blocks z'_i. Store z'_i in an efficiently searchable data structure $\mathscr{Z}$.
2. For b initial values x_i, obtain a keystream prefix Z_i of length ℓ'. Search for a collision in $\mathscr{Z}$. If a collision is found for $a \cdot b = 2^{\ell_v}$, we are in the real world.

Complexity. As the key remains fixed through the session, an internal state collision occurs if the volatile internal states collide. Due to the birthday paradox, this occurs with high probability if $a \cdot b = 2^{\ell_v}$. The optimal point is $a = b = 2^{\ell_v/2}$. Note that this distinguishing attack only works for a chosen-IV attacker and does not allow for the recovery of an internal state. In a world with truly random outputs, one would obtain a keystream collision of length ℓ' after generating and observing $a = b = 2^{\ell'/2} > 2^{\ell_v/2}$ blocks.

8.2.3 TMDTO Attacks Against CIV

For CIV, an attacker can perform the original TMDTO attack on the entire internal state. This will yield the tradeoff curve $D \cdot T = 2^{\ell_s} = 2^{\ell_v + \ell_{nv}}$, i.e., the optimal point is at $D = T = 2^{\ell_s/2}$. Also, note that the maximum amount of data that can be obtained is $\ell_p \cdot 2^{\ell_{\text{IV}}}$. The alternative is to exploit the structure of the cipher and obtain an attack with better complexity.

Attack description. The attacker proceeds as follows to attack the CIV construction with an overall complexity in the order of $2^{\ell_v}/\ell_p$.

1. Obtain a keystream *packets* p_i of length ℓ_p. Each packet contains approximately ℓ_p windows W_i^j of length ℓ_s. Store W_i^j in an efficiently searchable data structure. Note that the data complexity is $D = a \cdot \ell_p$.

2. For each packet p_i with initial value x_i, generate b random volatile internal states s_j and compute their corresponding keystream prefix Z_j^i of length ℓ_s. Search for a collision between Z_j^i and W_i^j. Note that the time complexity is $T = a \cdot b$.

Complexity. A collision in the volatile internal state is likely if $D \cdot b = a \cdot b \cdot \ell_p = 2^{\ell_v}$. In particular, we obtain that the time complexity of this attack is $T = 2^{\ell_v}/\ell_p$. Note that T does not depend on the amount of keystream packets observed.

8.2.4 TMDTO Attacks Against CIVK

Similarly to CIV, one can apply both attack types to CIVK. The first one remains unchanged. The second one needs to account for the key prefix and guess the corresponding key prefix in step 2.

Attack description. The attacker proceeds as follows to attack the CIVK construction with an overall complexity in the order of 2^{ℓ_v}.

1. Obtain a keystream *packets* p_i of length ℓ_p. Each packet contains approximately ℓ_p windows W_i^j of length ℓ_s. Store W_i^j in an efficiently searchable data structure. Note that the data complexity is $D = a \cdot \ell_p$.

2. For each packet p_i with initial value x_i, generate b pairs of key prefixes *and* random volatile internal states (k^{pre}, s_j) and compute their corresponding keystream prefix Z_j^i of length ℓ_s. Search for a collision between Z_j^i and W_i^j. Note that the time complexity is $T = a \cdot b$.

Complexity. A collision in the volatile internal state is likely if $D \cdot b = a \cdot b \cdot \ell_p = 2^{\ell_v}$. Further, one needs to consider that the key prefix guesses are correct with a probability of ℓ_p^{-1}, i.e., $a \cdot b \cdot \ell_p \cdot \ell_p^{-1} = 2^{\ell_v}$. This yields a total time complexity of the attack of $T = 2^{\ell_v}$. Note that T does not depend on the amount of keystream packets observed.

Remark 8.1

Note that for CIV and CIVK, assuming a chosen-IV attacker, all $a \cdot b = 2^{\ell_v}$ keystream packets can be precomputed and need to be stored in an efficiently searchable data structure, i.e., a binary tree. This will require a time complexity of 2^{ℓ_v} in the offline phase and a space (memory) complexity of 2^{ℓ_v}. The online phase, when observing a keystream packet, will then have a time complexity of $\log_2(2^{\ell_v}) = \ell_v$ to search for the collision on the previously computed keystream packets. While the online time complexity is significantly reduced, there is an excessive amount of precomputation time necessary, and the space complexity in the online phase is 2^{ℓ_v}. Even if a time complexity of 2^{ℓ_v} were not excessive, a space complexity of 2^{ℓ_v} may very well be.

9 | Proving Security

In the following, we will present the preliminaries, notation, and random oracle model necessary for the proof of security. In this section, we will stick with the generic $\langle a \mid b \rangle$ notation, where a describes the non-volatile part of the cipher and b describes the volatile part of the cipher. This will suffice for describing the model and keep the notation simple in this section. In the proofs of the respective constructions, we will give detailed information about the internal state, queries, and transcripts.

9.1 Proof Preliminaries

In this section, we present the random oracle model, the distinguishing game, and the adversarial strategy that will be used throughout our proofs.

9.1.1 Random Oracle Model

An adversary will be interacting with a set of three oracles in one of two worlds: the real world or the ideal world. There will be an oracle P for the mixing function, an oracle F for the output function, and an oracle E for the construction function.[1] E is defined differently in either world. The adversary can query the oracles with the inputs of the respective functions, and it will receive answers from the oracle. These query-answer pairs are collected by the adversary in a transcript τ. Finally, the adversary has to output a decision bit.

Oracles. The P- and F-oracles will answer their queries using ideal randomized primitives in either world. The P-oracle will use a random permutation $\mathbf{P}$ (for CIV and CIVK: multiple random permutations as described later), and the F-oracle will use a random function $\mathbf{F}$. In the real world, the E-oracle uses $\mathbf{P}$ and $\mathbf{F}$ as underlying building blocks. In the ideal world, the E-oracle will have access to another independent random function $\mathbf{E}$. By assuming the underlying building blocks to be ideal, one can abstract from possible weaknesses in the mixing function and the output function that an implementation may have and show that the scheme, the interaction of those building blocks, is secure. This will *not*

[1] π is publicly known and hence no oracle is needed.

prove an instantiation to be secure but provide a plausible justification for the structure of a cipher built upon the respective construction.

In the ideal world, **E**, corresponding to the encryption function, will sample the output bits uniformly at random from $\{0,1\}$. We will show that the adversary cannot distinguish the ideal world from the real world in this scenario. In particular, this will show that the keystream generated by the 'real' encryption function is indistinguishable from a truly random bitstream.

9.1.2 The Distinguishing Game

In the beginning of the adversary's interaction with the oracles, a key $k \twoheadleftarrow \mathscr{K}$ will be sampled uniformly at random from the key space $\mathscr{K}$. Next, the adversary poses its questions to the P-, F-, and E-oracles with the additional limit of at most ℓ_p E-queries per IV x.[2] All of the query-answer pairs will be collected in the corresponding τ_P, τ_F, and τ_E transcripts.

After the interaction. When the adversary is finished with its interaction with the oracles, it is given the secret key k, as well as the transcript τ_α, which contains the intermediate values corresponding to internal states generated during an E-query and will be defined more explicitly later.

Final decision. Based on the transcript $\tau = (\tau_P, \tau_E, \tau_F, \tau_\alpha, k)$, the adversary has to make its decision whether it was interacting with the real world or the ideal world and output a decision bit. If the adversary's guess is correct, it wins the game. Our task is to upper bound the adversary's success probability.

9.1.3 Oracle Queries

The adversary will be given access to the P-, F-, and E-oracles that correspond to the mixing function, output function, and construction function, respectively. For clarity, in the beginning of the proof of each construction, we provide a table that describes the structure of the inputs and outputs of the respective queries, as chosen by the adversary. We will then explain how the oracles are implemented. Note that we index the query variables with the query type, i.e., $\langle a_P \mid b_P \rangle$ denotes a P-query where a_P and b_P are chosen by the adversary.

P-Oracle. The P-oracle that corresponds to the mixing function will be implemented using random permutations. Note that the mixing function p keeps the non-volatile internal state unchanged and only the volatile internal state gets permuted. Hence, for every $a \in \mathscr{Q}_{nv}$, the volatile internal state $b \in \mathscr{Q}_v$ is permuted using an independent random permutation $\mathbf{P}_a : \mathscr{Q}_v \to \mathscr{Q}_v$. For each $\mathbf{P}_a$, we will use lazy sampling: as the first P-query

[2] Note that the key remains fixed throughout the game, so only the IV is changed.

$\langle a_P \,|\, b_P \rangle$ arrives, the oracle will sample the answer uniformly at random from all 2^{ℓ_v} possible answers. As the second P-query arrives, the oracle will sample the answer uniformly at random from all $2^{\ell_v} - 1$ remaining possible answers, and so on. The permutation $\mathbf{P}$ is defined as follows:

$$\mathbf{P} : \mathcal{Q}_{nv} \times \mathcal{Q}_v \to \mathcal{Q}_{nv} \times \mathcal{Q}_v, \quad \langle a_P \,|\, b_P \rangle \mapsto \langle a_P \,|\, \mathbf{P}_{a_P}(b_P) \rangle$$

The adversary may query the P-oracle in either the forward or the backward direction. That is, the adversary gets oracle access to $\mathbf{P}$ as well as $\mathbf{P}^{-1}$.

F-Oracle. The F-oracle that corresponds to the output function will be implemented using a random function $\mathbf{F} : \mathcal{Q}_{nv} \times \mathcal{Q}_v \to \{0,1\}$. Only queries in the forward direction are allowed. We again use lazy sampling; as an F-query arrives, the output is sampled uniformly at random from $\{0,1\}$.

E-Oracle. The E-oracle is defined differently in the real world and in the ideal world: In the real world, it is defined similarly to the construction function e and implicitly uses the secret key k, the random permutation $\mathbf{P}$, the state update function π, and the random function $\mathbf{F}$:

$$\mathbf{E} : \mathcal{IV} \times \{0, \ldots, \ell_p - 1\} \to \{0,1\}, \quad (x_E, r_E) \mapsto \mathbf{F}(\pi^{r_E}(\mathbf{P}(\mathtt{load}(k, x_E))))$$

In the ideal world, $\mathbf{E} : \mathcal{IV} \times \{0, \ldots, \ell_p - 1\} \to \{0,1\}$ is a random function with outputs sampled uniformly at random from $\{0,1\}$.

No redundant queries. We will assume that the adversary does not make redundant queries, i.e., the adversary will not query for a value that has been queried before. As redundant queries yield no additional information, this assumption does not lower the adversary's distinguishing advantage.

9.1.4 Transcripts

All of the adversary's queries to the oracles and the corresponding answers will be collected in a transcript τ. In particular, we will keep separate transcripts τ_P, τ_F, and τ_E for each of the corresponding oracles P, F, and E defined as follows:

$$\begin{aligned}
\tau_P &:= \{(a_P, b_P, y_P) \,|\, \langle a_P \,|\, b_P \rangle \text{ is a } P\text{-query and } \langle a_P \,|\, y_P \rangle \text{ its answer.}\} \\
\tau_F &:= \{(a_F, b_F, z_F) \,|\, \langle a_F \,|\, b_F \rangle \text{ is an } F\text{-query and } z_F \text{ its answer.}\} \\
\tau_E &:= \{(x_E, r_E, z_E) \,|\, (x_E, r_E) \text{ is an } E\text{-query and } z_E \text{ its answer.}\}
\end{aligned}$$

The transcripts mentioned above are visible to the adversary as it makes its queries to the oracles. Once the adversary's interaction with the oracles is finished, it will additionally be

given the secret key k and the transcript τ_α defined as follows:

$$\tau_\alpha := \{(x_E, r_E, \alpha_E, \alpha_E^r) \mid (x_E, r_E) \text{ is an } E\text{-query and } (\alpha_E, \alpha_E^r) \text{ its } \alpha\text{-values.}\}$$

The α-values for each E-query (x_E, r_E) are defined as follows:

$$\alpha_E := P(\mathtt{load}(k, x_E)) \text{ and } \alpha_E^{r_E} := \pi^{r_E}(P(\mathtt{load}(k, x_E))).$$

The α-values correspond to the internal states that are generated during an E-query. α_E represents the initial state, and $\alpha_E^{r_E}$ represents the internal state that is the input to the output function $\mathbf{F}$ after r_E state updates on the initial state. We will also sample these values in the ideal world, even though the construction function E does not depend on these.

The full transcript can be written as a 5-tuple $\tau = (\tau_P, \tau_E, \tau_F, \tau_\alpha, k)$.

9.1.5 H-coefficient Technique

We will briefly recapture the H-coefficients technique as it is central to our proof. The H-coefficients technique is a proof method due to Patarin, where we consider the variant by Chen and Steinberger [CS14, Pat08]. The results of the interaction of an adversary $\mathbf{A}$ with its oracles are collected in a transcript τ. The oracles can sample randomness prior to the interaction (often a key or an ideal primitive that is sampled beforehand), and are then deterministic throughout the experiment [CS14]. The task of $\mathbf{A}$ is to distinguish the real world $\mathcal{O}_{\mathsf{real}}$ from the ideal world $\mathcal{O}_{\mathsf{ideal}}$. Let Θ_{real} and Θ_{ideal} denote the distribution of transcripts in the real and the ideal world, respectively. A transcript τ is called *attainable* if the probability to obtain τ in the ideal world – i.e. over Θ_{ideal} – is non-zero. Then, the fundamental Lemma of the H-coefficients technique, the proof to which is given in [CS14, Pat08], states:

Lemma 9.1 (***H-coefficients technique***)

Assume that the set of attainable transcripts can be partitioned into two disjoint sets GoodT and BadT. Further, assume that there exist $\delta, \epsilon \in [0, 1]$ such that for any transcript $\tau \in \mathsf{GoodT}$, it holds that:

$$\Pr[\Theta_{\mathsf{ideal}} \in \mathsf{BadT}] \leq \delta \quad \text{and} \quad \frac{\Pr[\Theta_{\mathsf{real}} = \tau]}{\Pr[\Theta_{\mathsf{ideal}} = \tau]} \geq 1 - \epsilon.$$

Then for all adversaries $\mathbf{A}$ we have that the distinguishing advantage satisfies

$$\underset{\mathbf{A}}{\triangle}(\Theta_{\mathsf{ideal}}, \Theta_{\mathsf{real}}) \leq \delta + \epsilon.$$

9.1.6 The Adversarial Strategy

From an attacker's point of view, the goal is to recover an internal state by obtaining a collision between an observed keystream and a generated keystream. If the keystreams collide, there is a high probability that they were generated from the same internal state. In particular, the same internal state always generates the same keystream in the real world.

Collisions. In our model, the adversary does *not* have to rely on observing collisions in the keystream as it is provided the α-values at the end of the interaction. The α-values correspond to the internal states of the respective output bit. Hence, the adversary's goal will be to obtain a collision between the α-values and the inputs of the F-queries.

EF-Collisions. If there occurs a collision between an α-value (corresponding to an E-query) and an F-query, the corresponding outputs of the E-query and the F-query will always be identical *in the real world*, as $\mathbf{E}$ internally uses $\mathbf{P}$, π, and $\mathbf{F}$. *In the ideal world*, however, $\mathbf{E}$ is another random function independent of $\mathbf{P}$, π, and $\mathbf{F}$, and hence, colliding α-values and F-query inputs only lead to the same output bit with a probability of $\frac{1}{2}$. This will be used as a distinguishing, i.e., bad, event. In general, there are two ways to obtain such a collision: (1) guessing an internal state corresponding to the α-values and (2) guessing the secret key k and deriving the α-value from a P-query.

EE-Collisions. Further, for LSSK and CKEY, there may occur internal state collisions between two distinct (i.e., with different IVs) E-queries. In the ideal world, the output bits may be different even though the α-values collide. For CIV and CIVK, an α-value collision between two distinct E-queries may not occur as distinct IVs imply distinct α-values.

9.1.7 Structure of the Analysis

Using the H-coefficients technique, the bounds in all of our proofs will ultimately only be determined by δ, as we will show that $\Pr[\Theta_{\mathsf{real}} = \tau] = \Pr[\Theta_{\mathsf{ideal}} = \tau]$ and thus $\epsilon = 0$. Hence, for a good transcript, the real and the ideal world are indistinguishable, so the security bounds will consist only of the bad events' probability. In Sections 9.2 to 9.5, we will analyze and upper bound the probability of a bad event in the ideal world for each of the four constructions. Once this has been done, the analysis of the good transcripts is identical for all constructions and will be done in Section 9.6.

We chose this structure because the analyses in Sections 9.2 to 9.5 are quite similar, so it makes sense to group them together. Additionally, when analyzing the good transcripts, it is necessary to know which specific bad events have been excluded in order to perform a proper analysis and obtain a good bound for ϵ. If a bad event were missed, it would appear in the good transcripts and lead to a poor bound for ϵ.

Table 9.1: Inputs and outputs of oracle queries to LSSK.

Query Type	Input	Output
P	(x_P, k_P)	y_P
P^{-1}	y_P	(x_P, k_P)
F	y_F	$z_F \in \{0,1\}$
E	(x_E, r_E)	$z_E \in \{0,1\}$

Query notation. When writing a query, we will use indexed letters like x_P, k_P, and y_P to denote the adversary's choice. These need not be equal to an actual IV, key, or internal state. For example, k_P denotes a *key guess* in a P-query by the adversary and needs not be equal to the actual secret key k.

9.2 Analysis of LSSK

We begin with the analysis of the conventional LSSK construction. All parts of the internal state are volatile, i.e., the non-volatile part of the cipher is empty, respectively non-existent. Therefore, the state length ℓ_s is equal to the volatile state length ℓ_v, i.e., $\ell_s = \ell_v$. Further, LSSK does not define a packet length, and thus up to 2^{ℓ_s} construction queries are possible per key-IV pair (k, x). The inputs and outputs of the queries to the respective oracles can be found in Table 9.1. For an E-query (x_E, r_E), the α-values of LSSK are defined as follows:

$$\alpha_E := P(x_E, k) \quad \text{and} \quad \alpha_E^{r_E} := \pi^{r_E}(\alpha_E).$$

The four transcripts are defined in analogy to the oracle queries and α-values:

$$\tau_P := \{(x_P, k_P, y_P) \mid (x_P, k_P) \text{ is a } P\text{-query and } y_P \text{ its answer.}\}$$
$$\tau_F := \{(y_F, z_F) \mid y_F \text{ is an } F\text{-query and } z_F \text{ its answer.}\}$$
$$\tau_E := \{(x_E, r_E, z_E) \mid (x_E, r_E) \text{ is an } E\text{-query and } z_E \text{ its answer.}\}$$
$$\tau_\alpha := \left\{(x_E, r_E, \alpha_E, \alpha_E^{r_E}) \mid (x_E, r_E) \text{ is an } E\text{-query and } (\alpha_E, \alpha_E^{r_E}) \text{ its } \alpha\text{-values.}\right\}$$

From Lemma 4.1 (H-coefficients technique), Lemma 9.2 (bad events), and Theorem 9.1 (good transcripts), we will obtain the following bound on the adversarial advantage for LSSK:

Corollary 9.1

For the LSSK construction as defined in Subsection 8.1.1, where the mixing function p and the output function f are replaced with their ideal counterparts as defined in Subsection 9.1.3, we have that for all adversaries $\mathbf{A}$ asking at most q queries, it holds

that:

$$\underset{\mathrm{A}}{\Delta}(\mathcal{O}_{\mathsf{real}}, \mathcal{O}_{\mathsf{ideal}}) \leq \frac{q}{2^{\ell_k}} + \frac{2q^2}{2^{\ell_v} - q}.$$

9.2.1 Overview of the Bad Events

We will define three distinguishing events for LSSK. The first bad event covers the adversary guessing an internal state as input to the F-oracle that collides with an α-value (internal state) of an E-query. The second bad event is similar to the first one in that the adversary is trying to create the same type of collision. This time, however, the adversary guesses the key and tries to obtain the colliding internal state through a P-query (if the key guess is correct). The third bad event differs. In LSSK, there may be α-value collisions between two E-queries that allow for distinguishing.

In the real world, these collisions would imply identical outputs. In the ideal world, however, E is independent of F, and thus the collisions need not have identical outputs as a consequence. This is used for distinguishing.

9.2.2 Bad Events

According to Section 9.1, we will define our bad events.

EF-Collisions. The first bad event covers collisions between construction queries and output function queries. This type of collision corresponds to a correct internal state guess.

Bad Event (bad_1)

There exists an E-query (x_E, r_E) and an F-query y_F such that

$$\alpha_E^{r_E} = y_F.$$

Note that $P(x_E, k)$, i.e., the α-value, is sampled at the end of the interaction.

Key guesses. The second bad event covers correct key guesses. Here, a correct key guess allows the adversary to obtain a valid internal state for a given IV x_P after asking the corresponding P-query.

Bad Event (bad_2)

There exist a P-query (x_P, k_P), an E-query (x_P, r_E), and an F-query $\pi^{r_E}(P(x_P, k_P))$ such that

$$k_P = k.$$

EE-Collisions. In case of the LSSK construction, we also need to consider internal state collisions between two E-queries. Generally, the internal states of two different E-queries can collide. In the ideal world, however, the outputs can differ as E is modeled as a random function.

Bad Event (bad_3)

There exist two E-queries (x_E, r_E) and (x'_E, r'_E), where $x_E \neq x'_E$, such that

$$\alpha_E^{r_E} = \alpha_E^{r'_E}.$$

Remark 9.1

Note that to successfully distinguish, the adversary further needs to observe that

- $E(x_E, r_E) \neq F(y_F)$ for bad_1,
- $E(x_P, r_E) \neq F(\pi^{r_E}(P(x_P, k_P)))$ for bad_2,
- $E(x_E, r_E) \neq E(x'_E, r'_E)$ for bad_3.

We omit these to simplify the analysis of the good transcripts in Section 9.6.

9.2.3 Bounding the Bad Events

In this subsection, we will present an upper bound on the bad events introduced in Subsection 9.2.2. In particular, we will show the following lemma:

Lemma 9.2

The probability of a bad event occurring for LSSK is upper bounded by:

$$\Pr[\Theta_{\mathsf{ideal}} \in \mathsf{BadT}] \leq \frac{q}{2^{\ell_k}} + \frac{2q^2}{2^{\ell_v} - q}.$$

Proof. Since we are in the ideal world, the answers to the E-, F-, and P-queries are independent of the secret key k. For simplicity, we will sample the secret key k after the adversary's interaction with the oracles. Also, we will sample the values α_E for each query after the adversary's interaction with the oracles.

Key guesses. We will first consider the event bad_2. There are at most q P-queries with at most q distinct k_P. The secret key k is sampled independently at random with regard to the uniform distribution from the set $\{0,1\}^{\ell_k}$. The probability of a collision with

Table 9.2: Inputs and outputs of oracle queries to CKEY.

Query Type	Input	Output
P	$\langle k_P \mid x_P \rangle$	$\langle k_P \mid y_P \rangle$
P^{-1}	$\langle k_P \mid y_P \rangle$	$\langle k_P \mid x_P \rangle$
F	$\langle k_F \mid y_F \rangle$	$z_F \in \{0,1\}$
E	(x_E, r_E)	$z_E \in \{0,1\}$

the secret key k can therefore be upper bounded by $q/2^{\ell_k}$, i.e., we obtain:

$$\Pr[\mathsf{bad}_2] \leq \frac{q}{2^{\ell_k}}.$$

EF-Collisions. We will now consider the event bad_1. Since we already bounded the probability for bad_2, we will now consider $\Pr[\mathsf{bad}_1 \mid \neg\mathsf{bad}_2]$. We decided to sample α_E after the adversary's interaction with the oracles. Further, since we conditioned on $\neg\mathsf{bad}_2$, the corresponding values α_E^r for all E-queries (x_E, r) have not yet been sampled.

The amount of E- and F-queries, respectively, can trivially be upper bounded by q. Hence, the amount of pairs can be upper bounded by q^2. As we sample the α_E values at the end of the interaction uniformly at random without replacement from $\{0,1\}^{\ell_v}$, we can upper bound the collision probability by $(2^{\ell_v} - q)^{-1}$ for each pair. Thus, we obtain:

$$\Pr[\mathsf{bad}_1] \leq \frac{q^2}{2^{\ell_v} - q}.$$

EE-Collisions. Last, we consider the event bad_3. The total amount of distinct E-queries can be upper bounded by q. Fix some E-query (x_E, r_E) and consider the value $\alpha_E^{r_E}$. Now consider a second E-query (x'_E, r'_E) where the corresponding value $\alpha_E^{r'_E}$ has not yet been sampled. The probability that $P(x'_E, k)$ maps to $\pi^{-r'_E}(\alpha_E^{r_E})$ is upper bounded by $(2^{\ell_v} - q)^{-1}$. Over all q^2 pairs of E-queries, we obtain:

$$\Pr\left[\mathsf{bad}_3\right] \leq \frac{q^2}{2^{\ell_v} - q}.$$

Lemma 9.2 follows from the union bound of the above individual bad events. ∎

9.3 Analysis of CKEY

Of the enhanced state ciphers, we will consider the CKEY construction first. Here, the non-volatile part of the cipher consists only of the secret key k. Accordingly, the volatile loading state consists of the IV x only. CKEY does not define a packet length, and thus up to 2^{ℓ_v} construction queries are possible per key-IV pair (k, x). The inputs and outputs

of the queries to the respective oracles can be found in Table 9.2. For an E-query (x_E, r_E), the α-values of CKEY are defined as follows:

$$\alpha_E := P\langle k \,|\, x_E\rangle \quad \text{and} \quad \alpha_E^{r_E} := \pi^{r_E}(\alpha_E).$$

The four transcripts are defined in analogy to the oracle queries and α-values:

$$\begin{aligned}
\tau_P &:= \{(x_P, k_P, y_P) \,|\, \langle k_P \,|\, x_P\rangle \text{ is a } P\text{-query and } \langle k_P \,|\, y_P\rangle \text{ its answer.}\}\\
\tau_F &:= \{(k_F, y_F, z_F) \,|\, \langle k_F \,|\, y_F\rangle \text{ is an } F\text{-query and } z_F \text{ its answer.}\}\\
\tau_E &:= \{(x_E, r_E, z_E) \,|\, (x_E, r_E) \text{ is an } E\text{-query and } z_E \text{ its answer.}\}\\
\tau_\alpha &:= \left\{(x_E, r_E, \alpha_E, \alpha_E^{r_E}) \,\middle|\, (x_E, r_E) \text{ is an } E\text{-query and } (\alpha_E, \alpha_E^{r_E}) \text{ its } \alpha\text{-values.}\right\}
\end{aligned}$$

From Lemma 4.1 (H-coefficients technique), Lemma 9.3 (bad events), and Theorem 9.1 (good transcripts), we will obtain the following bound on the adversarial advantage for CKEY:

Corollary 9.2

For the CKEY construction as defined in Subsection 8.1.1, where the mixing function p and the output function f are replaced with their ideal counterparts as defined in Subsection 9.1.3, we have that for all adversaries $\mathbf{A}$ asking at most q queries, it holds that:

$$\underset{\mathbf{A}}{\Delta}(\mathcal{O}_{\text{real}}, \mathcal{O}_{\text{ideal}}) \leq \frac{q}{2^{\ell_k}} + \frac{q^2}{2^{\ell_v} - q}.$$

9.3.1 Overview of the Bad Events

We will define two distinguishing events for CKEY. Essentially, the distinguishing events are the same as for LSSK. One should note that for guessing an internal state, the adversary would have to guess the volatile internal state as well as the non-volatile key correctly. Therefore, it makes more sense for an adversary to guess the key and obtain the colliding internal state through a P-query. Hence, we only consider these as one bad event, namely correct key guesses. As is the case for LSSK, there may be α-value collisions between two E-queries that allow for distinguishing.

In the real world, these collisions would imply identical outputs. In the ideal world, however, E is independent of F, and thus the collisions need not have identical outputs as a consequence. This is used for distinguishing.

9.3.2 Bad Events

According to Section 9.1, we will define our bad events.

EF-Collisions. The first distinguishing event covers a collision between a construction query and an output function query. This corresponds to an adversary correctly guessing the internal state.

Bad Event (bad_1)

There exists an E-query (x_E, r_E) and an F-query $\langle k_F \mid y_F \rangle$ such that:

$$\alpha_E^{r_E} = \langle k_F \mid y_F \rangle.$$

Key guesses. Note that in case of the CKEY construction, a prerequisite to get a proper internal state collision, the attacker needs to guess the key right. Here, we will cover all key guesses, i.e., also through P-queries, in the relaxed variant of bad_1:

Bad Event (bad_2)

There exists a P-query $\langle k_P \mid x_P \rangle$ or an F-query $\langle k_F \mid y_F \rangle$ such that:

$$k_P = k \text{ or } k_F = k.$$

Remark 9.2

Ultimately, the only way for the adversary to distinguish is to observe different outputs for identical internal states. This can only happen in the ideal world. In the case of the CKEY construction, it is, in fact, easier for the adversary to guess the key and then derive a corresponding internal state using a P-query than to guess an internal state for an F-query, as this also involves guessing the key. We, therefore, ignore bad_1 as it is implied by bad_2.

EE-Collisions. In the case of the CKEY construction, we also need to consider internal state collisions between two E-queries. Colliding internal states between two different E-queries with different outputs give an adversary a tool to distinguish.

Bad Event (bad_3)

There exist two E-queries (x_E, r_E) and (x'_E, r'_E), where $x_E \neq x'_E$ such that:

$$\alpha_E^{r_E} = \alpha_E^{r'_E}.$$

Remark 9.3

Note that to successfully distinguish, the adversary further needs to observe that

– $E(x_E, r_E) \neq F\langle k_F \mid y_F \rangle$ for bad_1,

– $E(x_E, r_E) \neq E(x'_E, r'_E)$ for bad_3.

We omit these to simplify the analysis of the good transcripts in Section 9.6.

9.3.3 Bounding the Bad Events

In this subsection, we will present an upper bound on the bad events introduced in Subsection 9.3.2. In particular, we will show the following lemma:

Lemma 9.3

The probability of a bad event occurring for CKEY is upper bounded by:

$$\Pr[\Theta_{\mathsf{ideal}} \in \mathsf{BadT}] \leq \frac{q}{2^{\ell_k}} + \frac{q^2}{2^{\ell_v} - q}.$$

Proof. Since we are in the ideal world, the answers to the E-, F-, and P-queries are independent of the secret key k. For simplicity, we will sample the secret key k after the adversary's interaction with the oracles. Also, we will sample the values α_E for each query after the adversary's interaction with the oracles.

Key guesses. We will first consider the event bad_2. There are at most q queries with at most q distinct k_P or k_F. The secret key k is sampled independently at random with regard to the uniform distribution from the set $\{0,1\}^{\ell_k}$. The probability of a collision with the secret key k can therefore be upper bounded by:

$$\Pr[\mathsf{bad}_2] \leq \frac{q}{2^{\ell_k}}.$$

EE-Collisions. Next, we consider the event bad_3. The total amount of distinct E-queries can be upper bounded by q. Fix some E-query (x_E, r_E) and consider the value $\alpha_E^{r_E}$. Now consider a second E-query (x'_E, r'_E) where the corresponding value $\alpha_E^{r'_E}$ has not yet been sampled. The probability that $P\langle k \mid x'_E \rangle$ maps to $\pi^{-r'_E}(\alpha_E^{r_E})$ is upper bounded by $(2^{\ell_v} - q)^{-1}$. Over all q^2 pairs of E-queries, we obtain:

$$\Pr\left[\mathsf{bad}_3\right] \leq \frac{q^2}{2^{\ell_v} - q}.$$

Lemma 9.3 follows from the union bound of the above individual bad events. ∎

Table 9.3: Inputs and outputs of the oracle queries to CIV.

Query Type	Input	Output
P	$\langle x_P \mid k_P \rangle$	$\langle x_P \mid y_P \rangle$
P^{-1}	$\langle x_P \mid y_P \rangle$	$\langle x_P \mid k_P \rangle$
F	$\langle x_F \mid y_F \rangle$	$z_F \in \{0,1\}$
E	(x_E, r_E)	$z_E \in \{0,1\}$

9.4 Analysis of CIV

CIV uses the IV in the non-volatile part of the cipher, whereas the key is contained in the volatile part of the loading state. A packet length is defined, and up to ℓ_p bits may be output per key-IV pair (k, x). The inputs and outputs of the queries to the respective oracles can be found in Table 9.3. The α-values for each E-query (x_E, r_E) are defined as follows:

$$\alpha_E := P\langle x_E \mid k \rangle \quad \text{and} \quad \alpha_E^{r_E} := \pi^{r_E}(P\langle x_E \mid k \rangle).$$

The four transcripts are defined in analogy to the oracle queries and α-values:

$$\tau_P := \{(x_P, k_P, y_P) \mid \langle x_P \mid k_P \rangle \text{ is a } P\text{-query and } \langle x_P \mid y_P \rangle \text{ its answer.}\}$$
$$\tau_F := \{(x_F, y_F, z_F) \mid \langle x_F \mid y_F \rangle \text{ is an } F\text{-query and } z_F \text{ its answer.}\}$$
$$\tau_E := \{(x_E, r_E, z_E) \mid (x_E, r_E) \text{ is an } E\text{-query and } z_E \text{ its answer.}\}$$
$$\tau_\alpha := \left\{(x_E, r_E, \alpha_E, \alpha_E^{r_E}) \mid (x_E, r_E) \text{ is an } E\text{-query and } (\alpha_E, \alpha_E^{r_E}) \text{ its } \alpha\text{-values.}\right\}$$

From Lemma 4.1 (H-coefficients technique), Lemma 9.4 (bad events), and Theorem 9.1 (good transcripts), we will obtain the following bound on the adversarial advantage for CIV:

Corollary 9.3

For the CIV construction as defined in Subsection 8.1.1, where the mixing function p and the output function f are replaced with their ideal counterparts as defined in Subsection 9.1.3, and up to ℓ_p adversarial queries per IV $x \in \mathcal{IV}$, we have that for all adversaries $\mathbf{A}$ asking at most q queries, it holds that:

$$\underset{\mathbf{A}}{\Delta}(\mathcal{O}_{\text{real}}, \mathcal{O}_{\text{ideal}}) \leq \frac{q}{2^{\ell_k}} + \frac{\ell_p \cdot q}{2^{\ell_v} - q}.$$

9.4.1 Overview of the Bad Events

We will define two distinguishing events for CIV. Essentially, the distinguishing events are identical to the first two of LSSK. One distinguishing event corresponds to guessing an internal state to provoke a collision, and the other distinguishing event corresponds to guessing the secret key to then derive an internal state collision.

Internal state collisions between two E-queries cannot occur. As the IV x is part of the non-volatile internal state, a collision between queries with different IVs is not possible. Collisions of internal states from queries with the same IV are not possible as the state update function π is required to have a period larger than ℓ_p.

In the real world, these collisions would imply identical outputs. In the ideal world, however, E is independent of F, and thus the collisions need not have identical outputs as a consequence. This is used for distinguishing.

9.4.2 Bad Events

We have to identify bad events that will trivially allow to distinguish the real world from the ideal world. According to Section 9.1, we will define our bad events.

EF-Collisions. The first distinguishing event covers a collision between a construction query and an output function query. This corresponds to an adversary correctly guessing the internal state.

Bad Event (bad_1)

There exists an E-query (x_E, r_E) and an F-query $\langle x_F \mid y_F \rangle$ such that:

$$\alpha_E^{r_E} = \langle x_F \mid y_F \rangle.$$

Note that $P\langle x_E \mid k \rangle$ is sampled at the end of the interaction.

Key guesses. We also define a bad event that covers key guesses. The adversary guesses a key, performs a P-query with its guess, and uses the then obtained internal state to produce a collision between an E- and an F-query.

Bad Event (bad_2)

There exists a P-query $\langle x_P \mid k_P \rangle$, an E-query (x_P, r_E), and an F-query $\pi^{r_E}(P\langle x_P \mid k_P \rangle)$ such that:

$$k_P = k.$$

Remark 9.4

Note that bad_2 represents a special case of bad_1. In particular, we have that:

$$P\langle x_P \mid k_P\rangle = \alpha_E \text{ and } \pi^{r_E}(P\langle x_P \mid k_P\rangle) = \alpha_E^{r_E}.$$

Ultimately, the correct key guess is used to acquire a valid internal state. That internal state is then used to check whether the corresponding E- and F-queries differ, which then allows to distinguish. Yet, for simplicity, we consider these as separate bad events. We will first bound the probability of bad_2 and then derive a bound for bad_1 conditioned on bad_2 *not* occurring.

Also note that to successfully distinguish, the adversary further needs to observe $E(x_E, r_E) \neq F\langle x_F \mid y_F\rangle$ for bad_1, and $E(x_P, r_E) \neq F(\pi^{r_E}(P\langle x_P \mid k_P\rangle))$ for bad_2. We omit these to simplify the analysis of the good transcripts in Section 9.6.

9.4.3 Bounding the Bad Events

In this subsection, we will present an upper bound on the bad events introduced in Subsection 9.4.2. In particular, we will show the following lemma:

Lemma 9.4

The probability of a bad event occurring for CIV is upper bounded by:

$$\Pr[\Theta_{\mathsf{ideal}} \in \mathsf{BadT}] \leq \frac{q}{2^{\ell_k}} + \frac{\ell_p \cdot q}{2^{\ell_v} - q}.$$

Proof. Since we are in the ideal world, the answers to the E-, F-, and P-queries are independent of the secret key k. For simplicity, we will sample the secret key k after the adversary's interaction with the oracles. Also, we will sample the values α_E for each query after the adversary's interaction with the oracles.

Key guesses. We will first consider the event bad_2. There are at most q P-queries with at most q distinct k_P. The secret key k is sampled independently at random with regard to the uniform distribution from the set $\{0,1\}^{\ell_k}$. The probability of a collision with the secret key k can therefore be upper bounded by $q/2^{\ell_k}$. We obtain:

$$\Pr[\mathsf{bad}_2] \leq \frac{q}{2^{\ell_k}}.$$

EF-Collisions. We will now consider the event bad_1. Since we already bounded the probability for bad_2, we will now consider $\Pr[\mathsf{bad}_1 \mid \neg\mathsf{bad}_2]$. We decided to sample α_E after the adversary's interaction with the oracles. Further, since we conditioned on $\neg\mathsf{bad}_2$, the corresponding α-values $P\langle x_E \mid k\rangle$ for all E-queries (x_E, r_E) have not yet been sampled.

Table 9.4: Inputs and outputs of the oracle queries to CIVK.

Query Type	Input	Output
P	$\langle x_P, k_P^{\mathsf{pre}} \mid k_P \rangle$	$\langle x_P, k_P^{\mathsf{pre}} \mid y_P \rangle$
P^{-1}	$\langle x_P, k_P^{\mathsf{pre}} \mid y_P \rangle$	$\langle x_P, k_P^{\mathsf{pre}} \mid k_P \rangle$
F	$\langle x_F, k_F^{\mathsf{pre}} \mid y_F \rangle$	$z_F \in \{0,1\}$
E	(x_E, r_E)	$z_E \in \{0,1\}$

The amount of F-queries can trivially be upper bounded by q. Fix any F-query $\langle x_F \mid y_F \rangle$. Since the amount of E-queries is bounded to ℓ_p queries per IV x_E, there are at most ℓ_p α-values to collide with.

Fix some E-query (x_E, r_E) where $x_E = x_F$. Now, consider the internal state $\rho = \pi^{-r_E}\langle x_F \mid y_F \rangle$. The α-value $P\langle x_E \mid k \rangle$ is sampled after the F-query. As there are at most q queries, we obtain for a fixed E-query and a fixed F-query:

$$\Pr\left[P\langle x_E, k^{\mathsf{pre}} \mid k \rangle = \rho\right] \leq (2^{\ell_v} - q)^{-1}.$$

Over at most q F-queries and up to ℓ_p matching E-queries per F-query, we obtain:

$$\Pr[\mathsf{bad}_1 \mid \neg\mathsf{bad}_2] \leq \frac{\ell_p \cdot q}{2^{\ell_v} - q}.$$

Lemma 9.4 follows from the union bound of the above individual bad events. ∎

9.5 Analysis of CIVK

CIVK uses the IV as well as a key prefix in the non-volatile part of the cipher, whereas the key is contained in the volatile part of the loading state. A packet length is defined, and up to ℓ_p bits may be output per key-IV pair (k, x). The inputs and outputs of the queries to the respective oracles can be found in Table 9.4. The α-values for each E-query (x_E, r_E) are defined as follows:

$$\alpha_E := P\langle x_E, k^{\mathsf{pre}} \mid k \rangle \quad \text{and} \quad \alpha_E^{r_E} := \pi^{r_E}(P\langle x_E, k^{\mathsf{pre}} \mid k \rangle).$$

The four transcripts are defined in analogy to the oracle queries and α-values:

$$\tau_P := \left\{(x_P, k_P^{\mathsf{pre}}, k_P, y_P) \mid \langle x_P, k_P^{\mathsf{pre}} \mid k_P \rangle \text{ is a } P\text{-query and } \langle x_P, k_P^{\mathsf{pre}} \mid y_P \rangle \text{ its answer.}\right\}$$
$$\tau_F := \left\{(x_F, k_F^{\mathsf{pre}}, y_F, z_F) \mid \langle x_F, k_F^{\mathsf{pre}} \mid y_F \rangle \text{ is an } F\text{-query and } z_F \text{ its answer.}\right\}$$
$$\tau_E := \{(x_E, r_E, z_E) \mid (x_E, r_E) \text{ is an } E\text{-query and } z_E \text{ its answer.}\}$$
$$\tau_\alpha := \left\{(x_E, r_E, \alpha_E, \alpha_E^{r_E}) \mid (x_E, r_E) \text{ is an } E\text{-query and } (\alpha_E, \alpha_E^{r_E}) \text{ its } \alpha\text{-values.}\right\}$$

From Lemma 4.1 (H-coefficients technique), Lemma 9.5 (bad events), and Theorem 9.1 (good transcripts), we will obtain the following bound on the adversarial advantage for CIVK:

Corollary 9.4

For the CIVK construction as defined in Subsection 8.1.1 where the mixing function p and the output function f are replaced with their ideal counterparts, as defined in Subsection 9.1.3, and up to ℓ_p adversarial queries per IV $x \in \mathcal{IV}$, we have that for all adversaries $\mathbf{A}$ asking at most q queries, it holds that:

$$\underset{\mathbf{A}}{\Delta}(\mathcal{O}_{\mathsf{real}}, \mathcal{O}_{\mathsf{ideal}}) \leq \frac{q}{2^{\ell_k}} + \frac{q}{2^{\ell_v} - q}.$$

Remark 9.5

Note that there is a minor glitch in the proof of CIVK in [HMKM22]. The bad events in [HMKM22] additionally contain the condition that the outputs of the corresponding E- and F-queries differ. This means that the α-values and the inputs to the F-queries may collide in the good transcripts if the corresponding outputs of the E- and F-queries are identical. The consequence is that for the analysis of the good transcripts, one cannot say that $\mathcal{A} \cap \mathcal{F} = \emptyset$. This glitch also occurred in the submitted version of this work.

By removing the condition in the bad events that the outputs of the E- and F-queries must differ, one can say that $\mathcal{A} \cap \mathcal{F} = \emptyset$. That makes the bounds in [HMKM22] worse by a factor of 2. As asymptotic security is considered, this factor is negligible. Corollary 9.4 presents the corrected bound.

9.5.1 Overview of the Bad Events

Note that the IV x is chosen by the adversary. There are two strategies for the adversary to obtain a collision between the α-values and an F-query input:

1. Guess the key prefix k^{pre} as well as a volatile internal state y correctly and ask the corresponding F-query $F\langle x, k^{\mathsf{pre}} \,|\, y\rangle$.
2. Guess the key k correctly, ask the corresponding P-query $P\langle x, k^{\mathsf{pre}} \,|\, k\rangle$ to obtain $\langle x, k^{\mathsf{pre}} \,|\, y\rangle$, and ask the corresponding F-query $F\langle x, k^{\mathsf{pre}} \,|\, y\rangle$.

If the outputs of the F-query and the output of the E-query corresponding to the colliding α-value differ, the adversary surely is in the ideal world. We will introduce two bad events that correspond to the two strategies mentioned above.

Remark 9.6

In either world, it is possible to choose k_P^{pre} and k_P such that k_P^{pre} is *not* a prefix of k_P. As the mixing function is bijective, the corresponding internal state $\langle x_P, k_P^{\mathsf{pre}} \mid y_P \rangle$ will be invalid, i.e., it will not occur during an actual encryption using CIVK. These states cannot be used to obtain a distinguishing event. We will ignore queries of this type as they yield no advantage to the adversary.

9.5.2 Bad Events

According to Section 9.1, we will define our bad events. The two bad events are similar to the ones for the CIV construction presented in Subsection 9.4.2. The main difference now is that we have to account for the key prefix.

EF-Collisions. The first distinguishing event covers a collision between a construction query and an output function query. This corresponds to an adversary correctly guessing the internal state.

Bad Event (bad_1)

There exists an E-query (x_E, r_E) and an F-query $\langle x_F, k_F^{\mathsf{pre}} \mid y_F \rangle$ such that

$$\alpha_E^{r_E} = \langle x_F, k^{\mathsf{pre}} \mid y_F \rangle.$$

Key guesses. The second bad event covers key guesses from which the adversary can obtain an internal state.

Bad Event (bad_2)

There exist the following three queries: a P-query $\langle x_P, k_P^{\mathsf{pre}} \mid k_P \rangle$, an E-query (x_P, r_E), and an F-query $\pi^{r_E}(P\langle x_P, k_P^{\mathsf{pre}} \mid k_P \rangle)$ such that:

$$(k_P^{\mathsf{pre}}, k_P) = (k^{\mathsf{pre}}, k).$$

Remark 9.7

Note that bad_2 represents a special case of bad_1. In particular, we have that:

$$P\langle x_P, k_P^{\mathsf{pre}} \mid k_P \rangle = \alpha_E \text{ and } \pi^{r_E}(P\langle x_P, k_P^{\mathsf{pre}} \mid k_P \rangle) = \alpha_E^{r_E}.$$

Considering these as separate bad events will simplify our analysis.

Also note that to successfully distinguish, the adversary further needs to observe

– $E(x_E, r_E) \neq F\langle x_F, k^{\mathsf{pre}} \,|\, y_F\rangle$ for bad_1,

– $E(x_P, r_E) \neq F(\pi^{r_E}(P\langle x_P, k_P^{\mathsf{pre}} \,|\, k_P\rangle))$ for bad_2.

We omit these to simplify the analysis of the good transcripts in Section 9.6.

9.5.3 Bounding the Bad Events

In this subsection, we will present an upper bound on the bad events introduced in Subsection 9.5.2. In particular, we will show the following lemma:

Lemma 9.5

The probability of a bad event occurring for CIVK is upper bounded by:

$$\Pr[\Theta_{\mathsf{ideal}} \in \mathsf{BadT}] \leq \frac{q}{2^{\ell_k}} + \frac{q}{2^{\ell_v} - q}.$$

Proof. Since we are in the ideal world, the answers to the E-, F-, and P-queries are independent of the secret key (k^{pre}, k). For simplicity, we will sample the secret key (k^{pre}, k) after the adversary's interaction with the oracles. Also, we will sample the values α_E for each query after the adversary's interaction with the oracles.

Key guesses. We will first consider the event bad_2. There are at most q P-queries with at most q distinct $(k_P^{\mathsf{pre}}, k_P)$. The secret key (k^{pre}, k) is sampled independently at random with regard to the uniform distribution from the set $\{0,1\}^{\ell_k}$. The probability of a collision with the secret key (k^{pre}, k) can therefore be upper bounded by $q/2^{\ell_k}$. We obtain:

$$\Pr[\mathsf{bad}_2] \leq \frac{q}{2^{\ell_k}}.$$

EF-Collisions. We will now consider the event bad_1. Since we already bounded the probability for bad_2, we will now consider $\Pr[\mathsf{bad}_1 \mid \neg\mathsf{bad}_2]$. We decided to sample α_E after the adversary's interaction with the oracles.

By conditioning on $\neg\mathsf{bad}_2$, we know that for all $x \in \mathcal{IV}$ no value $P\langle x, k^{\mathsf{pre}} \,|\, k\rangle$ has yet been sampled. This applies since there was no adversarial P-query with the correct key. This implies, as we sample the α-values after the adversary's interaction with the oracle, no α-value has yet been sampled.

To obtain a collision between an F-query and an α-value, three conditions need to apply:

1. The key prefix k^{pre} needs to be guessed correctly.

2. The F-query and the E-query, resp. the α-value, need to utilize the same IV x.

3. A collision in the volatile state needs to occur.

We will individually bound the three conditions above. Fix any F-query $F\langle x_F, k_F^{\mathsf{pre}} \mid y_F\rangle$.

1. As we sample the key k after the adversary's interaction with the oracles, a key prefix collision occurs with a probability of:

$$\Pr\left[k_F^{\mathsf{pre}} = k^{\mathsf{pre}}\right] = \ell_p^{-1}.$$

2. There are at most ℓ_p E-queries, respectively α-values, to collide with, as the E-queries are limited to ℓ_p queries per IV $x \in \mathcal{IV}$.

3. Fix some E-query (x_E, r_E) where $x_E = x_F$. Now, consider the internal state $\rho = \pi^{-r_E}\langle x_F, k^{\mathsf{pre}} | y_F\rangle$. The value $\alpha_E = P\langle x_E, k^{\mathsf{pre}} \mid k\rangle$ is sampled after the F-query. As there are at most q queries we obtain:

$$\Pr\left[P\langle x_E, k^{\mathsf{pre}} \mid k\rangle = \rho\right] \leq (2^{\ell_v} - q)^{-1}.$$

Note that there could be P-queries utilizing x_E and k^{pre} with $k_P \neq k$, and therefore we need to upper bound the amount of queries above by q. We obtain that the probability of a single fixed F-query to collide with an α-value is upper bounded by:

$$\Pr\left[k_F^{\mathsf{pre}} = k^{\mathsf{pre}}\right] \cdot \ell_p \cdot \Pr\left[P\langle x_E, k^{\mathsf{pre}} \mid k\rangle = \rho\right] = \frac{1}{\ell_p} \cdot \frac{\ell_p}{2^{\ell_v} - q} = \frac{1}{2^{\ell_v} - q}.$$

Summing up over at most q F-queries, we obtain:

$$\Pr[\mathsf{bad}_1 \mid \neg\mathsf{bad}_2] \leq \frac{q}{2^{\ell_v} - q}.$$

Lemma 9.5 follows from the union bound of the above individual bad events. ■

9.6 Good Transcripts

We have upper bounded probability of the occurrence of a distinguishing event in the ideal world in the previous four sections. In this section, we will show that in the absence of a distinguishing event, the ideal construction is indistinguishable from the real construction:

Theorem 9.1

For any good transcript $\tau \in \mathsf{GoodT}$, it holds that

$$\Pr[\Theta_{\mathsf{real}} = \tau] = \Pr[\Theta_{\mathsf{ideal}} = \tau].$$

Proof. In either world, the permutation $\mathbf{P}$ and the secret key k are sampled uniformly at random, and therefore they are trivially indistinguishable.

EE-Collisions. As the random function $\mathbf{F}$ is not sampled by E-queries in the ideal world, we still need to argue about the answers to the E- and the F-queries. We define $\mathscr{A}$ to be the *multiset* of all α^r-values, and $\mathscr{F}$ as the set of all F-query inputs that are contained in the transcript τ. Formally, $\mathscr{A}$ is defined as:

$$\mathscr{A} := \{\pi^r(P(\mathtt{load}(x,k))) \mid (x,r) \text{ is an } E\text{-query}\}.$$

Remember that, in the ideal world, the output of E is sampled independently of the α-values. Consider two E-queries $(x,r) \neq (x',r')$ with colliding α^r-values. E-queries with colliding α^r-values correspond to elements in $\mathscr{A}$ with a multiplicity greater than 1 and thus:

$$1 = \Pr_{real}\big[E(x,r) = z \mid E(x',r') = z\big] \neq \Pr_{ideal}\big[E(x,r) = z \mid E(x',r') = z\big] = 1/2.$$

We will now argue why, for good transcripts, each element in $\mathscr{A}$ has a multiplicity of 1 and thus $\mathscr{A}$ is a set. For LSSK and CKEY, all values in $\mathscr{A}$ have a multiplicity of 1, as we excluded collisions between two E-queries with the bad event bad_3 in Subsection 9.2.2 for LSSK and in Subsection 9.3.2 for CKEY. For CIV and CIVK, all values in $\mathscr{A}$ have a multiplicity of 1, as all states within a packet need to be distinct due to the requirement on the packet length to have a period of at least ℓ_p. Additionally, as the entire IV is part of the non-volatile internal state, states cannot collide for two different IVs.[3]

As no collisions occur in the α^r-values, we can say that for all E-queries (x,r) and their corresponding α^r-values $\alpha^r \in \mathscr{A}$, and for all $z \in \{0,1\}$, we have:

$$\Pr_{real}[F(\alpha^r) = z] = \Pr_{ideal}[E(x,r) = z].$$

Note that, in the real world, $E(x,r) = F(\alpha^r)$.

EF-Collisions. Collisions between E- and F-queries do not occur in the good transcripts, as these have been excluded in the bad events bad_1 and bad_2. Hence, we know that $\mathscr{A} \cap \mathscr{F} = \emptyset$. In the ideal world, the random function $\mathbf{F}$ is only evaluated on F-queries. In the real world, E-queries also evaluate $\mathbf{F}$. Since $\mathscr{A} \cap \mathscr{F} = \emptyset$, we know that for all $y \in \mathscr{F}$, the values $\mathbf{F}(y)$ are 'fresh' in the real world and have not been sampled through an E-query. The same applies in the ideal world as E-queries do not sample the random function $\mathbf{F}$.

Indistinguishable good transcripts. In particular, all random draws from $\mathscr{A} \cup \mathscr{F}$ are independent and will output 0 or 1 with equal probability in either world. We can

[3] For CIV and CIVK the multiplicity of 1 will also hold true in the bad transcripts.

conclude that for all E-queries $(x, r) \in \tau_E$ and their corresponding α^r-values $\alpha^r \in \mathscr{A}$, for all $y \in \mathscr{F}$, and for all $z, z' \in \{0,1\}$:

$$\Pr_{real}[F(\alpha^r) = z, F(y) = z'] = \Pr_{ideal}[E(x, r) = z, F(y) = z'].$$

Again, note that, in the real world, $E(x, r) = F(\alpha^r)$. ■

10 | Presenting the DRACO Stream Cipher

For the CIVK construction, a security level of the entire volatile state length was shown in the last chapter. The Draco stream cipher builds upon CIVK and uses a 128-bit volatile state size and a 128-bit non-volatile state size. The 128-bit secret key is loaded into the non-volatile register cells, and the 96-bit IV and a 32-bit key prefix are loaded into the non-volatile register cells. The packet length is 2^{32} bits.

We will first present the specification of Draco that was presented at FSE 2023 and published in ToSC 2022 [HMKM22]. Unfortunately, this version has a weakness in its key schedule that was pointed out by Banik [Ban22]. Essentially, Banik's attack exploits the fact that the IV bits and the key prefix bits are combined cyclically using the bitwise exclusive-or operation. This allows an attacker to craft IVs such that the key-IV-schedule bit is zero for several (approximately 96) clock cycles. This stream of zeros repeats periodically.

In Chapter 11, we will present possibilities on how to fix these vulnerabilities. This is currently a work in progress. The main idea is to choose the key prefix bits based on an address that is generated from one of the NFSRs.

10.1 Design Specification of DRACO

The design of Draco is similar to that of Lizard [HKM17a], which was in turn inspired by the Grain family [HJMM08] of stream ciphers. In particular, the internal state of Draco is distributed over two interconnected feedback shift registers (FSRs) as depicted in Figure 10.1.

Note, however, that while Grain uses one linear feedback shift register (LFSR) and one nonlinear feedback shift register (NFSR), which, moreover, are of the same length, Draco (like Lizard) uses two NFSRs of different lengths instead. The reasons for this design choice will be explained in Section 10.2. Like in Grain, the third important building block besides the two FSRs is a nonlinear output function, which takes inputs from both shift registers and is also used as part of the state initialization algorithm. A major difference to Grain (and also Lizard) is that Draco continuously uses the IV and (parts of) the key not

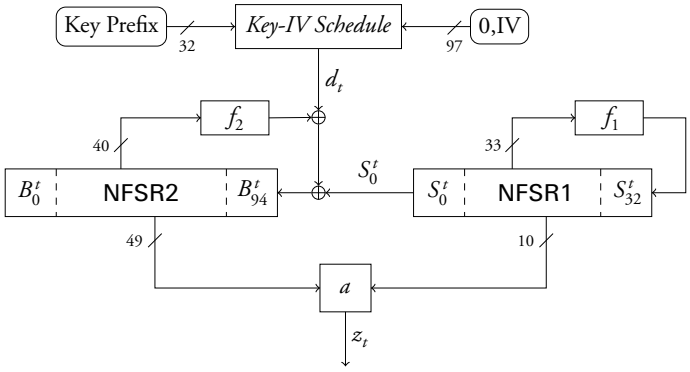

Figure 10.1: DRACO in keystream generation mode.

only during state initialization but also during keystream generation.

We describe the components of the cipher in detail in Section 10.1.1 and specify how it is operated during state initialization in Section 10.1.2 and during keystream generation in Section 10.1.3. For the sake of clarity, subsections 10.1.1–10.1.3 contain only the technical aspects of DRACO. Explanations of important design choices for our construction are given separately in Section 10.2, along with a discussion of the security properties of the particular components,e.g., the algebraic properties of the feedback functions.

10.1.1 Components

Let $K = (K_0, \ldots, K_{127})$ denote the 128-bit secret key and $IV = (IV_0, \ldots, IV_{95})$ the 96-bit public IV. The 128-bit volatile internal state of DRACO is distributed over two NFSRs, NFSR1 and NFSR2, whose contents at time $t = 0, 1, \ldots$ we denote by $(S_0^t, \ldots, S_{32}^t)$ and $(B_0^t, \ldots, B_{94}^t)$, respectively (cf. Figure 10.1). As NFSR1 and NFSR2 are Fibonacci-type, for $t \in \mathbb{N}$ it holds that $S_i^{t+1} := S_{i+1}^t$, $i = 0, \ldots, 31$, and $B_j^{t+1} := B_{j+1}^t$, $j = 0, \ldots, 93$.

Non-volatile state. Besides the 128-bit volatile internal state, DRACO additionally employs a 128-bit non-volatile internal state, which is formed by the 96-bit public IV and the first 32 bits (i.e., $K_0, \ldots, K_{31}$) of the 128-bit secret key.

Specification (*Key schedule*)

Based on the 96-bit public IV, the first 32 key bits, and the public 1-bit constant 0, in clock cycle t the *key-IV-schedule bit* (KIS bit) d_t is computed as

$$d_t := \begin{cases} x_{t \bmod 97}, & \text{for } 0 \le t \le 255, \\ K_{t \bmod 32} \oplus x_{t \bmod 97}, & \text{for } t \ge 256, \end{cases}$$

where $x_0 := 0$ and $x_i := IV_{i-1}$ for $i = 1, \ldots, 96$.

The KIS-bit d_t is fed to NFSR2 as depicted in Figure 10.1 and described more formally below.

NFSR1. DRACO's NFSR1 replaces the LFSR of the Grain family of stream ciphers. NFSR1 is 33 bits wide and corresponds (with a slight adaption, see below) to the NFSR A_{12} of the eSTREAM Phase 2 (hardware portfolio) candidate ACHTERBAHN-128/80 [GGK06]. For *all* starting states, it has a guaranteed period of 2^{33}, i.e., it is truly maximal.

Specification (***Feedback bit of NFSR1***)

NFSR1 can be specified by the following update relation (during keystream generation), defining f_1 depicted in Figure 10.1:

$$\begin{aligned} S_{32}^{t+1} := {} & S_0^t \oplus S_2^t \oplus S_7^t \oplus S_9^t \oplus S_{10}^t \oplus S_{15}^t \oplus S_{23}^t \oplus S_{25}^t \oplus S_{30}^t \oplus S_8^t S_{15}^t \oplus S_{12}^t S_{16}^t \\ & \oplus S_{13}^t S_{15}^t \oplus S_{13}^t S_{25}^t \oplus S_1^t S_8^t S_{14}^t \oplus S_1^t S_8^t S_{18}^t \oplus S_8^t S_{12}^t S_{16}^t \oplus S_8^t S_{14}^t S_{18}^t \\ & \oplus S_8^t S_{15}^t S_{16}^t \oplus S_8^t S_{15}^t S_{17}^t \oplus S_{15}^t S_{17}^t S_{24}^t \oplus S_1^t S_8^t S_{14}^t S_{17}^t \oplus S_1^t S_8^t S_{17}^t S_{18}^t \\ & \oplus S_1^t S_{14}^t S_{17}^t S_{24}^t \oplus S_1^t S_{17}^t S_{18}^t S_{24}^t \oplus S_8^t S_{12}^t S_{16}^t S_{17}^t \oplus S_8^t S_{14}^t S_{17}^t S_{18}^t \\ & \oplus S_8^t S_{15}^t S_{16}^t S_{17}^t \oplus S_{12}^t S_{16}^t S_{17}^t S_{24}^t \oplus S_{14}^t S_{17}^t S_{18}^t S_{24}^t \oplus S_{15}^t S_{16}^t S_{17}^t S_{24}^t \\ & \oplus \neg S_1^t \neg S_2^t \cdots \neg S_{30}^t \neg S_{31}^t \neg S_{32}^t. \end{aligned}$$

Note that NFSR1 of DRACO differs from NFSR A_{12} of ACHTERBAHN-128/80 only in the additional final term $\oplus \neg S_1^t \neg S_2^t \cdots \neg S_{30}^t \neg S_{31}^t \neg S_{32}^t$, which turns the period $2^{33} - 1$ of A_{12} into the truly maximal period 2^{33} for NFSR1. That is, NFSR1 of DRACO cannot get stuck in the all-zero state.

NFSR2. NFSR2 is 95 bits wide and uses a modified version of g from Grain-128a [ÅHJM11] as its feedback polynomial. More precisely, f_2 of NFSR2 (cf. Figure 10.1) shifts six taps of g by two positions to the left in order to fit a 95 bit register (i.e., tap shifts $86 \leftarrow 88$, $89 \leftarrow 91$, $90 \leftarrow 92$, $91 \leftarrow 93$, $93 \leftarrow 95$, $94 \leftarrow 96$). Moreover, four quadratic monomials and one degree-three monomial are added to further strengthen DRACO, inter alia, against algebraic attacks. This results in the following update relation (during keystream generation):

Specification (***Feedback bit of NFSR2***)

The feedback bit of NFSR2 is calculated as follows:

$$\begin{aligned} B_{94}^{t+1} := {} & S_0^t \oplus d_t \oplus B_0^t \oplus B_{26}^t \oplus B_{56}^t \oplus B_{89}^t \oplus B_{94}^t \oplus B_3^t B_{67}^t \oplus B_{11}^t B_{13}^t \\ & \oplus B_{17}^t B_{18}^t \oplus B_{27}^t B_{59}^t \oplus B_{36}^t B_{39}^t \oplus B_{40}^t B_{48}^t \oplus B_{50}^t B_{79}^t \oplus B_{54}^t B_{71}^t \\ & \oplus B_{58}^t B_{63}^t \oplus B_{61}^t B_{65}^t \oplus B_{68}^t B_{84}^t \oplus B_8^t B_{46}^t B_{87}^t \oplus B_{22}^t B_{24}^t B_{25}^t \end{aligned}$$

$$\oplus B_{70}^t B_{78}^t B_{82}^t \oplus B_{86}^t B_{90}^t B_{91}^t B_{93}^t.$$

Note that this update relation for B_{94}^{t+1} additionally contains the masking bit S_0^t from NFSR1 (analogously to the Grain family) as well as the KIS bit d_t (unlike the Grain family).

Output function a. The output function $a : \{0,1\}^{59} \longrightarrow \{0,1\}$ builds on the construction scheme introduced in [MJSC16] as part of the FLIP family of stream ciphers. Please refer to Section 10.2 for details.

Specification (*Output bit of* DRACO)

a is defined through the output bit z_t of DRACO at time t, which is computed as

$$z_t := \mathscr{L}_t \oplus \mathscr{Q}_t \oplus \mathscr{T}_t^{(1)} \oplus \mathscr{T}_t^{(2)} \oplus \mathscr{T}_t^{(3)},$$

where

$$\begin{aligned}
\mathscr{L}_t &:= B_7^t \oplus B_{15}^t \oplus B_{32}^t \oplus B_{47}^t \oplus B_{66}^t \oplus B_{80}^t \oplus B_{92}^t,\\
\mathscr{Q}_t &:= B_5^t B_{85}^t \oplus B_{12}^t B_{74}^t \oplus B_{20}^t B_{69}^t \oplus B_{34}^t B_{57}^t,\\
\mathscr{T}_t^{(1)} &:= B_{53}^t \oplus B_{38}^t B_{44}^t \oplus B_{23}^t B_{49}^t B_{83}^t \oplus B_6^t B_{33}^t B_{51}^t B_{73}^t\\
&\quad \oplus B_4^t B_{29}^t B_{43}^t B_{60}^t B_{81}^t \oplus B_9^t B_{14}^t B_{35}^t B_{42}^t B_{55}^t B_{77}^t\\
&\quad \oplus B_1^t B_{16}^t B_{28}^t B_{45}^t B_{64}^t B_{75}^t B_{88}^t,\\
\mathscr{T}_t^{(2)} &:= S_{26}^t \oplus S_5^t S_{19}^t \oplus S_{11}^t S_{22}^t S_{31}^t,\\
\mathscr{T}_t^{(3)} &:= B_{76}^t \oplus S_3^t B_{10}^t \oplus S_{20}^t B_{21}^t B_{30}^t \oplus S_6^t S_{29}^t B_{62}^t B_{72}^t.
\end{aligned}$$

10.1.2 State Initialization

The state initialization process can be divided into two phases, first key loading and then grain-like mixing, and is similar to the classical Grain-type initialization.

Phase 1, key loading. The first phase of the initialization consists of loading the key into the registers.

Specification (*Key loading*)

The registers of the keystream generator are initialized as follows:

$$B_j^0 := \begin{cases} K_j \oplus 1, & \text{for } j = 0, \\ K_j, & \text{for } j \in \{1, \ldots, 94\}, \end{cases}$$
$$S_i^0 := K_{i+95}, \qquad \text{for } i \in \{0, \ldots, 32\}.$$

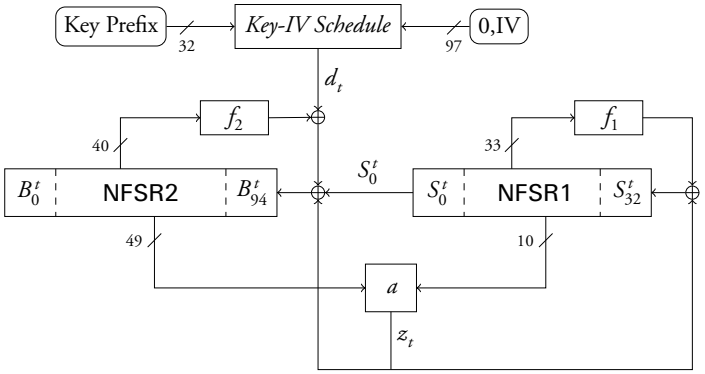

Figure 10.2: Draco in phase 2 of the state initialization.

Phase 2, Grain-like mixing. Clock the cipher 512 times without producing actual keystream. Instead, at time $t = 0, \ldots, 511$, the output bit z_t is fed back into both FSRs as depicted in Figure 10.2. To avoid ambiguity, we now give the full update relations that will be used for NFSR2 and NFSR1 in phase 2.

Specification (*Grain-like mixing*)

For $t = 0, \ldots, 511$, compute

$$B_j^{t+1} := B_{j+1}^t \quad \text{for } j \in \{0, \ldots, 93\},$$
$$B_{94}^{t+1} := z_t \oplus S_0^t \oplus d_t \oplus B_0^t \oplus B_{26}^t \oplus B_{56}^t \oplus B_{89}^t \oplus B_{94}^t \oplus B_3^t B_{67}^t \oplus B_{11}^t B_{13}^t \\ \oplus B_{17}^t B_{18}^t \oplus B_{27}^t B_{59}^t \oplus B_{36}^t B_{39}^t \oplus B_{40}^t B_{48}^t \oplus B_{50}^t B_{79}^t \oplus B_{54}^t B_{71}^t$$

$$
\begin{aligned}
&\oplus B_{58}^t B_{63}^t \oplus B_{61}^t B_{65}^t \oplus B_{68}^t B_{84}^t \oplus B_8^t B_{46}^t B_{87}^t \oplus B_{22}^t B_{24}^t B_{25}^t \\
&\oplus B_{70}^t B_{78}^t B_{82}^t \oplus B_{86}^t B_{90}^t B_{91}^t B_{93}^t, \\
S_i^{t+1} &:= S_{i+1}^t \quad \text{for } i \in \{0,\ldots,31\}, \\
S_{32}^{t+1} &:= z_t \oplus S_0^t \oplus S_2^t \oplus S_7^t \oplus S_9^t \oplus S_{10}^t \oplus S_{15}^t \oplus S_{23}^t \oplus S_{25}^t \oplus S_{30}^t \oplus S_8^t S_{15}^t \oplus S_{12}^t S_{16}^t \\
&\oplus S_{13}^t S_{15}^t \oplus S_{13}^t S_{25}^t \oplus S_1^t S_8^t S_{14}^t \oplus S_1^t S_8^t S_{18}^t \oplus S_8^t S_{12}^t S_{16}^t \oplus S_8^t S_{14}^t S_{18}^t \\
&\oplus S_8^t S_{15}^t S_{16}^t \oplus S_8^t S_{15}^t S_{17}^t \oplus S_{15}^t S_{17}^t S_{24}^t \oplus S_1^t S_8^t S_{14}^t S_{17}^t \oplus S_1^t S_8^t S_{17}^t S_{18}^t \\
&\oplus S_1^t S_{14}^t S_{17}^t S_{24}^t \oplus S_1^t S_{17}^t S_{18}^t S_{24}^t \oplus S_8^t S_{12}^t S_{16}^t S_{17}^t \oplus S_8^t S_{14}^t S_{17}^t S_{18}^t \\
&\oplus S_8^t S_{15}^t S_{16}^t S_{17}^t \oplus S_{12}^t S_{16}^t S_{17}^t S_{24}^t \oplus S_{14}^t S_{17}^t S_{18}^t S_{24}^t \oplus S_{15}^t S_{16}^t S_{17}^t S_{24}^t \\
&\oplus \neg S_1^t \neg S_2^t \cdots \neg S_{30}^t \neg S_{31}^t \neg S_{32}^t,
\end{aligned}
$$

where z_t and d_t are computed as described in Subsection 10.1.1.

10.1.3 Keystream Generation

At the end of the state initialization, the 33-bit (initial) state of NFSR1 is $\left(S_0^{512},\ldots,S_{32}^{512}\right)$ and the 95-bit (initial) state of NFSR2 is $\left(B_0^{512},\ldots,B_{94}^{512}\right)$. The first keystream bit that is used for plaintext encryption is z_{512}. For $t \geq 512$, the states $\left(S_0^{t+1},\ldots,S_{32}^{t+1}\right)$ and $\left(B_0^{t+1},\ldots,B_{94}^{t+1}\right)$ and the output bit z_t are computed using the relations given in Subsection 10.1.1. Figure 10.1 depicts the structure of Draco during keystream generation.

Packet mode. As Draco is designed to be operated in packet mode, the maximum size of a plaintext packet encrypted under the same key-IV pair is 2^{32} bits and no key-IV pair may be used more than once, i.e., for more than one packet. Let $X = \left(x_0,\ldots,x_{|X|-1}\right)$ denote such a plaintext packet and let $z_{512}, z_{513},\ldots$ be the keystream generated for it as described before. Then the corresponding ciphertext packet $Y = \left(y_0,\ldots,y_{|X|-1}\right)$ can be produced via $y_i := x_i \oplus z_{i+512}$, $i = 0,\ldots,|X|-1$. Decryption (given that the secret session key and the public IV are known) works analogously.

Note that, though we use the terms plaintext/ciphertext packet here, Draco is really a (synchronous) stream cipher, i.e., the keystream bits $z_{512}, z_{513},\ldots$ are generated in a bitwise fashion (and independently of the plaintext/ciphertext) and, consequently, the individual plaintext bits x_i can be encrypted and then,in the form of y_i, transmitted as they arrive. The same obviously holds for the decryption of the ciphertext bits y_i.

10.2 Design Considerations

In this section, we provide additional explanations with regard to our design, which were omitted in Section 10.1 for the sake of clarity. As Draco has a Grain-like structure, we particularly focus on respective similarities and differences. Based on several of the follow-

ing properties, we will then argue in Section 10.3 why we believe that DRACO resists the currently known types of attacks against stream ciphers.

When describing the properties of the NFSRs and the output function, we will use standard terminology like *nonlinearity*, *order of correlation immunity*, *diffusion parameter*, etc. We refer the reader to [MJSC16, CCH10] for the corresponding definitions and explanations of these concepts.

10.2.1 The Key-IV Schedule

The key-IV schedule defined in Subsection 10.1.1 is designed in such a way that for all $t \geq 0$ the mapping $D_t : \{0,1\}^{32} \times \{0,1\}^{96} \longrightarrow \{0,1\}^{128}$, defined as

$$D_t(K_0, \ldots, K_{31}, IV_0, \ldots, IV_{95}) := (d_t, d_{t+1}, \ldots, d_{t+127})$$

is a bijective GF(2)-linear mapping. This follows directly from Lemma 10.1.

Lemma 10.1

Fix some $r \in \{0, \ldots, 96\}$. The system of GF(2)-linear equations in the variables $u_0, \ldots, u_{31}, v_0, \ldots, v_{96}$

$$v_r = 0 \tag{1}$$
$$u_0 \oplus v_0 = u_1 \oplus v_1 = \cdots = u_{31} \oplus v_{31} = 0 \tag{2}$$
$$u_0 \oplus v_{32} = \cdots = u_{31} \oplus v_{63} = 0 \tag{3}$$
$$u_0 \oplus v_{64} = \cdots = u_{31} \oplus v_{95} = 0 \tag{4}$$
$$u_0 \oplus v_{96} = u_1 \oplus v_0 = \cdots = u_{31} \oplus v_{30} = 0. \tag{5}$$

has only one solution: $u_0 = \cdots = u_{31} = v_0 = \cdots = v_{96} = 0$.

This system corresponds the situation that t is chosen in such a way that the constant 0 is added at position $t + r$, which is the case if $r \equiv 97 - t \mod 97$. Moreover, $(u_0, \ldots, u_{31})$ is obtained from $(K_0, \ldots, K_{31})$ by some cyclic shift, and $(v_0, \ldots, v_{96})$ is obtained from $(x_0, \ldots, x_{96})$ by a cyclic right shift by r positions.

Proof. Note that $u_s = 0$ for $s = r \mod 32$ follows from $v_r = 0$. This implies $v_{s-1} = 0$ (by (5)) and $u_{s-1} = 0$ (by (2)) and $v_{s-2} = 0$ (by (5)) and $u_{s-2} = 0$ (by (2)) and so on, i.e., $u_s = u_{s-1} = \cdots = u_0 = 0$. On the other hand, $v_s = 0$ (by (2)) which implies $u_{s+1} = 0$ (by (5)) which implies $v_{s+1} = 0$ (by (2)) and so on, i.e., $u_s = u_{s+1} = \cdots = u_{31} = 0$. As all u-bits are zero, by (2-5), all v-bits have to be zero, too. ∎

Note that the bijectivity of D_t makes DRACO immune against the following type of chosen-IV TMDTO-attacks, which we call zero d-stream attacks.

Zero d-stream attacks. Let $\mathscr{K}(0) \subseteq \{0,1\}^{32}$ be the set of all key prefixes k^{pre} for which there is some initial value $IV \in \{0,1\}^{96}$ such that $d_t(k^{\mathsf{pre}}, IV) = 0$ for all $t \geq 0$. Correspondingly, let $\mathscr{IV}(0) \subseteq \{0,1\}^{96}$ be the set of all initial value $IV \in \{0,1\}^{96}$ for which there is a key prefix $k^{\mathsf{pre}} \in \{0,1\}^{32}$ such that $d_t(k^{\mathsf{pre}}, IV) = 0$ for all $t \geq 0$. For all volatile internal states $S \in \{0,1\}^{128}$ let $Z_{\vec{0}}(S)$ denote the block of the first 128 keystream bits generated on S under the condition that the corresponding 128 d-stream bits are all zero.

Offline

1. The attacker computes $\mathscr{K}(0)$ and $\mathscr{IV}(0)$
2. The attacker computes $Z_{\vec{0}}(S)$ for 2^{96} randomly and mutually independently chosen internal states $S \in \{0,1\}^{128}$ and stores them in a database $\mathscr{D}$.

Online

3. The attacker asks for the keystream packet $P(k^*, IV) \in \{0,1\}^{32}$, $k^* \in \{0,1\}^{128}$ the secret key, for all initial values $IV \in \mathscr{IV}(0)$ and checks if $P(k^*, IV) \in \{0,1\}^{32}$ contains a sub block D of length 128 which occurs in $\mathscr{D}$.
4. In the case that a collision $D = Z_{\vec{0}}(S)$ was found, compute k^* from S.

Note that, if $k^* \in \mathscr{K}(0)$ then, in step 3, the attacker knows the packet $P(k^*, IV^*)$ for some $IV^* \in \mathscr{IV}(0)$. This packet contains 2^{32} keystream blocks $Z_{\vec{0}}(S')$ for the internal state $S' \in \{0,1\}^{128}$. By the birthday paradox, there will be a collision with $\mathscr{D}$ which yields the secret key. If $k^* \notin \mathscr{K}(0)$ then the attack fails. Consequently, the success probability of the attack is around $|\mathscr{K}(0)| \cdot 2^{-32}$, while the cost is at least 2^{96}.

Due to the fact that D_t is bijective for all $t \geq 0$ (see Subsection 10.2.1) we have that $|\mathscr{K}(0)| = |\mathscr{IV}(0)| = 1$, which implies that the zero d-stream attack against Draco does not beat exhaustive key search.

10.2.2 NFSR1

NFSR1 has a width of 33 bits and replaces the maximum-length LFSR of the Grain family. Considering the reduced size of the cipher's volatile internal state (128 bits for Draco compared to 256 bits for Grain-128a), the use of an NFSR is advantageous to enhance the design's resistance against algebraic and fast correlation attacks such as [TIM+18]. Unfortunately, there is limited knowledge on the generic construction of large, cryptographically strong NFSRs with maximal period. Nevertheless, for FSR sizes ranging from 30 to 40 bits, corresponding non-linear feedback functions can be identified through experimentation. In [GGK06], the designers of the eSTREAM Phase 2 (hardware portfolio) candidate ACHTERBAHN-128/80 provide a collection of 13 such NFSRs, varying in size from 21 bits (NFSR A_0) to 33 bits (NFSR A_{12}). The latter size is perfectly adequate for our design

since, given the packet mode limitation with a maximum of 2^{32} keystream bits per key-IV pair, DRACO does not require as large guaranteed periods as the Grain family.

All-zero state. While in the original Grain family, an initial all-zero state for the LFSR is acceptable due to the sizes of the FSRs (both of key length), it must be strictly avoided for *small-state* Grain-like stream ciphers. Specifically, for one out of every 2^{33} key-IV combinations, the mixing process used by DRACO during state initialization will result in an all-zero state for NFSR1. In such a scenario, NFSR A_{12} of ACHTERBAHN would get stuck in the all-zero state because, similar to a maximum-length LFSR of this size, it only has a period of $2^{33}-1$ for *non-zero* starting states. In contrast, DRACO's NFSR1 achieves a truly maximal period of 2^{33} for *any* starting state. This is accomplished by introducing the term $\oplus\neg S_1^t \neg S_2^t \cdots \neg S_{31}^t \neg S_{32}^t$ to ACHTERBAHN's NFSR A_{12} as described in Section 10.1.1, effectively 'gluing' the all-zero state into the original state cycle of NFSR A_{12} between the states $(1,0,\ldots,0)$ and $(0,\ldots,0,1)$.

Properties. Therefore, we can evaluate the security of our NFSR1 based on the following properties of A_{12} given in [GGK06]: a nonlinearity of 114688, a order of correlation immunity of 6, and a diffusion parameter[1] of 54. Additionally, it is evident that the feedback function of A_{12} is balanced and, thus (as it is 6th order correlation-immune), 6-resilient. For comparison, in Grain-128a, the feedback function of the LFSR (which is replaced by NFSR1 in DRACO) has a resiliency of 5. Finally, the algebraic degree of the feedback function of A_{12} is 4.

Efficiency. Another important factor in selecting A_{12} from ACHTERBAHN as the basis for DRACO's NFSR1 is its hardware efficiency. Despite having a relatively large algebraic normal form, the designers of ACHTERBAHN have managed to provide a compact hardware implementation of A_{12}'s feedback function, which consumes only 31.75 gate equivalents (GE) and has a logical depth of three (for more information regarding hardware complexity and the explanation of measurement units such as GE, please refer to Section 10.4).

10.2.3 NFSR2

NFSR2 is 95 bits wide and its feedback polynomial is a modified version of g from Grain-128a [ÅHJM11]. In contrast to NFSR1, the period of NFSR2 during keystream generation is unknown because even after state initialization, it is not operated in a self-contained manner. More precisely, due to the masking bit from NFSR1 and the KIS bit d_t, NFSR2

[1] The diffusion parameter λ was determined experimentally in [GGK06] and denotes 'the minimum number of clock cycles needed in order to transform any two initial states of the shift register A_j of Hamming distance 1 into shift register states of Hamming distance close to $N_j/2$' (N_j denotes the size of the shift register A_j).

is actually a filter instead of a real NFSR (cf. corresponding remark for the Grain family in [HJMM08]).

Modifying g. As detailed in Section 10.1.1, for f_2 of Draco, we shifted six taps of g from Grain-128a two positions to the left, while ensuring that the property of g where no tap appears more than once is preserved in f_2. Additionally, we introduced the following new monomials to further strengthen Draco despite its smaller volatile internal state: $B^t_{36}B^t_{39}$, $B^t_{50}B^t_{79}$, $B^t_{54}B^t_{71}$, $B^t_{58}B^t_{63}$, $B^t_8B^t_{46}B^t_{87}$. It is worth noting that the tap indices of these new nonlinear monomials are disjoint from the tap indices of all other monomials in f_2. Consequently, important properties of g in Grain-128a, such as its balancedness and its resiliency of 4, carry over to f_2. Similarly, the security of f_2 in Draco against linear approximations can be lower bounded by that of Grain-128a (which has 2^{14} best linear approximations with a bias of $63 \cdot 2^{-15}$). As proven in [MJSC16], the aforementioned disjointness property with respect to the newly added taps implies that the nonlinearity of f_2 in Draco can be calculated based on the nonlinearity of g in Grain-128a (267403264) and its number of variables (29), along with the nonlinearity of the sum of the new monomials (976) and its number of variables (11), as follows: $2^{11} \cdot 267403264 + 2^{29} \cdot 976 - 2 \cdot 267403264 \cdot 976 = 549656723456$ (approximately 2^{39}). Similar to g in Grain-128a, the algebraic degree of f_2 in Draco is 4.

In addition to these properties, g has successfully thwarted all cube-like attacks targeting the initialization of Grain-128a. In Section 10.3.3, we derive corresponding security arguments for Draco.

10.2.4 Output Function a

In FSR-based stream cipher design, an important consideration is how to distribute the responsibility of ensuring security between the driving register(s) and the output function. To compensate for the fact that the volatile internal state of Draco is smaller than that of Grain-128a, we decided that the output function should have more inputs and larger algebraic degree instead. The construction scheme used for Draco's output function a is based on the approach introduced in [MJSC16] as part of the FLIP family of stream ciphers. Specifically, Draco's output function a can be expressed as the direct sum of a linear function with seven monomials, a quadratic function with four monomials, a triangular function with seven monomials, another triangular function with three monomials, and a final triangular function with four monomials. Each tap of NFSR1 and NFSR2 appears at most once in a.

Properties. As a result, the output function of Draco is defined over 59 variables, is balanced, and possesses the following security properties based on lemmata 3–6 in [MJSC16]: a nonlinearity of 287580136809693184 (approximately 2^{57}), a resiliency of 9, an algebraic

immunity of at least 7, and a fast algebraic immunity of at least 8. The algebraic degree of a is 7.

Guessing NFSR1. In the scenario where the attacker has knowledge of the content of NFSR1 at time t (e.g., as part of a guess-and-determine attack), the output function still relies on a minimum of 44 variables and 'gracefully degrades' into the direct sum of a linear function with eight or nine (depending on the value of S_3^t) monomials, a quadratic function with four to six (depending on the values of S_{20}^t and $S_6^t S_{29}^t$) monomials, and a triangular function with seven monomials. This conforms to the construction principle introduced in [MJSC16] and results in the following worst-case security properties for that situation: a nonlinearity of 8634823344128 (approximately 2^{42}), a resiliency of 8, an algebraic immunity of at least 7, and a fast algebraic immunity of at least 8.

Tap positions. While the selection of tap positions for state update functions is often restricted by the requirement to ensure a certain period (e.g., as seen in the case of NFSR1), the choice of tap positions for an output function is typically less rigorously justified. The design documents introducing members of the Grain family (e.g., [HJM06, ÅHJM11, HJMM08]) primarily focus on the conceptual decision of whether certain taps used in the output function should originate from the NFSR or the LFSR (and the quantity of taps from each). The more specific question of *which* tap positions within each FSR are actually selected for the output function is mostly discussed in the context of hardware acceleration or when addressing issues that arise from actual attacks on previous versions (e.g., the attack by Dinur and Shamir [DS11] on Grain-128, which led to a modification of tap positions in the output function of Grain-128a [ÅHJM11]).

Positive difference sets. In the absence of canonical criteria for selecting tap positions in Grain-like constructions, we primarily rely on the concept of (full) positive difference sets introduced by Golić in [Gol96]. This concept was originally used to evaluate the security of nonlinear filter generators consisting of a single LFSR and a nonlinear output function. Notably, the fast correlation attacks against Grain v1 and Grain-128a described in [TIM+18] align with this cipher model as they treat Grain's NFSR as (part of) a filter for its LFSR. For more information on Golić's findings and their influence on DRACO's output function, please refer to Subsection 10.2.5.

Binary decision diagrams. Another factor guiding our selection of tap positions for DRACO's output function is the consideration of attacks based on binary decision diagrams (BDDs). This type of attack, introduced by Krause in [Kra02] and applied to Grain-128 by Stegemann in [Ste07], highlights the importance of having a significant difference in tap indices between the lowest and highest taps in each monomial. Section 10.3.6 provides more detailed information on this topic.

10.2.5 Output Function a: Tap Selection

As mentioned in Subsection 10.2.4, the selection of tap positions for Grain-like constructions lacks canonical criteria. Therefore, we primarily rely on the concept of (full) positive difference sets introduced by Golić in [Gol96] to evaluate the security of nonlinear filter generators that consist of a single LFSR and a nonlinear output function. A similar approach has been employed in other stream ciphers, such as Espresso [DH15] and Lizard [HKM17a], which are based on NFSR and have Grain-like characteristics.

Golić defines 'for a positive integer λ, call Γ a λth-order positive difference set if λ is the maximum number of pairs of its elements with the same mutual difference (for $\lambda = 1$, we get a full positive difference set)' [Gol96] and, as a security criterion for output functions, requires that the taps 'should be chosen according to a full or a λth-order positive difference set, with λ as small as possible' [Gol96].

Properties. In line with this, the output function a of Draco has the following properties:

- No taps from NFSR1, except some of those in the additionally required term of f_1, $\oplus\neg S_1^t \neg S_2^t \cdots \neg S_{31}^t \neg S_{32}^t$ (cf. Section 10.1.1), are used at the same time for its feedback function f_1 and the output function. (In Grain-128a, the feedback function of the LFSR, which corresponds to NFSR1 in our construction, and the output function do not share any taps, either.)

- The set
$$\{5, 11, 19, 22, 26, 31\}$$
of the tap indices (all from NFSR1) of $\mathscr{T}_t^{(2)}$ is a full positive difference set. This means that each two bits of the internal bitstream of NFSR1 never appear more than once together as part of this triangular function.

- No taps from NFSR2 are used at the same time for its feedback function and the output function. (In Grain-128a, the feedback function of the NFSR, which corresponds to NFSR2 in our construction, and the output function share only a single tap called 'b_{i+95}' in [ÅHJM11].)

- The direct sum $\mathscr{L}_t + \mathscr{Q}_t + \mathscr{T}_t^{(1)}$ uses only taps from NFSR2. (To maintain a sufficient security level even when the content of the smaller NFSR1 is known to the attacker, e.g., due to guessing; cf. Section 10.3.2.)

- The set
$$\{7, 15, 32, 47, 53, 66, 76, 80, 92\}$$
of the tap indices (all from NFSR2) of the linear monomials of $\mathscr{L}_t + \mathscr{T}_t^{(1)} + \mathscr{T}_t^{(3)}$ is a full positive difference set.

– The set

$$\{5, 12, 20, 34, 38, 44, 57, 69, 74, 85\}$$

of the tap indices (all from NFSR2) of the quadratic monomials of $\mathcal{Q}_t + \mathcal{T}_t^{(1)}$ is a full positive difference set. One consequence of this is that each two bits of the internal bitstream of NFSR2 can form at most once a quadratic monomial together.

– The sets

$$\{|5-85|, |12-74|, |20-69|, |34-57|, |38-44|\}$$

of differences between the two taps from NFSR2 of each quadratic monomial in $\mathcal{Q}_t + \mathcal{T}_t^{(1)}$ and

$$\{|3-67|, |11-13|, |17-18|, |27-59|, |36-39|, |40-48|,$$
$$|50-79|, |54-71|, |58-63|, |61-65|, |68-84|\}$$

of differences between the two taps from NFSR2 of each quadratic monomial in the feedback function of NFSR2 are disjoint. Hence, even during phase 2 of the state initialization, each two bits of the internal bitstream of NFSR2 can form at most once a quadratic monomial together.

– None of the differences

$$\{|5-85|, |12-74|, |20-69|, |34-57|, |38-44|\}$$

between the two taps (all from NFSR2) of each quadratic monomial in $\mathcal{Q}_t + \mathcal{T}_t^{(1)}$ appears as a difference between two taps of a higher degree monomial of $\mathcal{T}_t^{(1)}$.

– Each of the sets

$$\{23, 49, 83\},$$
$$\{6, 33, 51, 73\},$$
$$\{4, 29, 43, 60, 81\},$$
$$\{9, 14, 35, 42, 55, 77\},$$
$$\{1, 16, 28, 45, 64, 75, 88\}$$

of tap indices (all from NFSR2) of the monomials of degree $3, \ldots, 7$ of $\mathcal{T}_t^{(1)}$ is a full positive difference set. Consequently, each two bits of the internal bitstream of NFSR2 never appear more than once together as part of each (i.e., the same) of those monomials.

10.2.6 Continuous Key and IV Usage

In this subsection, we provide further details about how the generic CIVK construction introduced in Subsection 8.1.1 is instantiated concretely through Draco. In particular, we argue about the choice of the different parameter lengths. With Draco we want to achieve a security level of 128 bits. Accordingly, the key length is chosen to be 128 bits.

Parameter lengths. As Draco is designed to operate in packet mode and a key-IV pair may be used to generate at most 2^{32} keystream bits, the key prefix length is $\log_2 2^{32} = 32$ bits. This also determines the length of the initial value: As the desired security level is 2^{128} and the packet length is 2^{32} bits, we need to set the length of the initial value to be $128 - 32 = 96$ bits. This corresponds to a total volatile internal state size of 128 bits and a total non-volatile internal state size of 128 bits.

Mixing phase. It is easy to see that Draco's mixing phase in fact realizes a bijection (between the 256-bit internal states at $t = 0$ and those at $t = 512$) as assumed by our security proof for the CIVK construction given in Section 9.5. Draco deviates from the generic construction scheme only in a tiny detail. During key loading at $t = 0$, the first key bit is inverted (i.e., $B_0^0 := K_0 \oplus 1$). This is to avoid the 'sliding property' of Grain v1 and Grain-128 that was pointed out in [DCKP08] (see Section 10.3.4 for further details). In terms of our TMDTO security proof, however, this modification is completely irrelevant.

10.3 Cryptanalysis

In the following subsections, we will argue for several types of attacks which weakened or even broke other stream ciphers in the past, why we believe that Draco will resist them. In Section 8.2 we already argued about time-memory-data tradeoff attacks against the CIVK construction that underlies Draco.

The discussion in this section will build on a variety of results that illuminate the security of Grain-v1 against these attacks, and that address a number of established security parameters. Since Grain and Draco are very similar in the structural elements relevant to algebraic and correlation attacks, it seems sufficient to us in the given context to describe the extent to which Draco meets or exceeds the design criteria relevant to Grain's security. Of course, this does not mean provable security against every conceivable variant of such attacks. But establishing a more extended and detailed formal framework for formally proving the security of grain-like ciphers would have to go beyond the existing state of the art and would thus be a challenging scientific project in its own right, clearly beyond the scope of this paper. We think that further development of this formal framework should come from new attacks ideas being published from within the scientific community.

10.3.1 Correlation Attacks, Linear Approximations

Correlation attacks and fast correlation attacks, as described in the publications [MJSC16, TIM+18, ZGM17, TMA20, WLLM19], pose a significant threat to Grain-like stream ciphers. These attacks specifically focus on the LFSR of the cipher and aim to identify sufficiently biased linear approximations of the NFSR's feedback and the output function. In the Draco cipher, the LFSR of Grain is substituted with NFSR1, which exhibits high nonlinearity and correlation immunity. Additionally, the output function of Draco incorporates a greater number of inputs, increased resiliency, and significantly higher nonlinearity compared to Grain-128a. Moreover, the feedback function of NFSR2 in Draco has been further strengthened, as explained in Section 10.2.3.

In [MJSC16], Méaux et al. highlight the significance of 'good balancedness, non-linearity, and resiliency properties' of the filtering function to withstand correlation attacks [Sie85] and fast correlation attacks [MS89]. As explained in Section 10.2.4, Draco incorporates a relatively complex output function to compensate for the smaller volatile part of the internal state when compared to the original Grain family. The output function of Draco is defined over 59 variables and exhibits a nonlinearity of approximately 2^{57}, whereas the output function of Grain-128a is defined over 17 variables with a nonlinearity of 61440. Additionally, the resiliency of Draco's output function is 9, whereas that of Grain-128a is 7.

In [BGM06], Berbain, Gilbert and Maximov present an attack on Grain v0 that combines linear approximations of the NFSR's feedback function and of the output function in order to recover the initial state of the LFSR given a sufficient amount of keystream bits. As possible countermeasures, Berbain, Gilbert and Maximov proposed the following modifications [BGM06]: 'Introduce several additional masking variables from the NFSR in the keystream bit computation', 'replace g by a 2-resilient function', 'modify the filtering function h in order to make it more difficult to approximate' and 'modify the function g and h to increase the number of inputs'. For Grain-128a, the feedback function g of the NFSR was constructed with the above attack in mind. The designers state: 'The best linear approximation of g is of considerable interest, and for it to contain many terms, we need the resiliency of the function g to be high. We also need a high nonlinearity in order to obtain a small bias.' As a consequence, g was chosen such that it has nonlinearity 267403264 ($\approx 2^{28}$) and resiliency 4.

As explained in Section 10.2.3, the feedback function f_2 of NFSR2 in Draco builds on that of Grain-128a in a way that preserves its balancedness and resiliency, but features an even higher nonlinearity ($\approx 2^{39}$). Moreover, in accordance with the above suggestions from [BGM06] and the construction principle underlying g of Grain-128a (see previous paragraph), the output function of Draco has more than three times as many inputs, a much higher nonlinearity and a higher resiliency than that of Grain-128a (cf. values at the beginning of this subsection) in order to strengthen it against linear approximations.

In particular, it is defined over the state variables of both FSRs, featuring monomials of all degrees between one and seven defined over NFSR2 (cf.triangular function $\mathcal{T}_t^{(1)}$ in Subsection 10.1.1), monomials of degrees one, two, and three over NFSR1 (cf.triangular function $\mathcal{T}_t^{(2)}$), and monomials of degrees two, three, and four with variables from both FSRs mixed (cf.triangular function $\mathcal{T}_t^{(3)}$).

It should be noted that (fast) correlation attacks against Grain-like structures, as published in [MJSC16, TIM+18, ZGM17, TMA20, WLLM19], primarily target the ciphers' LFSR. However, in the case of Draco, as explained in Section 10.2.2, the LFSR of the Grain family is replaced by an NFSR with high nonlinearity and correlation immunity. This approach has already been utilized in Lizard, making it the only Grain-like stream cipher that remains completely unaffected by (fast) correlation attacks thus far.

In their work [TMA20], Todo, Meier, and Aoki investigate the data limitation of small-state stream ciphers in the context of correlation attacks. For the Plantlet cipher, which aims for 80-bit security, they demonstrate that the secret key can be recovered if approximately 2^{53} keystream bits per key-IV combination are available. Fortunately, this condition cannot be met in practice as the cipher's designers set a corresponding limit of 2^{30} bits, which is considered 'conservative' by Todo, Meier and Aoki. Based on these findings, the designers of Draco (which has larger key and state sizes compared to Plantlet) have imposed a maximum packet length of 2^{32} bits, aligning with the recommended limitations.

Finally, as explained in Section 10.2.4, the choice of tap positions for Draco's output function follows the concept of (full) positive difference sets, which was introduced by Golić in [Gol96] as a design criterion to strengthen nonlinear filter generators against correlation attacks.

10.3.2 Algebraic Attacks

Algebraic attacks against Draco can be classified into two primary approaches. The first approach involves representing the observed keystream bits as functions of the unknown 128 key bits and attempting to solve the resulting system of equations. However, this method requires including all state transitions down to $t = 0$. Considering the nonlinearity of both FSRs and the high algebraic degree of the output function (which is employed in phase 2 of the state initialization), this approach is evidently more complex compared to the second attack approach: expressing the observed keystream bits as functions of the unknown 128 bits of the volatile initial state at $t = 512$ and the unknown 32-bit key prefix (which is continuously used during keystream generation), and then solving the corresponding system of equations. Therefore, in the subsequent part of this subsection, we will concentrate on the second approach.

First of all, note that, to the best of our knowledge, no successful (i.e., having complexity lower than 2^{128}) algebraic attacks that can recover arbitrary initial states for Grain-128a

have been reported so far.[2] Due to the smaller volatile internal state of Draco, the corresponding system of equations in such an attack would, in fact, involve a lower number of variables. However, this reduction is compensated by the larger degree of the output function, which is now 7 compared to 3 for Grain-128a. As emphasized in Section 10.2.4, Draco's output function follows the construction scheme introduced in [MJSC16]. It depends on 59 variables, exhibits a nonlinearity of approximately 2^{57}, an algebraic immunity of at least 7, and a fast algebraic immunity of at least 8. Moreover, both FSRs are now nonlinear, and NFSR1, corresponding to the LFSR of the original Grain family, has an algebraic degree of 4. Additionally, we have hardened NFSR2 against algebraic attacks by incorporating five additional nonlinear monomials compared to g in Grain-128a (cf. Section 10.2.3). Considering these properties, we expect that algebraic attacks against full Draco will not have a complexity lower than that of exhaustive key search.

Guessing NFSR1. It is important to note that even when guessing the shorter NFSR1, Draco's output function still relies on at least 44 variables and possesses the following worst-case security properties: nonlinearity of approximately 2^{42}, algebraic immunity of at least 7, and fast algebraic immunity of at least 8. For comparison, the complete output function of Grain-128a is dependent on 17 variables, exhibits a nonlinearity of 61440, and has an algebraic degree of 3. Therefore, in the context of a corresponding guess-and-determine attack, an algebraic attack on NFSR2 similar to the one described in [BGJ09] will have a large enough complexity.

Guessing NFSR2. Guessing the larger NFSR2 is even less promising from an attacker's perspective. Due to its size of 95 bits and the 128-bit security level targeted by Draco, a successful state-recovery attack against the full cipher would need to subsequently recover the 33-bit internal state of NFSR1 and the 32-bit key prefix used in the key-IV schedule, with a time-memory-data complexity below $2^{128-95} = 2^{33}$. Notably, these 65 remaining unknowns also influence the further state updates of NFSR2, which, unlike NFSR1, does not operate autonomously during keystream generation. Therefore, while guessing NFSR1 allows for its *elimination* (including its feedback function and its contribution to the output function) from subsequent cryptanalysis steps, this is not the case when guessing NFSR2. In other words, an attacker guessing NFSR2 would need to recover 65 unknowns with a time-memory-data complexity below 2^{33}, while still being faced with both feedback functions and the complete output function of Draco.

[2] The currently best result seems to be an algebraic attack by Berbain, Gilbert and Joux against a modified version of Grain-128, which requires 2^{115} computations and 2^{39} keystream bits [BGJ09]. They point out, however, that '[t]his attack is not applicable to the original Grain-128'. Moreover, note that the required amount of keystream bits (belonging to a single initial state) would not be available for Draco due to the maximum packet length of 2^{32} bits.

10.3.3 Conditional Differentials, Cube Attacks

In [LM12], Lehmann and Meier study the security of Grain-128a against dynamic cube attacks and differential attacks. They come to the following conclusion:

> To analyse the security of the cipher, we study the monomial structure and use high order differential attacks on both the new and old versions. The comparison of symbolic expressions suggests that Grain-128a is immune against dynamic cube attacks. Additionally, we find that it is also immune against differential attacks as the best attack we could find results in a bias at round 189 out of 256.

The currently best key-recovery cube attack against round-reduced Grain-128a is presented in [TIHM17]. It is based on the division property and works for 183 initialization rounds.

Draco has 512 rounds in phase 2 of the state initialization, where the Grain-like mixing is performed as described in Section 10.1.2. On top of that, from the second half of phase 2 onwards (i.e, for all $t \geq 256$), the 32-bit prefix of the secret key is continuously involved in the state update of NFSR2.

Note that the volatile internal state of Draco (128 bits) is smaller than that of Grain-128a (256 bits), whereas the output function is much more dense. It depends on 59 variables as compared to 17 in Grain-128a. The output function of Draco also has more nonlinear monomials (15) than that of Grain-128a (5). Moreover, now both FSRs are nonlinear and the feedback function of NFSR1 is defined over more inputs (40 vs.19) and has more nonlinear monomials (15 vs.10) than that of Grain-128a's NFSR.

The combination of a smaller volatile state and more dense feedback and output functions causes a faster diffusion of differentials and of the monomial structure for Draco. Together with the doubled number of initialization rounds, this should make Draco at least as resistant against differential attacks and cube attacks as Grain-128a, which seems to be already sufficiently secure in that respect.

Earlier version of Draco. In 2021 Horn [Hor21] studied the resistance of an earlier version of Draco against cube attacks. Since then, the key-IV schedule was slightly modified to prevent the zero d-stream attacks mentioned in Subsection 10.2.1. In particular the work by Horn considers a key prefix of length 33 instead of 32 and an IV of length 95 instead of 96 with an additional 0-prefix. This will not significantly change the results obtained in the analysis against cube attacks.

Horn [Hor21] considered only 99 and 100 initialization rounds instead of the full 512 as 'the superpoly recovery for Draco frequently turned out to become computationally infeasible even for a very small number of initialization rounds.' Horn observed that in each clock cycle only one IV bit enters the internal state of NFSR2. He found practical distinguishers for 99 and 100 initialization rounds. Yet, he was not successful in attacking 101 initialization rounds. The author states that 'it was not possible to recover the

superpoly of a cube just a few rounds after the cube variable with the highest index is introduced.' Further, since DRACO uses 512 initialization rounds, Horn considers the margin more than sufficient to provide very high security against his cube attacks. Horn concludes that the 'extremely fast growing complexity of these superpolys of an even simplified version of DRACO, again supports our assumption that DRACO is extremely resistant against the considered attack.'

10.3.4 Slide Attacks, Related Key Attacks

In [Küç06], Küçük first pointed out a *sliding property* of the state initialization of Grain v1, which was later formally published by De Cannière, Küçük and Preneel in [DCKP08] as: 'For a fraction of $2^{-2 \cdot n}$ of pairs (K, IV), there exists a related pair (K^*, IV^*) which produces an identical but n-bit shifted key stream.' In the same paper, the authors describe how this property can be exploited to speed up exhaustive key search for Grain v1 (and also for Grain-128) by a factor of two.[3] In addition, they also suggest a related-key slide attack, for which they note: 'As is the case for all related key attacks, the simple attack just described is admittedly based on a rather strong supposition.' [DCKP08] As a reaction, the designers of Grain-128a changed the 22-bit constant $(1, \ldots, 1)$ that was used in the state initialization of Grain-128 to $(1, \ldots, 1, 0)$.

Avoiding the sliding property. To avoid the above sliding property, we set $B_0^0 := K_0 \oplus 1$ in phase 1 of the state initialization (cf. Section 10.1.2). As a result, for a key-IV pair (K, IV), a related key-IV pair (K^*, IV^*) in the sense of [DCKP08] would have to satisfy

$$K_0^* = K_1 \oplus 1. \tag{1}$$

Let d_t and d_t^* denote the key-IV-schedule bits computed on the basis of (K, IV) and (K^*, IV^*), respectively, as described previously in Section 10.1.1. For the sliding property from [DCKP08] to occur, $d_{t+1} = d_t^*$ would need to hold for $t \geq 0$. In particular, we get

$$IV_1 = d_1 = d_0^* = IV_0^* \tag{2}$$

and

$$K_1 \oplus IV_1 = d_{289} = d_{288}^* = K_0^* \oplus IV_0^*. \tag{3}$$

It is easy to see that equations (1), (2), and (3) cannot be satisfied simultaneously.

Note that, without inverting K_0 in phase 1 of the state initialization together with

[3] More precisely, this speed up refers to making the key candidate checks more efficient. The actual number of key candidate checks, however, is not reduced compared to exhaustive key search. Still, we consider such a sliding property undesirable as it might pave the way for other attacks and, hence, seek to avoid it for DRACO.

having different definitions of d_t for $t \leq 255$ and $t \geq 256$, Draco would in fact suffer from a variant of the sliding property, despite continuously employing the IV and the 32-bit key prefix for state update during keystream generation.

Related key & fault attacks. Let us also emphasize that, similar to the statement made by De Cannière, Küçük, and Preneel in [DCKP08] as cited earlier, we also consider the assumptions underlying related-key attacks to be quite strong. Specifically, we do not claim security in situations where a potential victim generates keystreams using secret related keys, while an attacker attempts to recover one or more of these keys. This scenario is often motivated by fault attacks, where an attacker manipulates the secret internal state and/or the unknown key bits through techniques like clock glitches, voltage spikes, optical or electromagnetic fault injection, etc. Additionally, it may arise due to the usage of a weak (session) key derivation method, which is an obvious security vulnerability that should be avoided regardless of the employed cipher. Furthermore, various established countermeasures are available at the hardware design level to protect against fault attacks (see, for example, [KSV13] for an overview). Therefore, we do not make security claims regarding other types of side-channel attacks either.

10.3.5 Weak Key-IV Pairs

In [ZW09], Zhang and Wang introduce the notion of *weak key-IV pairs* for the Grain family of stream ciphers. They show that Grain-128 has 2^{96} such pairs, which lead to an all-zero initial state of the LFSR, and use them to mount distinguishing attacks and initial state recovery attacks. In [ÅHJM11], the designers of Grain-128a point out: 'We note that the IV is normally assumed to be public, and that the probability of using a weak key-IV pair is 2^{-128}. Any attacker guessing this to happen and then launching a rather expensive attack, is much better off just guessing a key.'

All-zero state. In analogy to the definition of Zhang and Wang, weak key-IV pairs for Draco would lead to an all-zero initial state of NFSR1. Such pairs, however, are now completely unproblematic (and, hence, not *weak* anymore) as the 33-bit-wide NFSR1 has truly maximal period 2^{33}. In particular, unlike the LFSR of the Grain family, it cannot get stuck in the all-zero state.

Note that without adding the final term $\oplus\neg S_1^t \neg S_2^t \cdots \neg S_{30}^t \neg S_{31}^t \neg S_{32}^t$ to ACHTERBAHN's NFSR A_{12} for obtaining NFSR1 of Draco (cf. Section 10.1.1), there would have actually been about 2^{191} weak key-IV pairs out of 2^{224} total key-IV pairs, leading to a probability of 2^{-33} for using a weak pair. Thus, corresponding attacks might have posed a real threat to Draco, which is now avoided.

10.3.6 BDD-based Attacks

In the paper [Kra02], Krause introduced the concept of utilizing binary decision diagrams (BDDs) to attack LFSR-based stream ciphers such as A5/1 in the GSM standard or E_0 in Bluetooth. Later, Stegemann demonstrated in [Ste07] how this approach can be extended to NFSR-based stream ciphers like Trivium and Grain.

In contrast to TMD tradeoff attacks or correlation attacks, which potentially require a lot of known keystream, BDD attacks are *short-keystream attacks* in the sense that only the information-theoretic minimum of keystream bits (i.e., often only few more than n bits of keystream for a keystream generator of internal state length n) is required to recover the corresponding initial state.

Although we are currently not aware of any BDD attack faster than exhaustive key search against any member of the Grain family, the key design implication derived from Stegemann's BDD-related cryptanalytic findings for Grain-like stream ciphers is that the maximum number of what he terms *active monomials* in the feedback functions and output function should be maximized (refer to [Ste07] for more details). In Stegemann's setting, for Grain-128a, the maximum number of active monomials would be 0 for the LFSR, 3 for the NFSR, and 3 for the output function. In comparison, for DRACO, the maximum number of active monomials would be 19 for NFSR1, 6 for NFSR2, and at least 10 for the output function a. As a result, we anticipate that despite the smaller volatile internal state, DRACO will also resist BDD attacks.

10.3.7 Preventing Banik et al. and Esgin-Kara Attacks

Banik's attack [BCI+21] against Sprout is based on the LFSR-property that the constant zero internal state occurs in Sprout with a certain probability. This state generates a stream of zeros. As DRACO does not use any LFSR, this attack can not applied to DRACO.

The Banik, Baroti, Isobe attack against Plantlet [BBI19] exploits Plantlet's property that pairs of internal states which differ only in position 43 generate identical keystream blocks of length 41. This property is due to the comparatively large distance between certain taps of Plantlet's LFSR. To prevent this type of attack, the Atom stream cipher uses an additional second key filter which is driven by a 7-bit LFSR. As all pairs of neighbored taps of both of DRACO's NFSRs have a sufficiently small distance, the BBI-attack can not be applied to DRACO. This is the reason why we decided to use only a simple cyclic filter as key schedule for DRACO, and not an LFSR-driven one.

The Esgin-Kara attack against Sprout [EK15] exploits the fact that key bits from the key filter undergo multiplication by a term dependent on the volatile state. It can be demonstrated that these key bits can be zero during specific clock cycles. This indicates that certain blocks of the keystream solely rely on the volatile internal states, enabling nontrivial Time-Memory-Data Trade-Off (TMDTO) attacks. However, this attack methodology is not applicable to DRACO since the d-bits originating from the DRACO key-IV schedule are

linearly added to the state update function of NFSR1. Furthermore, the DRACO key-IV schedule ensures that each 128-bit key stream block is dependent on all key prefix and IV bits. Consequently, this design characteristic thwarts attacks similar to Esgin-Kara's attack described in Subsection 3.4 of [BCI+21].

10.4 Hardware Results

In this section, we present the hardware results for our new stream cipher DRACO and compare them to those of Atom [BCI+21] and Grain-128a [ÅHJM11][4], which, like DRACO, accepts 128-bit keys and 96-bit IVs. We also included hardware numbers for variants of DRACO where the key prefix is kept inside the hardware module, where the IV is kept inside the hardware module, and where both are kept inside the hardware module. These are indexed by [K], [I], and [KI], respectively.

Grain-128a. The reasons for focusing on Grain-128a are twofold. First, it is a natural choice for comparison due to the close structural relation between DRACO and the Grain family of stream ciphers as explained in Sections 10.1 and Section 10.2. Second, and more importantly, Grain v1 (the 80-bit version of Grain-128a) turned out to be the most hardware efficient member of the eSTREAM [ECR08] portfolio (see tables 1–4 and figures 1–3 in [GB08]) and, hence, the Grain family of stream ciphers can be considered as a benchmark for new designs. Also note that DRACO is the first small-state stream cipher offering full 128-bit security against key recovery *and* distinguishing attacks, which is why a comparison to, e.g., Plantlet or LIZARD would not be appropriate, here.

Atom. Second we chose to compare DRACO to Atom. Atom is a lightweight stream cipher that was recently published in ToSC [BCI+21]. Atom is a reasonable comparison as it also uses a 128-bit key and it further stores the secret key externally, i.e. it builds upon CKEY that we introduced earlier. We further implemented a version of Atom that stores its secret key internally in the hardware module in an additional register that is denoted in Table 10.1 as $\text{Atom}_{[K]}$. This is done to allow the comparison to variants of DRACO that store the key prefix, resp. IV, locally in the hardware module as described below. In particular, for $\text{DRACO}_{[KI]}$ and $\text{Atom}_{[K]}$ there are no dependencies to external resources, as is the case for Grain-128a.

ASICs. In line with publications like [Fel07, GB08], which evaluate candidates in the eSTREAM hardware category, we focus on application-specific integrated circuits (ASICs)

[4]Note that Grain-128a actually comes in two flavors: *authenticated encryption* and *encryption only*.
For reasons of fairness, we naturally consider the more lightweight encryption-only variant of Grain-128a in our comparison to DRACO. In fact, the authentication mechanism of Grain-128a is completely independent of the underlying keystream generator and could as well be used in connection with DRACO.

Design	**Area** [GE]	**Power** [µW]				
		100 KHz	1 MHz	10 MHz	100 MHz	1 GHz
Atom	2976	67.9	71.2	104.9	441.9	3811.7
$\text{Atom}_{[K]}$	3858	88.9	92.3	126.1	463.9	3842.3
Grain-128a	2795	67.3	71.6	115.3	551.4	4912.9
Draco	2142	48.8	51.6	79.2	355.6	3119.3
$\text{Draco}_{[K]}$	2368	54.2	57.0	84.7	369.1	3134.1
$\text{Draco}_{[I]}$	2805	64.6	67.7	95.1	372.3	3144.7
$\text{Draco}_{[KI]}$	3025	69.9	72.6	100.6	377.6	3150.0

Table 10.1: Hardware metrics for Draco and Grain-128a.

with standard CMOS libraries. ASICs are the prevalent hardware component in lightweight application scenarios, such as radio frequency identification (RFID) technology, and likewise important for highspeed cryptographic processing, such as bitcoin mining. The two main restrictions imposed on the design of cryptographic protocols for RFID tags are the circuit size and the power budget. The circuit size strongly influences the manufacturing costs of an RFID tag (see [AHM14] for details) and is commonly specified in gate equivalents (GE), where one GE corresponds to the area of a two-input drive-strength-one NAND gate. The power consumption is crucial as low-cost RFID tags are usually passively powered (i.e., via an electromagnetic field radiated by the reader). In ASIC-based high-speed processing, on the other hand, energy consumption is becoming the main cost factor (see, e.g., [DV18]).

Measuring power consumption. It is important to note that while the area requirement of cipher designs can be compared over different standard cell libraries by using the measure gate equivalents, '[p]ower cannot be scaled reliably between different processes and libraries' [GB08]. Consequently, it is crucial to use the same design flow for all implementations that are to be compared. The appendix of [HMKM22] provides a detailed specification of the tools and methodology employed for deriving the hardware evaluation results summarized in Table 10.1. After state initialization, all implementations produce one keystream bit per clock cycle, leading to identical throughput rates at identical clock speeds.

Large constant internal state. Remember that in contrast to Grain-128a, half of Draco's 256-bit internal state is actually held constant (consisting of the 32-bit key prefix and the 96-bit IV). This allows for maximizing Draco's resource efficiency by easily adapting the hardware implementation to each device's specific capabilities. For example,

if the secret key is burned into the device or stored in an EEPROM (a common RFID scenario [AHM14], assumed, e.g., by Plantlet) and the IV is constituted by the device's frame counter (as, e.g., in A5/1), then no storage cells for this data need to be allocated inside of the Draco hardware module, leading to the most lightweight variant labeled Draco in Table 10.1. If, on the other hand, the 32-bit key prefix and the 96-bit IV should both be available only at the beginning of state initialization (as generally assumed by Grain-128a), additional storage cells are required, leading to $\text{Draco}_{[\text{KI}]}$. The variants $\text{Draco}_{[\text{K}]}$ resp.$\text{Draco}_{[\text{I}]}$ represent the two intermediate scenarios that only the 32-bit key prefix resp.the 96-bit IV need to be held locally in the Draco hardware module.

10.4.1 Discussion of the Results

Grain-128a. The numbers presented in Table 10.1 show that the Draco stream cipher is likewise attractive for lightweight RFID and highspeed computation scenarios. For example, when making optimal use of an RFID tag's resources (i.e., burned/EEPROM key, transmission counter as IV), Draco requires 23 % less area (2142 vs.2795 GE) and 31 % less power (79.2 vs.115.3 µW) than Grain-128a at a clock frequency of 10 MHz. In the case of high speed computing, on the other hand, everything comes down to energy consumption. At a clock frequency of 1 GHz, all four implementation variants of Draco consume about 34 % less energy than Grain-128a for producing 10 kbit of keystream (including state initialization). This substantial advantage is achieved even if the 32-bit key prefix and the 96-bit IV have to be stored locally inside of the Draco hardware module (i.e., 32.7 nJ for $\text{Draco}_{[\text{KI}]}$ vs.50.4 nJ for Grain-128a; cf. Appendix C of [HMKM22]).

Atom. In direct comparison to Atom we can see that Draco needs 28 % less area (2142 vs.2976 GE) and 24 % less power (79.2 vs 104.9 µW) at a clock frequency of 10 MHz. Further comparing $\text{Atom}_{[\text{K}]}$ to $\text{Draco}_{[\text{KI}]}$ we see improvements of 21 % in area (3025 vs 3858 GE) and an improvement of 20 % in power consumption (100.6 vs 126.1 µW) at a clock frequency of 10 MHz.

Power consumption. The reason behind this is that already for moderate clock frequencies (here: between 10 MHz and 20 MHz) the dynamic power consumption (due to switching of values) dominates the static power consumption (due to leakage) of flip-flop storage cells. To the best of our knowledge, this effect has never been considered in stream cipher design before. Instead, the classical design paradigm (e.g., followed by Grain-128a, but also by Plantlet and Lizard) exclusively focused on the *number* of flip-flops, ignoring their actual *usage*. With $\text{Draco}_{[\text{KI}]}$, we demonstrate that even if $2n$-bit storage is required inside the cipher's hardware module to achieve n-bit security against TMDTO attacks, algorithmically keeping half of this state constant is much more efficient (cf. Tab. 10.1) and equally secure (see Section 10.3 and Section 9.5) as constantly updating the whole of it.

11 | Fixing the Key Schedule

The original version of the Draco stream cipher that was presented in this work and at FSE 2023 is vulnerable to an attack with a complexity of 107 bits [Ban22], which violates the security claim of 128 bits. The attack by Subhadeep Banik exploits a vulnerability in the key schedule. For some key-IV combinations, the key schedule periodically produces sequences that are all zero and thus several keystream bits are not affected by the non-volatile state. An attacker can exploit this to recover an internal state and the secret key. In particular, a combination of three properties enables the distinguishing attack by Subhadeep Banik.

1. A small key prefix.
2. The small period of the KIS-bit.
3. For some IVs the KIS-bit is zero for several cycles.

11.1 Banik's TMDTO Attacks

In this section, we will give an informal description of the two attacks by Banik [Ban22]. For a detailed description of the attacks, please refer to [Ban22].

First attack. Let ℓ_{IV} denote the IV length, let $\ell_{k^{\mathsf{pre}}}$ denote the key prefix length, let ℓ_p denote the packet length, and let p denote the period of the stream of key schedule bits d_t. The attack is based on the observation that for every key prefix k^{pre}, there exists an initial value $x_{k^{\mathsf{pre}}} \in \mathcal{IV}$ such that the stream of key schedule bits d_t has subsequences of length ℓ_{IV} where the key schedule bit is 0. These subsequences repeat every p bits, and the ℓ_p/p positions are known to the adversary. The all-zero subsequences of the key schedule bit d_t imply that at these known positions, the keystream depends only on the *volatile* internal state. The period of Draco's d-stream is small, i.e., $p \approx 2^{12}$, and $\ell_p/p \approx 2^{20}$. This allows for a birthday collision attack.

In the offline phase, T random volatile internal states y_i are generated. For each y_i, an output keystream z_i with length ℓ_{IV} is generated with the d-stream being 0, i.e., without considering the key schedule bits. Each (y_i, z_i) is stored in an efficiently searchable data structure $\mathcal{Z}$. In the online phase, for all $k^{\mathsf{pre}} \in \{0,1\}^{\ell_{k^{\mathsf{pre}}}}$, the packet for the corresponding

initial value $x_{k^{\text{pre}}} \in \mathcal{IV}$ is generated. If a collision in $\mathcal{Z}$ is found, the adversary verifies the key prefix guess by generating a few more keystream bits.

Second attack. The second attack by Banik [Ban22] uses the idea that for an all-zero IV, effectively only the key prefix influences the d-stream. The paper first describes the attack for an all-zero IV and then proceeds to describe a tradeoff between the online and offline phase, where the last E bits of the IV are non-zero.

The idea for the all-zero IV is that an attacker generates 128 keystream bits from 2^{96} randomly chosen internal states for every 2^{32} key prefix in the offline phase. For every key prefix, the keystream and corresponding internal state are stored in a table indexed by the keystream. This corresponds to an offline complexity of 2^{128}. In the online phase, the attacker will get one packet of size 2^{32} bits for the all-zero IV. For every 128-bit window in the obtained keystream (there are approximately 2^{32} windows), the attacker checks each of the 2^{32} tables for the corresponding keystream and then verifies a collision by computing a few more bits. This corresponds to an online complexity of 2^{64}.

For an all-zero IV, the total complexity is still that of exhaustive search, but it is remarkable that the online complexity is way lower than the offline complexity. The idea now is to consider IVs of the form $0^{96-E}||e$, where $e \in \{0,1\}^E$. The first $97-E$ keystream bits computed from such an IV do not depend on the IV bits and hence only on the key prefix. Also, as the non-volatile state has a size of 97 bits, these occurrences repeat every 97 cycles. Therefore, the idea in the offline phase is to compute a fewer amount of internal states, 2^{96-D} for every key prefix, assuming that only the first $96-E$ bits of the IV are zero. Accordingly, fewer keystream bits are computed. This will decrease the complexity in the offline phase.

In the online phase, the attacker will not only query the all-zero IV, but also all 2^E IVs of the form $0^{96-E}||e$. Furthermore, as fewer keystream bits per candidate state were generated, eliminating incorrect states is now more costly in the online phase.

By carefully balancing the parameters D and E [Ban22], achieves a total attack complexity of approximately $2^{107.1}$.

Possible fixes. We performed a detailed cost analysis of the two attacks by Banik in dependence of T, the key prefix length $\ell_{k^{\text{pre}}}$ and the IV length ℓ_{IV}. We found out that if the key prefix is slightly larger[1] than the initial value, i.e. $\ell_{k^{\text{pre}}} > \ell_{\text{IV}}$, the costs of Banik's attacks are worse than those of exhaustive key search. This motivated our changes in the design of Draco. We leave the key schedule for the inner d-stream as is but increase the length of the non-volatile key prefix from 32 bits to 128 bits, i.e., the full key length.

An alternative design change to prevent Banik's attack could be to modify the key schedule for the internal d-stream in such a way that the period of the d-stream p exceeds the

[1] A few bits difference are needed for Draco, as it takes six clock cycles for the first unknown d-bit to reach the first register cell of NFSR2 that is used by the output function a.

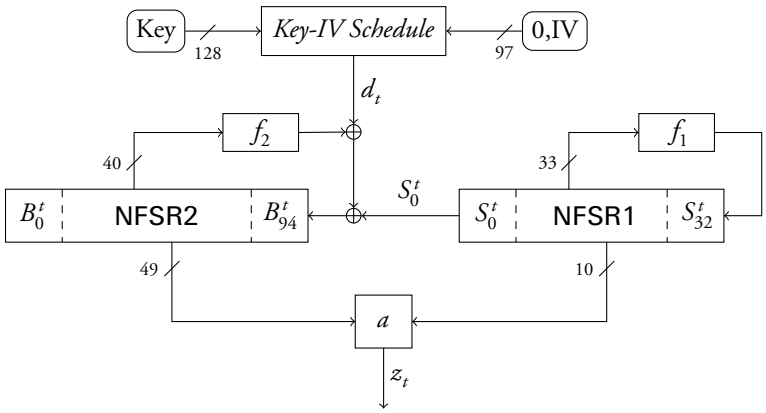

Figure 11.1: Draco^{QF} in keystream generation mode.

packet length ℓ_p. The advantage would be that we instantiate the minimal key prefix length suggested by the underlying proof of security for the generic CIVK construction. One idea to implement this could be to use some bits of the NFSRs to encode the addresses of the key prefix and IV bits to be included in the internal d-stream. A detailed security analysis of this more complex alternative approach is a matter of future research.

11.2 Design Specification of the Quick Fix

In this section we will only present the changes made to Draco to obtain the Quick Fix, which is not vulnerable to Banik's attack [Ban22].

The Quick Fix increases the key prefix length from 32 bits to 128 bits. This will change Subsection 10.1.1, as well as Figure 10.1 and Figure 10.2. The update of Subsection 10.1.1 can be found in Subsection 11.2.1 and Subsection 11.2.1 below. Figure 11.1 and Figure 11.2 below contain the updates to Figure 10.1 and Figure 10.2.

11.2.1 Components

Let $K = (K_0, \ldots, K_{127})$ denote the 128-bit secret key and $IV = (IV_0, \ldots, IV_{95})$ the 96-bit public IV. The 128-bit volatile internal state of Draco^{QF} is distributed over two NFSRs, NFSR1 and NFSR2, whose contents at time $t = 0, 1, \ldots$ we denote by $(S_0^t, \ldots, S_{32}^t)$ and $(B_0^t, \ldots, B_{94}^t)$, respectively (cf. Figure 11.1). As NFSR1 and NFSR2 are Fibonacci-type, for $t \in \mathbb{N}$ it holds that $S_i^{t+1} := S_{i+1}^t$, $i = 0, \ldots, 31$, and $B_j^{t+1} := B_{j+1}^t$, $j = 0, \ldots, 93$.

Computation of the KIS-bit. Besides the 128-bit volatile internal state, Draco^{QF} additionally employs a 224-bit non-volatile internal state, which is formed by the 96-bit public IV and the 128-bit secret key.

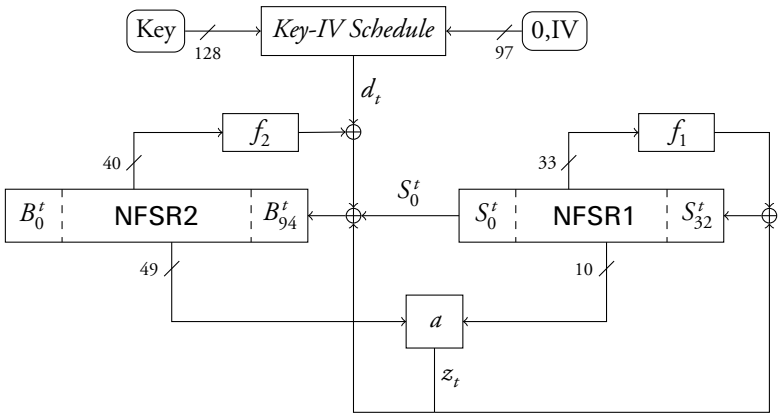

Figure 11.2: Draco^{QF} in phase 2 of the state initialization.

Specification (*Key schedule*)

Based on the 96-bit public IV, the 128-bit secret key, and the public 1-bit constant 0, in clock cycle t the *key-IV-schedule bit* (KIS bit) d_t is computed as

$$d_t := \begin{cases} x_{t \bmod 97}, & \text{for } 0 \le t \le 255, \\ K_{t \bmod 128} \oplus x_{t \bmod 97}, & \text{for } t \ge 256, \end{cases}$$

where $x_0 := 0$ and $x_i := IV_{i-1}$ for $i = 1, \ldots, 96$. The KIS-bit d_t is fed to NFSR2 as depicted in Figure 11.1 and Figure 11.2, respectively.

11.3 Hardware Metrics for the Quick Fix

In this section, we present the updated hardware metrics. The updated power and area values for Draco^{QF} are presented in Table 11.1. The indexed variants of Draco denote whether the key (prefix) ([K]), the IV ([I]), or both ([KI]) are kept in the hardware module. The main variant of Draco with an external key (prefix) and IV is denoted without an index.

Atom & Grain. The updated variant of Draco requires about 9% more area, measured in gate equivalents (GE). The increase in power consumption lies between 6.6% (100 KHz) and 3.8% (1 GHz). Compared to Grain-128a, the reduction in area is 16.5%, and the reduction in power consumption lies between 22.7% (100 KHz) and 34.1% (1 GHz). Compared to Atom, the reduction in area is 21.6%, and the reduction in power consumption lies between 23.4% (100 KHz) and 15.0% (1 GHz).

Internal non-volatile state. Obviously, the variants with an internal key (prefix), Draco$_{[K]}$ and Draco$_{[KI]}$, are affected most by increasing the length of the non-volatile

Design	**Area** [GE]	**Power** [µW]				
		100 KHz	1 MHz	10 MHz	100 MHz	1 GHz
Atom	2976	67.9	71.2	104.9	441.9	3811.7
$\text{Atom}_{[K]}$	3858	88.9	92.3	126.1	463.9	3842.3
Grain-128a	2795	67.3	71.6	115.3	551.4	4912.9
Draco	2142	48.8	51.6	79.2	355.6	3119.3
Draco^{QF}	2334	52.0	54.8	83.5	370.3	3238.4
$\text{Draco}_{[K]}$	2368	54.2	57.0	84.7	369.1	3134.1
$\text{Draco}^{QF}_{[K]}$	3215	73.0	75.9	104.6	392.3	3269.0
$\text{Draco}_{[I]}$	2805	64.6	67.7	95.1	372.3	3144.7
$\text{Draco}^{QF}_{[I]}$	2997	67.8	70.6	99.4	387.0	3263.8
$\text{Draco}_{[KI]}$	3025	69.9	72.6	100.6	377.6	3150.0
$\text{Draco}^{QF}_{[KI]}$	3872	88.7	91.5	120.3	408.0	3284.8

Table 11.1: Hardware metrics for Draco, Draco^{QF}, Atom and Grain-128a.

key prefix from 32 bits (Draco) to 128 bits (Draco^{QF}). $\text{Draco}^{QF}_{[K]}$ requires 35.8% more area than the original variant, and $\text{Draco}^{QF}_{[KI]}$ requires 28% more area than the original variant. With regard to power, the increase of $\text{Draco}^{QF}_{[K]}$ compared to the original variant is between 34.7% (100 KHz) and 4.3% (1 GHz), while the increase of $\text{Draco}^{QF}_{[KI]}$ is between 26.9% (100 KHz) and 4.3% (1 GHz).

11.4 Keeping the State Small

The quick fix is not a desirable solution, as it effectively destroys the idea behind the Draco stream cipher. Instead of increasing the size of the key prefix, one may try to get rid of the other two properties in the key schedule.

Generating an address. Increasing the period of the KIS-bit could be done by generating the address of the 32-bit key prefix bit chosen in the current clock cycle based on five bits taken from one of the NFSRs. This will equal the KIS-bit period to that of the NFSR. A problem with this approach is that it may happen that some key prefix bits do not enter the KIS-bitstream for several hundred clock cycles, as the contents of the NFSR are pseudorandom.

Cyclically chosen bits. To fix this problem, another two key prefix bits are added, which are cyclically chosen from the first 15 and the last 17 bits of the key prefix. Two bits are chosen so that at least one key prefix bit enters the KIS-bitstream in every clock cycle, as

Prefix bits	Offset																
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
First 15	1	0	1	0	1	0	1	0	1	1	0	0	1	0	0		
Last 17	0	1	0	1	1	0	0	0	0	0	0	1	1	0	0	1	1

Table 11.2: Seven consecutive erasures for the cyclically chosen bits.

it may happen that the key prefix bit generated from the address and one cyclically chosen key prefix bit may cancel each other out. The cyclically chosen bits will never cancel each other out due to being chosen from disjunct index sets. 15 and 17 were chosen as they are coprime and thus increase the period of the cyclically chosen bits.

Calculating the address. The address for the work in progress version of Draco is chosen based on NFSR1. NFSR1 has a size of 33 bits and a period of 2^{33} bits. This allows to run analyses and experimentally verify properties of the key prefix addresses chosen. We analyzed the following properties for various tap selections from which we pull the five address bits:

1. For all key bits $x \in [32]$: What is the largest number of clock cycles c_x missing key bit x? How large is $\max_x(c_x)$?
2. How many key prefix bits are missing in the worst window of size 128 (volatile state length)?
3. For all cyclically chosen key prefix bits $y \in [32]$: What is the largest window where y is cancelled[2] by the address, i.e., how often is bit y consecutively cancelled every 15, resp. 17, clock cycles?
4. How often do we hit the limit of 7 for 3.?

After several thousand test runs for randomly chosen tap selections, we identified an upper limit of 800 for 1. and an upper limit of 10 for 2. as these proved to be among the best. No test run had a value below 729 for 1. and no test run had a value smaller than 10 for 2. The limit of 7 for 3. and 4. is due to the fact that $7 \cdot 15 + 14 = 119$ and thus the KIS-bitstream is missing a bit from the first 15 consecutively chosen key prefix bits for at most 119 cycles and thus enters the KIS-stream at 120. As $7 \cdot 17 + 16 = 135$, the KIS-bitstream is missing a bit from the last 17 consecutively chosen key prefix bits for at most 135 cycles and thus enters the KIS-stream at 136. For the last 17 this is unfortunately larger than the volatile state size of 128, but after days of exhaustive search, no better values were found.

[2] If the cyclically chosen bit is the same as that chosen by the generated address, they cancel each other out due to being combined by the exclusive-or operation.

We eventually decided for the tap selection $\{24, 4, 22, 14, 28\}$ for our preliminary tests. Its values are 777 for 1. and 10 for 2. Seven consecutive erasures of the cyclically chosen bits appear at the respective offsets as given in Table 11.2.

Updated key schedule. The updated key schedule can be found below. Note that for area efficiency in hardware, the updated key schedule is identical during mixing but changes for keystream generation. Here, addr(t) describes the index of the key prefix bit that is chosen via the address generated from NFSR1.

Specification (*Updated key schedule*)

In clock cycle t the *key-IV-schedule bit* (KIS bit) d_t is computed as

$$d_t := \begin{cases} x_{t \bmod 97}, & \text{for } 0 \leq t \leq 255, \\ K_{t \bmod 32} \oplus x_{t \bmod 97}, & \text{for } 256 \leq t \leq 511, \\ K_{t \bmod 15} \oplus K_{(t \bmod 17)+15} \oplus K_{\text{addr(t)}} \oplus x_{t \bmod 97}, & \text{for } t \geq 512, \end{cases}$$

where $x_0 := 0$, $x_i := IV_{i-1}$ for $i = 1, \ldots, 96$, and

$$\text{addr}(t) := (S_{28}^t, S_{14}^t, S_{22}^t, S_4^t, S_{24}^t) \in \{0, \ldots, 31\}.$$

11.5 Hardware Metrics for the Work in Progress

The updated hardware metrics can be found in Table 11.3. In particular, one finds that the area increased by only 4.9% to 7.0% depending on the implementation variant, as opposed to the up to 35.8% increase of the quick fix. The power consumption also only increased modestly in the range of 2.2% to up to 5.9%, depending on the implementation variant and the clock speed. The increase in power consumption with the quick fix was up to 34.7%.

Grain & Atom. For the work-in-progress version, the area is still 18% smaller than Grain-128a and 23% smaller than Atom. The power consumption improvements over Grain-128a are in the range of 23.2% to 33.9%. The power consumption improvements over Atom are in the range of 14.8% to 23.9%.

Internal key prefix and IV. $\text{Draco}_{[\text{KI}]}^{\text{WIP}}$ uses 13.6% more area than Grain-128a and 6.7% more area than Atom. At low clock frequencies, $\text{Draco}_{[\text{KI}]}^{\text{WIP}}$ has increased power requirements over Grain and Atom. For Grain-128a, the increase is 8.0% at 100 kHz and 5.6% at 1 MHz. For Atom, the increase is 7.1% at 100 kHz and 6.2% at 1 MHz. From 10 MHz upwards, $\text{Draco}_{[\text{KI}]}^{\text{WIP}}$ sees decreased power requirements over Grain and Atom. For

Design	Area [GE]	Power [µW]				
		100 KHz	1 MHz	10 MHz	100 MHz	1 GHz
Atom	2976	67.9	71.2	104.9	441.9	3811.7
$\text{Atom}_{[K]}$	3858	88.9	92.3	126.1	463.9	3842.3
Grain-128a	2795	67.3	71.6	115.3	551.4	4912.9
Draco	2142	48.8	51.6	79.2	355.6	3119.3
$\text{Draco}^{\text{WIP}}$	2292	51.7	54.5	83.3	370.8	3245.9
$\text{Draco}_{[K]}$	2368	54.2	57.0	84.7	369.1	3134.1
$\text{Draco}^{\text{WIP}}_{[K]}$	2517	57.0	59.9	88.8	377.1	3260.8
$\text{Draco}_{[I]}$	2805	64.6	67.7	95.1	372.3	3144.7
$\text{Draco}^{\text{WIP}}_{[I]}$	2955	67.5	70.4	99.2	387.6	3271.3
$\text{Draco}_{[KI]}$	3025	69.9	72.6	100.6	377.6	3150.0
$\text{Draco}^{\text{WIP}}_{[KI]}$	3174	72.7	75.6	104.4	392.8	3276.6

Table 11.3: Hardware metrics for Draco, $\text{Draco}^{\text{WIP}}$, Atom and Grain-128a

Grain-128a, the decrease is 9.5% at 10 MHz, 28.8% at 100 MHz, and 33.3% at 1 GHz. For Atom, the decrease is 0.5% at 10 MHz, 11.1% at 100 MHz, and 14.0% at 1 GHz.

In direct comparison to $\text{Atom}_{[K]}$, we see a decrease in area and power requirements for $\text{Draco}^{\text{WIP}}_{[KI]}$. Area is reduced by 17.7%, and the power consumption is decreased by 14.7% at 1 GHz, to 18.2% at 100 kHz.

Outlook. The numbers for the work in progress version look promising. The work in progress version is only marginally worse than the original version of Draco. However, it is important to note that this version is preliminary and needs further checking for security vulnerabilities. This is a topic for further research.

11.6 Conclusion

In this part, we presented the new generic stream cipher construction CIVK and a new stream cipher proposal called Draco that instantiates CIVK. CIVK provably provides *full* volatile state length security against distinguishing attacks, providing a solid theoretical foundation to design stream ciphers upon.

Draco uses a 128-bit key, which is loaded to the volatile state cells of its feedback shift registers during initialization. A 32-bit prefix of this key, together with a 96-bit initial value, is continuously employed as part of the state update during keystream generation. If the key prefix and the initial value are stored 'externally' (e.g., inside an EEPROM), this design requires 18% less area and 27.8% less power than Grain-128a at 10 MHz.

For high-performance environments, we also considered an implementation variant called DRACO$_{[KI]}$ with the key prefix and the initial value stored inside the cipher hardware module, while still only half of the total internal state is updated during state updates. When clocked at 1 GHz, this variant consumes about 33.3% less energy than Grain-128a, still providing 128 bits of security and thus challenging the current paradigm of stream ciphers to always incorporate all internal state bits during state updates.

As future work, we suggest evaluating the performance of DRACO on other hardware platforms like FPGAs or microcontrollers. Moreover, it might be interesting to investigate whether, under the current security guarantees, even more lightweight variants of DRACO are possible, for example by choosing a lighter output function.

Bibliography

[AB12] Jean-Philippe Aumasson and Daniel J. Bernstein. Siphash: A fast short-input PRF. In Steven D. Galbraith and Mridul Nandi, editors, *Progress in Cryptology - INDOCRYPT 2012, 13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012. Proceedings*, volume 7668 of *Lecture Notes in Computer Science*, pages 489–508. Springer, 2012.

[ABD+13] Elena Andreeva, Andrey Bogdanov, Yevgeniy Dodis, Bart Mennink, and John P Steinberger. On the indifferentiability of key-alternating ciphers. In *CRYPTO 2013*, pages 531–550. Springer, 2013.

[ADMA15] Elena Andreeva, Joan Daemen, Bart Mennink, and Gilles Van Assche. Security of Keyed Sponge Constructions Using a Modular Proof Approach. In Gregor Leander, editor, *FSE*, volume 9054 of *LNCS*, pages 364–384. Springer, 2015.

[AGH18] Vahid Amin Ghafari and Honggang Hu. Fruit-80: a secure ultra-lightweight stream cipher for constrained environments. *Entropy*, 20(3):180, 2018.

[ÅHJM11] Martin Ågren, Martin Hell, Thomas Johansson, and Willi Meier. Grain-128a: A New Version of Grain-128 with Optional Authentication. *IJWMC*, 5(1):48–59, December 2011.

[AHM14] Frederik Armknecht, Matthias Hamann, and Vasily Mikhalev. Lightweight Authentication Protocols on Ultra-Constrained RFIDs - Myths and Facts. In *RFIDSec 2014*, pages 1–18. Springer International Publishing, Cham, 2014.

[AM15] Frederik Armknecht and Vasily Mikhalev. On lightweight stream ciphers with shorter internal states. In *FSE 2015*, pages 451–470. Springer, 2015.

[Aum17] Jean-Philippe Aumasson. *Serious cryptography: a practical introduction to modern encryption*. No Starch Press, 2017.

[Bab95] Steve H. Babbage. Improved "exhaustive search" attacks on stream ciphers. In *European Convertion on Security and Detection 1995*, pages 161–166, May 1995.

[Ban22] Subhadeep Banik. Cryptanalysis of draco. *IACR Trans. Symmetric Cryptol.*, 2022(4):92–104, 2022.

[BB12] D. Bider and M. Baushke. SHA-2 Data Integrity Verification for the Secure Shell (SSH) Transport Layer Protocol. RFC 6668 (Proposed Standard), July 2012.

[BBI19] Subhadeep Banik, Khashayar Barooti, and Takanori Isobe. Cryptanalysis of plantlet. *IACR Transactions on Symmetric Cryptology*, 2019, Issue 3:103–120, 2019.

[BCI+21] Subhadeep Banik, Andrea Caforio, Takanori Isobe, Fukang Liu, Willi Meier, Kosei Sakamoto, and Santanu Sarkar. Atom: A Stream Cipher with Double Key Filter. *IACR Transactions on Symmetric Cryptology*, pages 5–36, 2021.

[BCK96a] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 1996.

[BCK96b] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Message authentication using hash functions: The hmac construction. *RSA Laboratories' CryptoBytes*, 2(1):12–15, 1996.

[BD06] Steve Babbage and Matthew Dodd. The stream cipher MICKEY 2.0. eSTREAM: the ECRYPT Stream Cipher Project, 2006. `http://www.ecrypt.eu.org/stream/p3ciphers/mickey/mickey_p3.pdf`.

[Ber68] Elwyn R. Berlekamp. *Algebraic coding theory*. McGraw-Hill series in systems science. McGraw-Hill, 1968.

[Ber05a] Daniel J. Bernstein. Salsa20/12. eSTREAM: the ECRYPT Stream Cipher Project, 2005. `https://www.ecrypt.eu.org/stream/e2-salsa20.html`.

[Ber05b] Daniel J. Bernstein. Stronger Security Bounds for Wegman-Carter-Shoup Authenticators. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *LNCS*, pages 164–180. Springer, 2005.

[Ber05c] Daniel J. Bernstein. The Poly1305-AES Message-Authentication Code. In Henri Gilbert and Helena Handschuh, editors, *FSE*, volume 3557 of *LNCS*, pages 32–49. Springer, 2005.

[Ber07] Daniel J Bernstein. Polynomial evaluation and message authentication, 2007. `https://cr.yp.to/antiforgery/pema-20071022.pdf`.

[Ber08] Daniel J. Bernstein. ChaCha, a variant of Salsa20. `https://cr.yp.to/chacha/chacha-20080128.pdf`, 2008.

[BGJ09] Côme Berbain, Henri Gilbert, and Antoine Joux. Algebraic and Correlation Attacks against Linearly Filtered Non Linear Feedback Shift Registers. In *SAC 2008*, pages 184–198. Springer, Berlin, Heidelberg, 2009.

[BGM06] Côme Berbain, Henri Gilbert, and Alexander Maximov. Cryptanalysis of Grain. In *FSE 2006*, pages 15–29. Springer, Berlin, Heidelberg, 2006.

[BHK+99] John Black, Shai Halevi, Hugo Krawczyk, Ted Krovetz, and Phillip Rogaway. UMAC: fast and secure message authentication. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 216–233. Springer, 1999.

[BKL+12] Andrey Bogdanov, Lars R Knudsen, Gregor Leander, François-Xavier Standaert, John Steinberger, and Elmar Tischhauser. Key-alternating ciphers in a provable setting: Encryption using a small number of public permutations. In *EUROCRYPT 2012*, pages 45–62. Springer, 2012.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993*, pages 62–73. ACM, 1993.

[Bri12] David Brink. A (probably) exact solution to the Birthday Problem. *The Ramanujan Journal*, 28(2):223–238, 2012.

[BS00] Alex Biryukov and Adi Shamir. Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, pages 1–13. Springer, Berlin, Heidelberg, 2000.

[BSW01] Alex Biryukov, Adi Shamir, and David Wagner. Real Time Cryptanalysis of A5/1 on a PC. In *FSE 2000*, pages 1–18. Springer, Berlin, Heidelberg, 2001.

[CCH10] Claude Carlet, Yves Crama, and Peter L. Hammer. Boolean functions for cryptography and error-correcting codes. In Yves Crama and Peter L. Hammer, editors, *Boolean Models and Methods in Mathematics, Computer Science, and Engineering*, pages 257–397. Cambridge University Press, 2010.

[CDN22] Yu Long Chen, Avijit Dutta, and Mridul Nandi. Multi-user BBB security of public permutations based MAC. *Cryptogr. Commun.*, 14(5):1145–1177, 2022.

[CDN+23] Benoît Cogliati, Avijit Dutta, Mridul Nandi, Jacques Patarin, and Abishanka Saha. Proof of mirror theory for a wide range of $\xi _{\max }$. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part IV*, volume 14007 of *Lecture Notes in Computer Science*, pages 470–501. Springer, 2023.

[CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *Journal of the ACM (JACM)*, 51(4):557–594, 2004.

[CGS17] Debrup Chakraborty, Sebati Ghosh, and Palash Sarkar. A Fast Single-Key Two-Level Universal Hash Function. *IACR Trans. Symmetric Cryptol.*, 2017(1):106–128, 2017.

[CLL+14] Shan Chen, Rodolphe Lampe, Jooyoung Lee, Yannick Seurin, and John P. Steinberger. Minimizing the two-round even-mansour cipher. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 39–56. Springer, 2014.

[CLLL20] Wonseok Choi, ByeongHak Lee, Yeongmin Lee, and Jooyoung Lee. Improved security analysis for nonce-based enhanced hash-then-mask macs. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2020 - 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7-11, 2020, Proceedings, Part I*, volume 12491 of *Lecture Notes in Computer Science*, pages 697–723. Springer, 2020.

[CLS17] Benoît Cogliati, Jooyoung Lee, and Yannick Seurin. New Constructions of MACs from (Tweakable) Block Ciphers. *IACR Trans. Symmetric Cryptol.*, 2017(2):27–58, 2017.

[CP05] Christophe De Cannière and Bart Preneel. Trivium – Specifications. eSTREAM: the ECRYPT Stream Cipher Project, 2005. `http://www.ecrypt.eu.org/stream/p3ciphers/trivium/trivium_p3.pdf`.

[CP20] Benoît Cogliati and Jacques Patarin. Mirror theory: A simple proof of the pi+pj theorem with xi_max=2. *IACR Cryptol. ePrint Arch.*, page 734, 2020.

[CS14] Shan Chen and John P. Steinberger. Tight Security Bounds for Key-Alternating Ciphers. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT*, volume 8441 of *LNCS*, pages 327–350. Springer, 2014.

[CS16] Benoît Cogliati and Yannick Seurin. EWCDM: an efficient, beyond-birthday secure, nonce-misuse resistant MAC. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 121–149. Springer, 2016.

[CS18] Benoît Cogliati and Yannick Seurin. Analysis of the single-permutation encrypted Davies–Meyer construction. *Designs, Codes and Cryptography*, Mar 2018.

[CW79] Larry Carter and Mark N. Wegman. Universal Classes of Hash Functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979.

[DCKP08] Christophe De Cannière, Özgül Küçük, and Bart Preneel. Analysis of Grain's Initialization Algorithm. In Serge Vaudenay, editor, *AFRICACRYPT 2008*, pages 276–289. Springer, Berlin, Heidelberg, 2008.

[DDD21] Nilanjan Datta, Avijit Dutta, and Kushankur Dutta. Improved security bound of (E/D)WCDM. *IACR Trans. Symmetric Cryptol.*, 2021(4):138–176, 2021.

[DDN+15] Nilanjan Datta, Avijit Dutta, Mridul Nandi, Goutam Paul, and Liting Zhang. Building Single-Key Beyond Birthday Bound Message Authentication Code. Cryptology ePrint Archive, Report 2015/958, 2015. Version: 20160211:123920.

[DDNP18] Nilanjan Datta, Avijit Dutta, Mridul Nandi, and Goutam Paul. Double-block Hash-then-Sum: A Paradigm for Constructing BBB Secure PRF. *IACR Transactions on Symmetric Cryptology*, 2018(3):36–92, Sep. 2018. Full updated version at `https://eprint.iacr.org/2018/804`.

[DDNT23] Nilanjan Datta, Avijit Dutta, Mridul Nandi, and Suprita Talnikar. Tight multi-user security bound of dbhts. *IACR Trans. Symmetric Cryptol.*, 2023(1):192–223, 2023.

[DDNY18] Nilanjan Datta, Avijit Dutta, Mridul Nandi, and Kan Yasuda. Encrypt or Decrypt? To Make a Single-Key Beyond Birthday Secure Nonce-Based MAC. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO I*, volume 10991 of *LNCS*, pages 631–661. Springer, 2018.

[DH15] Elena Dubrova and Martin Hell. Espresso: A stream cipher for 5G wireless communication systems. *Cryptography and Communications*, pages 1–17, 2015.

[DHT17] Wei Dai, Viet Tung Hoang, and Stefano Tessaro. Information-theoretic indistinguishability via the chi-squared method. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 497–523. Springer, 2017.

[DJN16] Avijit Dutta, Ashwin Jha, and Mridul Nandi. Exact Security Analysis of Hash-then-Mask Type Probabilistic MAC Constructions. *IACR Cryptology ePrint Archive*, 2016:983, 2016.

[DJN17] Avijit Dutta, Ashwin Jha, and Mridul Nandi. Tight Security Analysis of EHtM MAC. *IACR Transactions of Symmetric Cryptology*, 2017(3):130–150, 2017.

[DKS12] Orr Dunkelman, Nathan Keller, and Adi Shamir. Minimalism in cryptography: The Even-Mansour scheme revisited. In *EUROCRYPT 2012*, pages 336–354. Springer, 2012.

[DN20] Avijit Dutta and Mridul Nandi. BBB secure nonce based MAC using public permutations. In Abderrahmane Nitaj and Amr M. Youssef, editors, *Progress in Cryptology - AFRICACRYPT 2020 - 12th International Conference on Cryptology in Africa, Cairo, Egypt, July 20-22, 2020, Proceedings*, volume 12174 of *Lecture Notes in Computer Science*, pages 172–191. Springer, 2020.

[DNS22] Avijit Dutta, Mridul Nandi, and Abishanka Saha. Proof of mirror theory for $\xi_{\max} = 2$. *IEEE Trans. Inf. Theory*, 68(9):6218–6232, 2022.

[DNT19] Avijit Dutta, Mridul Nandi, and Suprita Talnikar. Beyond Birthday Bound Secure MAC in Faulty Nonce Model. *IACR Cryptology ePrint Archive*, 2019:127, 2019. To appear in EUROCRYPT 2019.

[DR08] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), August 2008. Updated by RFCs 5746, 5878, 6176, 7465, 7507, 7568, 7627, 7685, 7905, 7919.

[DS11] Itai Dinur and Adi Shamir. Breaking Grain-128 with Dynamic Cube Attacks. In *FSE 2011*, pages 167–187. Springer, Berlin, Heidelberg, 2011.

[DV18] Alex De Vries. Bitcoin's growing energy problem. *Joule*, 2(5):801–805, 2018.

[ECR08] ECRYPT – European Network of Excellence for Cryptology. eSTREAM: the ECRYPT stream cipher project, 2008. `http://www.ecrypt.eu.org/stream/`.

[EK15] Muhammed F Esgin and Orhun Kara. Practical cryptanalysis of full sprout with tmd tradeoff attacks. In *International Conference on Selected Areas in Cryptography*, pages 67–85. Springer, 2015.

[EM97] Shimon Even and Yishay Mansour. A construction of a cipher from a single pseudorandom permutation. *Journal of cryptology*, 10(3):151–161, 1997.

[Fel07] Martin Feldhofer. Comparison of Low-Power Implementations of Trivium and Grain. eSTREAM, ECRYPT Stream Cipher Project, Report 2007/027, 2007. `http://www.ecrypt.eu.org/stream/papersdir/2007/027.pdf`.

[GB08] Tim Good and Mohammed Benaissa. Hardware performance of eStream phase-III stream cipher candidates. eSTREAM: the ECRYPT Stream Cipher Project, 2008. `http://www.ecrypt.eu.org/stream/docs/hardware.pdf`.

[GGK06] Berndt Gammel, Rainer Göttfert, and Oliver Kniffler. Achterbahn-128/80. eSTREAM: the ECRYPT Stream Cipher Project, 2006. `http://www.ecrypt.eu.org/stream/p2ciphers/achterbahn/achterbahn_p2.pdf`.

[GK14] Shay Gueron and Michael E. Kounavis. Intel Carry-Less Multiplication Instruction and its Usage for Computing the GCM Mode - Rev 2.02. Intel White Paper. Technical report, Intel corporation, April 20 2014.

[GL15] Shay Gueron and Yehuda Lindell. GCM-SIV: Full Nonce Misuse-Resistant Authenticated Encryption at Under One Cycle per Byte. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS*, pages 109–119. ACM, 2015.

[Gol96] Jovan Dj. Golić. On the security of nonlinear filter generators. In Dieter Gollmann, editor, *FSE 1996*, pages 173–188. Springer, Berlin, Heidelberg, 1996.

[Gol97] Jovan Dj Golić. Cryptanalysis of alleged a5 stream cipher. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 239–255. Springer, 1997.

[Hel80] Martin Hellman. A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory*, 26(4):401–406, Jul 1980.

[HJM06] Martin Hell, Thomas Johansson, and Willi Meier. Grain - A Stream Cipher for Constrained Environments. eSTREAM: the ECRYPT Stream Cipher Project, 2006. `http://www.ecrypt.eu.org/stream/p3ciphers/grain/Grain_p3.pdf`.

[HJMM08] Martin Hell, Thomas Johansson, Alexander Maximov, and Willi Meier. The Grain Family of Stream Ciphers. In *New Stream Cipher Designs: The eSTREAM Finalists*, pages 179–190. Springer, Berlin, Heidelberg, 2008.

[HK97] Shai Halevi and Hugo Krawczyk. MMH: software message authentication in the gbit/second rates. In Eli Biham, editor, *Fast Software Encryption, 4th International Workshop, FSE '97, Haifa, Israel, January 20-22, 1997, Proceedings*, volume 1267 of *Lecture Notes in Computer Science*, pages 172–189. Springer, 1997.

[HKM17a] Matthias Hamann, Matthias Krause, and Willi Meier. LIZARD – A Lightweight Stream Cipher for Power-constrained Devices. *IACR ToSC*, 2017(1):45–79, 2017.

[HKM17b] Matthias Hamann, Matthias Krause, and Willi Meier. A note on stream ciphers that continuously use the iv. *IACR Cryptology ePrint Archive*, 2017:1172, 2017.

[HKM19] Matthias Hamann, Matthias Krause, and Alexander Moch. Tight security bounds for generic stream cipher constructions. In *SAC 2019*, pages 335–364. Springer, 2019.

[HKMZ18] Matthias Hamann, Matthias Krause, Willi Meier, and Bin Zhang. Design and analysis of small-state grain-like stream ciphers. *Cryptography and Communications*, 10(5):803–834, 2018.

[HL10] E. Hammer-Lahav. The OAuth 1.0 Protocol. RFC 5849 (Proposed Standard), April 2010.

[HMKM22] Matthias Hamann, Alexander Moch, Matthias Krause, and Vasily Mikhalev. The draco stream cipher: A power-efficient small-state stream cipher with full provable security against tmdto attacks. *IACR Transactions on Symmetric Cryptology*, 2022, Issue 2:1–42, 2022.

[Hor21] Tobias Horn. On Cube Attacks on Stream Ciphers. Master's thesis, Universität Mannheim, 2021. `https://www.wim.uni-mannheim.de/media/Lehrstuehle/wim/ths/files/tohorn_masters.pdf`.

[HP08] Helena Handschuh and Bart Preneel. Key-recovery attacks on universal hash function based mac algorithms. In *Annual International Cryptology Conference*, pages 144–161. Springer, 2008.

[HS05] Jin Hong and Palash Sarkar. New applications of time memory data tradeoffs. In Bimal Roy, editor, *ASIACRYPT 2005*, pages 353–372, Berlin, Heidelberg, 2005. Springer.

[IM16] Tetsu Iwata and Kazuhiko Minematsu. Stronger Security Variants of GCM-SIV. *IACR Transactions of Symmetric Cryptology*, 2016(1):134–157, 2016.

[Ins21] Institute of Electrical and Electronics Engineers. IEEE Standard for Information Technology – Telecommunications and Information Exchange between Systems – Local and Metropolitan Area Networks – Specific Requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016)*, pages 1–4379, 2021.

[Jou06] Antoine Joux. Authentication failures in nist version of gcm. *NIST Comment*, page 3, 2006.

[KA98a] S. Kent and R. Atkinson. IP Authentication Header. RFC 2402 (Proposed Standard), November 1998.

[KA98b] S. Kent and R. Atkinson. IP Encapsulating Security Payload (ESP). RFC 2406 (Proposed Standard), November 1998.

[KL20] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC press, 2020.

[Kle13] Andreas Klein. *Stream ciphers*, volume 12. Springer, 2013.

[KLL20] Seongkwang Kim, ByeongHak Lee, and Jooyoung Lee. Tight security bounds for double-block hash-then-sum macs. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 435–465. Springer, 2020.

[KR00] Ted Krovetz and Phillip Rogaway. Fast universal hashing with small keys and no preprocessing: The polyr construction. In Dongho Won, editor, *Information Security and Cryptology - ICISC 2000, Third International Conference, Seoul, Korea, December 8-9, 2000, Proceedings*, volume 2015 of *Lecture Notes in Computer Science*, pages 73–89. Springer, 2000.

[Kra94] Hugo Krawczyk. LFSR-based Hashing and Authentication. In Yvo Desmedt, editor, *CRYPTO*, volume 839 of *LNCS*, pages 129–139. Springer, 1994.

[Kra02] Matthias Krause. BDD-Based Cryptanalysis of Keystream Generators. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, pages 222–237. Springer, Berlin, Heidelberg, 2002.

[Kro06] Ted Krovetz. Message authentication on 64-bit architectures. In Eli Biham and Amr M. Youssef, editors, *Selected Areas in Cryptography, 13th International Workshop, SAC 2006, Montreal, Canada, August 17-18, 2006 Revised Selected Papers*, volume 4356 of *Lecture Notes in Computer Science*, pages 327–341. Springer, 2006.

[KSV13] D. Karaklajić, J. Schmidt, and I. Verbauwhede. Hardware designer's guide to fault attacks. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 21(12):2295–2306, 2013.

[Küç06] Özgül Küçük. Slide Resynchronization Attack on the Initialization of Grain 1.0. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/044, 2006. `http://www.ecrypt.eu.org/stream`.

[KVW04] Tadayoshi Kohno, John Viega, and Doug Whiting. CWC: A high-performance conventional authenticated encryption mode. In Bimal K. Roy and Willi Meier, editors, *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, volume 3017 of *Lecture Notes in Computer Science*, pages 408–426. Springer, 2004.

[lib23] libsodium documentation. Online, 2023.

[LM12] Michael Lehmann and Willi Meier. Conditional Differential Cryptanalysis of Grain-128a. In *CANS 2012*, pages 1–11. Springer, Berlin, Heidelberg, 2012.

[LNS18] Gaëtan Leurent, Mridul Nandi, and Ferdinand Sibleyras. Generic Attacks Against Beyond-Birthday-Bound MACs. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO I*, volume 10991 of *LNCS*, pages 306–336. Springer, 2018.

[LP18] Atul Luykx and Bart Preneel. Optimal Forgeries Against Polynomial-Based MACs and GCM. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT (1)*, volume 10820 of *LNCS*, pages 445–467. Springer, 2018.

[LPS12] Rodolphe Lampe, Jacques Patarin, and Yannick Seurin. An asymptotically tight security analysis of the iterated even-mansour cipher. In *ASIACRYPT 2012*, pages 278–295. Springer, 2012.

[MAM16] Vasily Mikhalev, Frederik Armknecht, and Christian Müller. On ciphers that continuously access the non-volatile key. *IACR ToSC*, pages 52–79, 2016.

[Mas69] James L. Massey. Shift-register synthesis and BCH decoding. *IEEE Trans. Inf. Theory*, 15(1):122–127, 1969.

[MF21] Arno Mittelbach and Marc Fischlin. *The Theory of Hash Functions and Random Oracles - An Approach to Modern Cryptography*. Information Security and Cryptography. Springer, 2021.

[MG98] C. Madson and R. Glenn. The Use of HMAC-SHA-1-96 within ESP and AH. RFC 2404 (Proposed Standard), November 1998.

[MI11] Kazuhiko Minematsu and Tetsu Iwata. Building blockcipher from tweakable blockcipher: Extending FSE 2009 proposal. In Liqun Chen, editor, *Cryptography and Coding - 13th IMA International Conference, IMACC 2011, Oxford, UK, December 12-15, 2011. Proceedings*, volume 7089 of *Lecture Notes in Computer Science*, pages 391–412. Springer, 2011.

[Min10] Kazuhiko Minematsu. How to Thwart Birthday Attacks against MACs via Small Randomness. In Seokhie Hong and Tetsu Iwata, editors, *FSE*, volume 6147 of *LNCS*, pages 230–249. Springer, 2010.

[MJSC16] Pierrick Méaux, Anthony Journault, François-Xavier Standaert, and Claude Carlet. Towards Stream Ciphers for Efficient FHE with Low-Noise Ciphertexts. In *EUROCRYPT 2016*, pages 311–343. Springer, Berlin, Heidelberg, 2016.

[ML19] Alexander Moch and Eik List. Parallelizable MACs Based on the Sum of PRPs with Security Beyond the Birthday Bound. In *ACNS 2019*, pages 131–151. Springer, 2019.

[MN17] Bart Mennink and Samuel Neves. Encrypted Davies-Meyer and Its Dual: Towards Optimal Security Using Mirror Theory. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO III*, volume 10403 of *LNCS*, pages 556–583. Springer, 2017.

[Moc23] Alexander Moch. Provable security against generic attacks on stream ciphers. *Journal of Mathematical Cryptology*, 17(1):20220033, 2023.

[MS89] Willi Meier and Othmar Staffelbach. Fast correlation attacks on certain stream ciphers. *Journal of Cryptology*, 1(3):159–176, 1989.

[MV04] David A. McGrew and John Viega. The security and performance of the galois/counter mode (GCM) of operation. In Anne Canteaut and Kapalee Viswanathan, editors, *Progress in Cryptology - INDOCRYPT 2004, 5th International Conference on Cryptology in India, Chennai, India, December 20-22,*

2004, Proceedings, volume 3348 of *Lecture Notes in Computer Science*, pages 343–355. Springer, 2004.

[NaC23] NaCl: Networking and Cryptography library. Online, 2023.

[Nan17] Mridul Nandi. Birthday Attack on Dual EWCDM. *IACR Cryptology ePrint Archive*, 2017:579, 2017.

[Nan18] Mridul Nandi. Bernstein Bound on WCS is Tight - Repairing Luykx-Preneel Optimal Forgeries. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO II*, volume 10992 of *LNCS*, pages 213–238. Springer, 2018.

[NLR18] Y. Nir, A. Langley, and I. Rijndael. ChaCha20 and Poly1305 for IETF Protocols. RFC 8439 (Proposed Standard), June 2018.

[NPV17] Valérie Nachef, Jacques Patarin, and Emmanuel Volte. *Feistel Ciphers - Security Proofs and Cryptanalysis*. Springer, 2017.

[Ope14] OpenSSH 6.5 Release Notes, 2014. Accessed: 2023-06-29.

[Ope16] OpenSSL 1.1.0 Series Release Notes, 2016. Accessed: 2023-06-29.

[Pat08] Jacques Patarin. The "Coefficients H" Technique. In Roberto Maria Avanzi, Liam Keliher, and Francesco Sica, editors, *SAC*, volume 5381 of *LNCS*, pages 328–345. Springer, 2008.

[Pat10] Jacques Patarin. Introduction to Mirror Theory: Analysis of Systems of Linear Equalities and Linear Non Equalities for Cryptography. *IACR Cryptology ePrint Archive*, 2010:287, 2010.

[Pat17] Jacques Patarin. Mirror theory and cryptography. *Appl. Algebra Eng. Commun. Comput.*, 28(4):321–338, 2017.

[Pet83] Fabien Petitcolas. La cryptographie militaire. *Journal des sciences militaires*, 9:161–191, 1883.

[Pop15] A. Popov. Prohibiting RC4 Cipher Suites. RFC 7465 (Proposed Standard), February 2015.

[PP10] Christof Paar and Jan Pelzl. *Understanding Cryptography - A Textbook for Students and Practitioners*. Springer, 2010.

[Res18] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, August 2018.

[Rog95] Phillip Rogaway. Bucket Hashing and its Application to Fast Message Authentication. In Don Coppersmith, editor, *CRYPTO*, volume 963 of *LNCS*, pages 29–42. Springer, 1995.

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

[Sha15] Shadowsocks Encryption, 2015. Accessed: 2023-06-29.

[Sho96] Victor Shoup. On Fast and Provably Secure Message Authentication Based on Universal Hashing. In Neal Koblitz, editor, *CRYPTO*, volume 1109 of *LNCS*, pages 313–328. Springer, 1996.

[Sie85] Thomas Siegenthaler. Decrypting a Class of Stream Ciphers Using Ciphertext Only. *IEEE Transactions on Computers*, 34(1):81–85, January 1985.

[Ste07] Dirk Stegemann. Extended BDD-Based Cryptanalysis of Keystream Generators. In *SAC 2007*, pages 17–35. Springer, Berlin, Heidelberg, 2007.

[SWGW21] Yaobin Shen, Lei Wang, Dawu Gu, and Jian Weng. Revisiting the security of dbhts macs: Beyond-birthday-bound in the multi-user setting. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part III*, volume 12827 of *Lecture Notes in Computer Science*, pages 309–336. Springer, 2021.

[TIHM17] Yosuke Todo, Takanori Isobe, Yonglin Hao, and Willi Meier. Cube Attacks on Non-Blackbox Polynomials Based on Division Property. In *CRYPTO 2017*, pages 250–279, Cham, 2017. Springer International Publishing.

[TIM+18] Yosuke Todo, Takanori Isobe, Willi Meier, Kazumaro Aoki, and Bin Zhang. Fast Correlation Attack Revisited. In *CRYPTO 2018*, pages 129–159, Cham, 2018. Springer International Publishing.

[TMA20] Yosuke Todo, Willi Meier, and Kazumaro Aoki. On the Data Limitation of Small-State Stream Ciphers: Correlation Attacks on Fruit-80 and Plantlet. In *SAC 2019*, pages 365–392, Cham, 2020. Springer International Publishing.

[WC81] Mark N. Wegman and Larry Carter. New Hash Functions and Their Use in Authentication and Set Equality. *J. Comput. Syst. Sci.*, 22(3):265–279, 1981.

[Wir23] WireGuard: Next Generation Kernel Network Tunnel. Online, 2023.

[WLLM19] Shichang Wang, Meicheng Liu, Dongdai Lin, and Li Ma. Fast Correlation Attacks on Grain-like Small State Stream Ciphers and Cryptanalysis of Plantlet, Fruit-v2 and Fruit-80. Cryptology ePrint Archive, Report 2019/763, 2019. `https://eprint.iacr.org/2019/763`.

[Yas11] Kan Yasuda. A New Variant of PMAC: Beyond the Birthday Bound. In Phillip Rogaway, editor, *CRYPTO*, volume 6841 of *LNCS*, pages 596–609. Springer, 2011.

[YL06] T. Ylonen and C. Lonvick. The Secure Shell (SSH) Transport Layer Protocol. RFC 4253 (Proposed Standard), January 2006.

[ZGM17] Bin Zhang, Xinxin Gong, and Willi Meier. Fast Correlation Attacks on Grain-like Small State Stream Ciphers. *IACR ToSC*, 2017(4):58–81, Dec. 2017.

[ZW09] Haina Zhang and Xiaoyun Wang. Cryptanalysis of Stream Cipher Grain Family. Cryptology ePrint Archive, Report 2009/109, 2009. `http://eprint.iacr.org/2009/109`.

[ZWSW12] Liting Zhang, Wenling Wu, Han Sui, and Peng Wang. 3kf9: Enhancing 3GPP-MAC beyond the Birthday Bound. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT*, volume 7658 of *LNCS*, pages 296–312. Springer, 2012.