Cryptanalysis of STAP Primitives: Efficiency Meets Vulnerability

Irati Manterola Ayala

Thesis for the degree of Philosophiae Doctor (PhD)
University of Bergen, Norway
2025

UNIVERSITY OF BERGEN

Cryptanalysis of STAP Primitives: Efficiency Meets Vulnerability

Irati Manterola Ayala

Thesis for the degree of Philosophiae Doctor (PhD)
at the University of Bergen

Date of defense: 05.12.2025

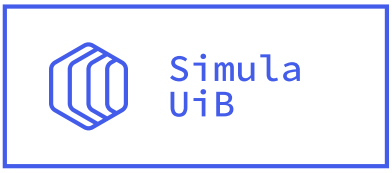
Year: 2025
Title: Cryptanalysis of STAP Primitives: Efficiency Meets Vulnerability

Name: Irati Manterola Ayala
Print: Skipnes AS / University of Bergen

Scientific Environment

This doctoral research has been conducted within the research environment of Simula UiB and the Department of Informatics at the University of Bergen (UiB). Throughout my PhD studies, I have been employed by and based at Simula UiB while being formally enrolled as a PhD student at the Department of Informatics at UiB. My research has been supervised by Håvard Raddum, chief research scientist at Simula UiB, and Øyvind Ytrehus, professor at UiB.

Additionally, part of the work presented in this thesis has been carried out with the support of the COSINUS (Collaboration On Secrecy to Investigate New USes) associate team, a collaboration between COSMIQ-Inria and Simula UiB.



Acknowledgements

Håvard, my thanks goes to you. You have been an incredible supervisor — always present, always thoughtful, and always looking out for me. You have guided me without ever making me feel limited, trusted me with responsibility, and considered my needs at every step. I have learned a lot from you, not just in cryptography but in how to work with patience and care. You have been understanding in moments when I needed it most, and your belief in me has shaped this journey more than you know.

To my lovely *hug club* — Issam, Sarah, Sujash, Stian, Tamás, Atharva, and Imane. You have been my everyday fuel, my safe space, and my source of joy. Thank you for the endless love, the daily updates, the vents, the rants, the gossip, and for standing by me through my hardest days. I owe you more than words can hold, and no matter where life takes me, I will imagine a hug from each of you every morning before I start my day. And to everyone at Simula UiB — you made me feel home from the very first moment, and in that warmth I found the confidence to grow in ways I never thought I could. I will forever miss our lunches, fikas, and every small detail that made being here so special.

A mi familia. Gracias por siempre creer en mí y en todas mis aventuras, aunque para ello haya tenido que estar lejos de vosotros durante tanto tiempo. Paso el año contando los días para Navidad para poder abrazaros de nuevo. Yaya, llamarte cada domingo me da luz y vida, y es toda la fuerza que necesito para seguir luchando. Haizea, todo lo que hago es para darte el ejemplo que te mereces, y ojalá pueda hacerte sentir tan orgullosa como lo estoy yo de ti. Os quiero y os echo de menos más de lo que os imagináis.

Baby, thank you for being such a special part of this chapter of my life. You've been my biggest fan, always supporting me and reminding me of what I'm capable of, and through that you've helped me grow in ways I'll carry with me forever. Your positivity, kindness, and all the little gestures of care lifted my spirits and made the hardest days feel so much lighter. I'll always treasure the memories of what we shared and everything we built together along the way, and I feel so lucky to share this milestone with you. Love you.

And to you — me. This may be unusual, but you deserve it. Let these pages stand as proof of your strength and resilience, not only in your work, but in how you've carried yourself through all the hardships life has thrown at you. Always remember the moments when you were exhausted, overwhelmed, or lost, yet still found the courage to keep moving forward. Trust yourself, and, above all, never forget to pause and be proud of the woman you have become — your inner child would be so impressed. You owe yourself that much.

Abstract

This thesis explores the security of symmetric cryptographic primitives tailored for advanced protocols such as Fully Homomorphic Encryption, Zero-Knowledge Proofs, and Multi-Party Computation—collectively referred to as Symmetric Techniques for Advanced Protocols (STAP). These emerging protocols are increasingly central to modern cryptographic applications, enabling privacy-preserving computation and secure outsourcing, among others. However, they impose strict efficiency requirements that diverge sharply from those assumed in traditional symmetric cryptography.

Where classical designs optimize for performance on standard hardware and rely on binary logic and bit-level operations, STAP constructions often operate over large finite fields or rings, and aim to reduce costly operations such as multiplication or complex non-linear transformations. These constraints require a fundamental rethinking of how such symmetric primitives are designed and analyzed.

The core of this work is a cryptanalytic study of several recently proposed STAP constructions. Through a combination of adapted classical techniques and novel attacks developed to exploit the structural properties of these primitives, the thesis uncovers significant weaknesses in designs such as *Rubato*, *Griffin*, *Anemoi*, *Arion*, and a recent weak pseudorandom function. Many of these constructions fail to meet their stated security claims, and in some cases are broken far below their intended security levels.

Taken together, these results suggest that the design of symmetric primitives in advanced settings may be outpacing the cryptanalysis applied to them, and highlight the urgent need for more rigorous frameworks to assess algebraic robustness in non-classical settings. Beyond specific attacks, this work contributes toward a broader understanding of how cryptographic design should evolve in response to new application contexts, ultimately guiding the development of STAP constructions that better balance efficiency and resilience in this evolving landscape.

Sammendrag

Denne avhandlingen undersøker sikkerheten til symmetriske kryptografiske primitiver tilpasset avanserte protokoller som Fully Homomorphic Encryption, Zero-Knowledge Proofs og Multi-Party Computation – samlet omtalt som Symmetriske Teknikker for Avanserte Protokoller (STAP). Disse nye protokollene spiller en stadig viktigere rolle i moderne kryptografiske anvendelser, og muliggjør blant annet personvernbevarende beregninger og sikker outsourcing. Samtidig stiller de strenge effektivitetskrav som skiller seg markant fra forutsetningene bak tradisjonell symmetrisk kryptografi.

Mens klassiske konstruksjoner er optimalisert for ytelse på standard maskinvare og baserer seg på binær logikk og bitvise operasjoner, arbeider STAP-konstruksjoner ofte over store endelige kropper eller ringer, og forsøker å redusere kostbare operasjoner som multiplikasjon eller komplekse ikke-lineære transformasjoner. Disse begrensningene krever en grunnleggende nytenkning rundt både design og analyse av symmetriske primitiver.

Kjernen i dette arbeidet er en kryptanalytisk studie av flere nylig foreslåtte STAP-konstruksjoner. Ved å kombinere tilpassede klassiske teknikker med nye angrep som utnytter strukturelle egenskaper ved disse primitivene, avdekker avhandlingen betydelige svakheter i design som *Rubato*, *Griffin*, *Anemoi*, *Arion*, samt en nylig foreslått svak psevdorandom funksjon. Mange av disse konstruksjonene oppfyller ikke sine egne sikkerhetsmål, og i enkelte tilfeller brytes de ved langt lavere sikkerhetsnivåer enn tiltenkt.

Samlet sett tyder resultatene på at utviklingen av symmetriske primitiver for avanserte bruksområder kan ha gått raskere enn den kryptanalysen som anvendes på dem. Dette fremhever behovet for mer robuste rammeverk for å vurdere algebraisk motstandskraft i ikke-klassiske settinger. Utover konkrete angrep bidrar dette arbeidet til en bredere forståelse av hvordan kryptografisk design bør utvikles i møte med nye anvendelseskontekster, og kan dermed legge grunnlaget for utviklingen av STAP-konstruksjoner som bedre balanserer effektivitet og motstandskraft i et landskap i rask endring.

List of Publications

1. Lorenzo Grassi, Irati Manterola Ayala, Martha Norberg Hovd, Morten Øygarden, Håvard Raddum, and Qingju Wang, *Cryptanalysis of Symmetric Primitives over Rings and a Key Recovery Attack on* Rubato, Advances in Cryptology – CRYPTO 2023, Lecture Notes in Computer Science (LNCS), vol 14083, pages 305–339, 2023.

2. Augustin Bariant, Aurélien Boeuf, Axel Lemoine, Irati Manterola Ayala, Morten Øygarden, Léo Perrin, and Håvard Raddum, *The Algebraic FreeLunch: Efficient Gröbner Basis Attacks Against Arithmetization-Oriented Primitives*, Advances in Cryptology – CRYPTO 2024, Lecture Notes in Computer Science (LNCS), vol 14923, pages 139–173, 2024.

3. Irati Manterola Ayala and Håvard Raddum, *Zeroed Out: Cryptanalysis of Weak PRFs in Alternating Moduli*, IACR Transactions on Symmetric Cryptology, 2025, vol 2, pages 1-15.

Contents

Chapter 1

Cryptography: the Art and Science of Secrecy

At its core, cryptography is the practice of securing communication and information from adversarial entities. It is the art of transforming readable messages into incomprehensible ones, ensuring that only the intended recipient can access their true meaning. Derived from the Greek words *kryptos* (hidden/secret) and *graphein* (to write), cryptography is both a science and an art—one that has shaped the course of history, and remains a cornerstone of modern security.

While often associated with espionage, military strategy, and clandestine messages, cryptography today extends far beyond these domains. It underpins online transactions, secures private communications, and ensures trust in digital interactions. Initially, its primary goal was to guarantee the confidentiality of information in transit, making it accessible solely to the intended recipient despite the presence of eavesdroppers. However, its role has expanded to encompass authentication (verifying identities), and integrity (ensuring data remains unaltered during transmission) [59]. In an era where data is as valuable as currency, cryptography plays a crucial role in safeguarding information in an increasingly interconnected world.

The Core Principle of Encryption. One of the central tools of cryptography is encryption, which relies on a simple yet powerful idea: a secret, properly protected, allows us to secure information by blending the secret with data in a way that only authorized parties can reverse. Only those with knowledge of the secret can retrieve the original information. This secret, referred to as the *(secret) key*, enables the transformation of readable data into an unreadable form (*encryption*) and its restoration to the original

form (*decryption*) when used correctly.

A simple example illustrates this core principle: Suppose one person wants to send a message—the *plaintext*—to another but does not want anyone else to read it. The sender encrypts the plaintext using a key, transforming it into an unreadable *ciphertext*, which is then transmitted to the recipient. With the correct key, the recipient decrypts the ciphertext and recovers the original plaintext. The keys used for encryption and decryption may be the same or different, depending on the scheme. This fundamental process, illustrated in Figure 1.1, lies at the heart of all cryptographic systems, from ancient ciphers to modern encryption protocols.

Figure 1.1: The encryption and decryption process: plaintext is transformed into ciphertext using an encryption key, and only the correct decryption key can reverse the transformation.

Historically, cryptographic keys were often just short numbers or phrases. However, as computational power has grown, these simple keys have become inadequate. Modern cryptographic keys are designed to withstand *brute-force attacks*, where an adversary systematically tries all possible values until the correct key is found. While brute-force attacks do not exploit any structural weakness in the encryption scheme itself, they are the baseline threat any system must resist. If the adversary succeeds, they recover the key and can decrypt all messages protected by it—effectively rendering the encryption useless.

In theory, brute-force attacks are always possible, as there is nothing preventing an eavesdropper from trying every key. However, in practice, the sheer computational effort required makes brute-force infeasible for well-chosen cryptographic key lengths. For instance, a 128-bit key does not seem particularly long, yet it allows for 2^{128} possible keys—a number so large that even with the fastest supercomputers, exhaustively searching through all possible keys would take an astronomical amount of time [76]. The strength of a cryptographic scheme therefore often relies on the absence of more efficient attacks than brute force—a guiding principle in the study of *cryptanalysis*, the focus of this thesis.

Cryptanalysis: The Adversarial Perspective. If cryptography is the art of keeping secrets, cryptanalysis is the science of finding them. Cryptanalysts analyze cryp-

tographic systems, searching for weaknesses that might allow an adversary to decrypt messages. In this context, to "break" a system typically means to defeat its intended *security claim*—undermining its promise to keep data private, authentic, or unchanged. Such claims are usually quantified in bits, reflecting the estimated computational effort required to compromise the scheme. A system with claimed n-bit security is expected to withstand attacks requiring about 2^n elementary operations—such as encryptions or key guesses. These claims should be clearly included in the cryptographic scheme and justified by mathematical proofs or complexity assumptions, and serve as benchmarks against which attacks are evaluated. If an attack is found that succeeds with significantly fewer operations than advertised, the scheme is considered broken and must be strengthened or replaced. Whether or not an attack can *realistically* be mounted, any method that performs significantly better than the stated claim calls the security of the primitive into question.

The development of cryptographic systems is inherently adversarial, as those who design algorithms must anticipate and defend against those who attempt to break them. To ensure resilience against attacks, modern cryptographic research rigorously tests algorithms by simulating said attacks. Each layer of a cryptographic system undergoes rigorous scrutiny, as even the smallest vulnerability can compromise security. As long as the key remains secret, an adversary should not be able to break the system or extract sensitive information. Because of this, a primary goal in cryptanalysis is to recover the secret key. Doing so allows the adversary to decrypt all past and future communications protected under that key, and renders the scheme fundamentally insecure. *Key recovery attacks* therefore represent a critical threat model, and much of modern cryptographic research—including the work in this thesis—focuses specifically on developing, improving, and analyzing such attacks.

However, recovering the secret key is only one possible objective of an attack. Cryptanalysis also includes attempts to distinguish a ciphertext from random data, to forge valid signatures without knowledge of the signing key, to manipulate encrypted messages in meaningful ways, or to extract partial information about the plaintext. These objectives are studied under different attack scenarios, which vary in the amount of information or control granted to the adversary. At the weakest end lies the ciphertext-only attack (COA), where the attacker has access only to intercepted ciphertexts. More powerful is the known-plaintext attack (KPA), in which some plaintext–ciphertext pairs are available, allowing structural patterns to be exploited. Stronger still are chosen-plaintext attacks (CPA), where the adversary can request ciphertexts for plaintexts of their choice, and chosen-ciphertext attacks (CCA), where the adversary may obtain decryptions of arbitrary ciphertexts (excluding the target one). These scenarios reflect practical threat environments and challenge even well-established schemes, forcing cryptographers to stay

ahead of potential threats.

Cryptanalysts must think like attackers to uncover vulnerabilities before real adversaries do. The first adversaries of any system are always its designers, who must scrutinize their own work with the mindset of a potential attacker. A fundamental question in cryptographic research is not just "Is this secure?" but rather, "How would I break this?" This thesis adopts that perspective, analyzing the strength of cryptographic schemes, improving existing attacks, and developing new ones where possible.

This adversarial dynamic is not a recent development—throughout history, cryptanalysis has been a driving force behind the evolution of cryptography: whenever an encryption method is broken, a more secure one must be developed.

A Brief Journey Through Time. The history of cryptography [51, 74] is, at its core, a story of cryptanalysis. From ancient times to the digital age, the evolution of ciphers has been driven by the attacks that rendered their predecessors obsolete. The earliest known techniques—such as the substitution methods used by Egyptian scribes—were devised to obscure messages, but offered little resistance to a determined analyst. The Spartans' scytale provided secure military communication through simple transposition, until the method became known and could be reversed. The Romans' Caesar cipher, shifting letters in the alphabet by a fixed amount, represents one of the earliest examples of systematic encryption, but is trivial to break due to its limited number of possible shifts. Even the more complex general substitution cipher, in which each letter of the alphabet is replaced with a different letter or symbol according to a fixed scheme, though much harder to brute-force, fell to frequency analysis, which exploits the fact that certain letters appear more frequently than others in natural language texts.

Each new advance emerged as a countermeasure to a successful attack. In the Renaissance, polyalphabetic ciphers like the Vigenère sought to defeat frequency analysis by cycling through multiple shifting alphabets. This held for centuries, but ultimately, cryptanalysts developed systematic methods to break them. It was during World War II that cryptography reached an unprecedented level of sophistication. The German *Enigma* machine, used to encrypt military communications, posed one of the greatest cryptographic challenges in history. Yet it too succumbed to persistent cryptanalytic effort—most famously by Allied cryptanalysts, led by figures like Alan Turing, marking a turning point in the war and a pivotal moment in the history of cryptography.

The rise of computers radically accelerated both sides of the contest, as cryptography evolved from a manual craft into a rigorous mathematical discipline. The Data Encryption Standard (DES) [60] (introduced in the 1970s as a US government standard)

stood for decades until advances in computing power made exhaustive key search feasible. It was replaced by the Advanced Encryption Standard (AES) [61] (standardised in 2001), designed to withstand known attack methods and offer greater security margins. Similarly, the introduction of public-key cryptography in the 1970s pioneered by Rivest, Shamir, and Adleman (RSA) [69], which revolutionized the field by solving how to establish secure communication between two parties who have never met before and therefore have not been able to agree on a secret key in a secure manner, was not merely a conceptual leap—it directly addressed the limitations of symmetric key exchange that adversaries could exploit.

Today, cryptography—embedded in nearly every aspect of digital communication— continues to evolve under the pressure of cryptanalysis. The protocols securing our messages, financial transactions, medical records, and software updates are therefore the latest winners in a centuries-long arms race, and their strength determines the safety of our interconnected world. This invisible yet indispensable force underpins our digital society—one that is an integral part of the everyday life of the average person.

A Tale of Two Cryptographies: Symmetric and Asymmetric. Cryptographic systems can broadly be divided into two classes: symmetric and asymmetric cryptography. Together, they form the foundation of today's security protocols, each with its own strengths, applications, and vulnerabilities.

Although symmetric cryptography emerged earlier and forms the central focus of this thesis, we begin with a brief overview of asymmetric techniques. This detour is included for completeness, and because asymmetric cryptography often supports or interacts with symmetric methods in modern systems. Before diving into specifics, we first set the stage by outlining their guiding principles and the mechanisms that make them secure—or vulnerable.

1.1 Asymmetric Cryptography: Public and Private Keys

Asymmetric cryptography, also known as public-key cryptography, marked a fundamental shift from traditional encryption methods that relied on a single shared secret for both encryption and decryption. It employs two mathematically linked keys: a public key, which is openly shared, and a private key, which must be kept secret.

How It Works. The core principle of asymmetric cryptography is that while the public key can encrypt data, only the corresponding private key can decrypt it. This fundamental property allows anyone with access to the recipient's public key to send them a confidential message that only the recipient can decrypt.

To illustrate, consider a scenario where one person wants to send a confidential message to another. The recipient first generates a key pair—one public, one private—sharing the former while keeping the latter secret. The sender encrypts the message with the public key, and only the corresponding private key—which only the recipient possesses—can decrypt it, ensuring confidentiality even if the ciphertext and public key are intercepted. Figure 1.2 illustrates this process.

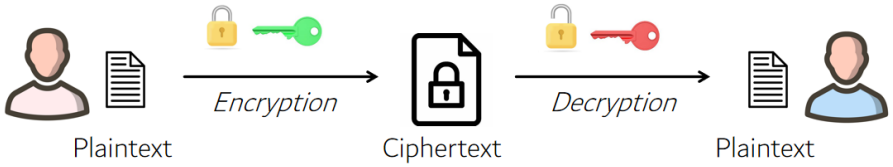


Figure 1.2: Asymmetric encryption process: a message is encrypted with the recipient's public key and decrypted with their private key.

In addition to encryption, asymmetric cryptography supports *digital signatures*, which allow a sender to authenticate messages. The sender signs a message using their private key, and anyone with the corresponding public key can verify the signature. Figure 1.3 illustrates this process.

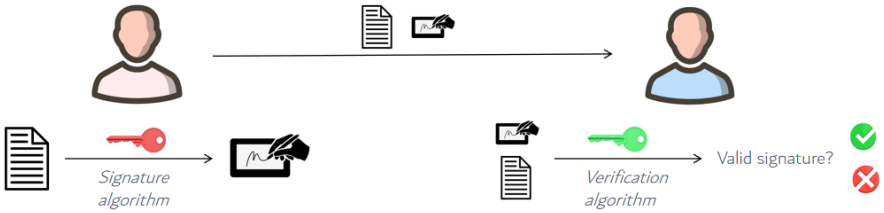


Figure 1.3: Digital signature process: a message is signed with the sender's private key and verified using their public key.

Applications. Public-key cryptography is integral to modern cybersecurity and is widely used in various applications. A critical function is in *key exchange mechanisms* [76, Chapter 14], where public-key cryptography securely establishes a shared symmetric key over an untrusted channel—after which more efficient symmetric encryption takes over, as we will explore later. This approach underlies protocols such as TLS (Transport Layer Security) [68], SSH (Secure Shell) [84], and Signal [73], which provide encrypted

communication over insecure networks. Digital signatures also have crucial applications—for example, in certificate-based authentication schemes like *Public Key Infrastructure* (PKI) [76, Chapter 16], or in cryptocurrencies where transactions are signed to prove ownership of money being paid [16, 21].

Security Considerations. The security of asymmetric cryptographic schemes relies on mathematical problems that are believed to be computationally infeasible to solve within a reasonable timeframe. The most widely used problems underpinning public-key encryption include *Integer Factorization* (used in RSA), the *Discrete Logarithm Problem* (used in Diffie-Hellman key exchange [30] and DSA [63]), and the *Elliptic Curve Discrete Logarithm Problem* [48, pp 784–787]. These problems resist efficient solution on classical computers, which forms the basis of trust in these systems.

However, public-key cryptography faces a serious long-term threat from quantum computing. Shor's algorithm [72], if implemented on a sufficiently powerful quantum computer, could efficiently solve these problems, rendering current public-key cryptosystems insecure. As a result, *post-quantum cryptography* has become a major research area [50], and several new algorithms—such as *CRYSTALS-Kyber* [64] and *CRYSTALS-Dilithium* [65]—are being standardized by the *National Institute of Standards and Technology* [62].

Since this thesis focuses on symmetric cryptography, which is less affected by quantum attacks, we do not discuss these developments further.

1.2 Symmetric Cryptography: One Key to Rule Them All

Symmetric cryptography, also known as secret-key cryptography, is a method of encryption in which the same key is used for both encryption and decryption, therefore relying on a single shared secret key that both the sender and receiver must have access to. It is valued for its efficiency and speed, but it comes with the challenge of securely distributing and managing the shared secret key between communicating parties. This thesis focuses primarily on symmetric cryptography, examining some particular symmetric schemes and the security challenges they face in depth.

How It Works. The fundamental principle of symmetric cryptography is that anyone with access to the shared key can both encrypt and decrypt messages. To illustrate, consider a scenario where two people want to exchange a confidential message. First,

they must establish and securely exchange a common secret key, which must remain private. The sender then encrypts a message with this key, and the recipient—who holds the same key—decrypts it to recover the plaintext. The security of this system hinges entirely on keeping the key secret—if an attacker obtains it, they can decrypt all communications at will. Figure 1.4 illustrates this process.

Figure 1.4: Symmetric encryption process: only those who possess the shared secret key can encrypt and decrypt messages.

Applications. Symmetric encryption is widely employed in various domains requiring efficient and secure data protection. It is essential in securing online communications, playing a fundamental role in protocols like TLS (for encrypting web traffic) [68], encrypting files and disk storage, and ensuring confidentiality [73] in secure messaging apps such as Signal and WhatsApp. Furthermore, symmetric encryption plays a crucial role in Virtual Private Networks (VPNs), where it protects data in transit between remote users and corporate networks.

Notably, many of these applications also involve asymmetric cryptography, as we have previously discussed. This overlap reflects how modern cryptographic systems typically interweave asymmetric and symmetric techniques, rather than using them in isolation. In relation to symmetric cryptography, its high performance and low computational overhead make it especially suitable for real-time and large-scale data encryption.

Challenges and Security Considerations. While symmetric cryptography provides strong confidentiality and efficiency, its major drawback lies in key management and distribution. Since both communicating parties must have access to the same secret key, a secure method of key exchange must be established beforehand. This challenge makes symmetric cryptography less practical for large-scale communications or scenarios where users frequently change.

To address this limitation, most modern applications employ *hybrid encryption* [76, Chapter 21], which combines the strengths of both symmetric and asymmetric cryptography. In this setup, asymmetric encryption is used to securely exchange a randomly generated symmetric key, which is then used for bulk encryption of the actual data.

Specifically, the sender encrypts the symmetric key using the recipient's public key and transmits the encrypted key alongside the ciphertext. Upon receiving it, the recipient uses their private key to decrypt and retrieve the symmetric key, which then allows them to efficiently decrypt the data. This approach ensures both security and efficiency: asymmetric encryption guarantees the secure transmission of the symmetric key, while symmetric encryption enables fast and efficient encryption of large volumes of data. Figure 1.5 illustrates this process.

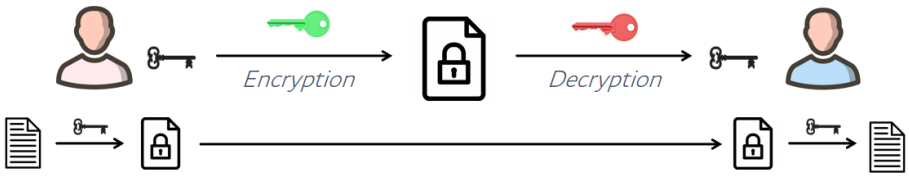


Figure 1.5: Hybrid encryption: a symmetric key is securely exchanged using asymmetric encryption and then used for efficient bulk encryption.

This hybrid model is particularly relevant for the techniques studied in this thesis. In Chapter 2, we introduce *Symmetric Techniques for Advanced Protocols* (STAPs), which are frequently deployed in hybrid settings like the one described here. Hybrid encryption thus serves not only as a common real-world strategy, but also as a key motivation for exploring advanced symmetric designs in depth.

We now delve into the core primitives found in symmetric cryptography, analyzing their properties and potential vulnerabilities.

1.2.1 Symmetric Primitives

Classical symmetric encryption schemes are divided into stream ciphers and block ciphers, each following distinct design principles. Beyond encryption, cryptographic hash functions play a crucial role in ensuring data integrity without requiring a key, while pseudorandom functions (PRFs) generate outputs computationally indistinguishable from random values, serving as building blocks for authentication and key derivation. In the remainder of this section, we explore all four of these primitives in more detail.

Stream Ciphers. Stream ciphers are algorithms that combine plaintext digits with a keystream produced by a keystream generator to perform encryption or decryption. They encrypt data sequentially, processing bits or bytes one at a time rather than in fixed-size blocks.

At the core of a stream cipher is a keystream generator, which produces a sequence of pseudorandom bits or bytes derived from a secret key k and, typically, a public initialization vector (IV), which may be public and deterministically generated, but crucially never reused with the same key. The encryption process consists of combining each plaintext digit m_i with the corresponding keystream digit k_i using an operation such as XOR to produce the ciphertext $c_i = m_i \oplus k_i$. Decryption is performed by the recipient, who generates the same keystream and combines it with the ciphertext to recover the original plaintext. Figure 1.6 illustrates the general structure of a stream cipher.

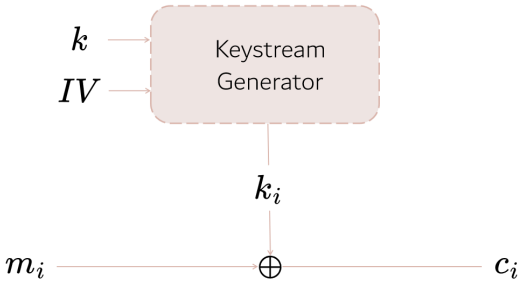


Figure 1.6: General structure of a stream cipher. The keystream generator produces a pseudorandom sequence, which is XOR-ed with the plaintext to produce the ciphertext.

This method ensures fast and efficient encryption while making stream ciphers highly adaptable to varying message lengths. The keystream itself is generated dynamically, with the internal state of the cipher evolving after each step based on an update function.

Stream ciphers are particularly well-suited for real-time applications due to their low latency and ability to encrypt data on the fly. They are commonly used in scenarios requiring fast encryption, such as wireless networks, real-time voice and video communication, and secure messaging. Notable examples include the once-popular RC4 [76, Chapter 7] and modern ciphers like ChaCha20 [11], which is widely employed in security protocols due to its robustness against known attacks.

Block Ciphers. Given a key k, a block cipher E_k is a family of permutations that maps a fixed-size plaintext block (typically 64 or 128 bits long) to a ciphertext block of the same size. Block ciphers are extensively used in securing digital communications, data encryption at rest, and cryptographic protocols such as TLS and IPsec [68, 71]. Unlike stream ciphers, which encrypt data continuously, block ciphers process entire data blocks independently, applying a series of transformation rounds.

Block ciphers are typically constructed by iteratively applying a round function, which consists of transformations such as substitution, permutation, and linear operations. The

encryption process can be expressed as

$$E_k(m) = F_{k_r} \circ \cdots \circ F_{k_1}(m),$$

where each round function F_{k_i} contributes to the overall security of the cipher, applying structured operations that introduce *confusion* (replacing data elements with values that are hard to predict, typically achieved through key additions and S-boxes) and *diffusion* (spreading small input changes across the entire ciphertext). The round keys $k_1, k_2, \ldots, k_r$ are derived from the master key k using a key schedule algorithm, ensuring that each round uses a different subkey. The number of rounds is chosen to ensure a balance between security and efficiency. Figure 1.7 illustrates the general structure of a block cipher, showing how the plaintext is processed through multiple encryption rounds to produce a ciphertext.

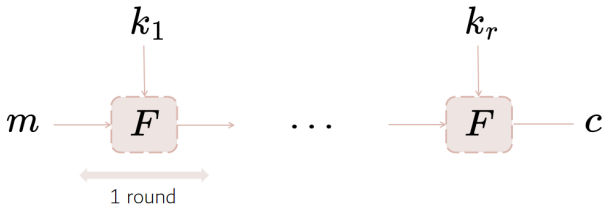


Figure 1.7: General structure of a block cipher. The plaintext is processed through multiple encryption rounds to produce the ciphertext.

A block cipher is considered secure if, for a randomly chosen key, it behaves as a pseudo-random permutation, meaning its output is indistinguishable from that of a truly random permutation. The existence of a distinguisher—an algorithm that can differentiate the cipher from a random permutation—is an undesirable property for such cryptographic primitive and often leads to attacks recovering the secret key.

Two major design principles dominate block cipher construction: Feistel networks and Substitution-Permutation Networks (SPNs).

- **Feistel Networks.** In a Feistel cipher, the input block is divided into two halves: x_L and x_R. Each round transforms these halves as follows:

  $$(x_L, x_R) \to (F_{k_i}(x_L) \oplus x_R, x_L).$$

  This structure allows encryption and decryption to follow a similar process, with the only difference being that the order of the round keys must be reversed. This

makes Feistel ciphers highly practical, as the same code or hardware circuit can be used for both encryption and decryption. A well-known example is the Data Encryption Standard (DES), which was widely used but is now considered insecure due to its small 56-bit key size. Figure 1.8 illustrates the general structure of a Feistel network.

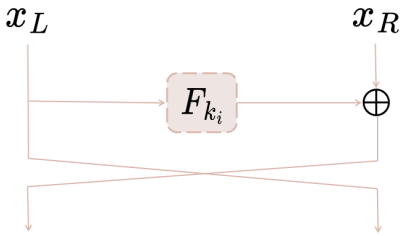


Figure 1.8: General structure of a Feistel network, where each round transforms the left and right halves of the input block using a round function and XOR operation.

- **Substitution-Permutation Networks (SPNs).** An SPN consists of three key components: an S-box layer for substitution, a diffusion layer for spreading input bits, and a key addition layer. Figure 1.9 provides an overview of the round function of an SPN structure. The Advanced Encryption Standard (AES), the most widely used block cipher, follows the SPN structure. It operates on 128-bit blocks and supports key sizes of 128, 192, or 256 bits, and is considered the gold standard in symmetric encryption due to its efficiency and the extensive cryptanalysis supporting its security.

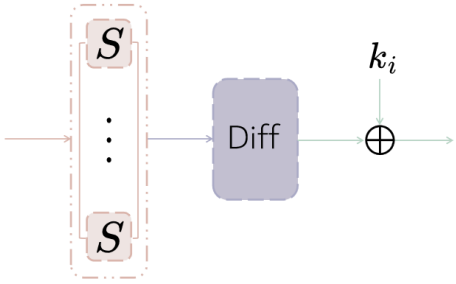


Figure 1.9: General structure of the round function of a Substitution-Permutation Network (SPN), illustrating the sequential application of substitution (S-box), diffusion, and key addition layers.

Hash Functions. A hash function H maps an input x of arbitrary length to a fixed-length output y, often called the *digest*. Cryptographic hash functions play a crucial

role in data integrity, password storage, and digital signatures [59, Chapter 9]. A secure cryptographic hash function should satisfy the following properties:

- *Collision resistance*: It should be computationally infeasible to find two distinct inputs $x_1 \neq x_2$ such that $H(x_1) = H(x_2)$.

- *Preimage resistance:* Given a hash output y, it should be computationally infeasible to find any input x such that $H(x) = y$.

- *Second preimage resistance:* Given an input x_1, it should be computationally infeasible to find another input x_2 such that $H(x_1) = H(x_2)$.

Beyond these, an ideal hash function should behave like a random function while being deterministic and efficiently computable.

There are multiple paradigms for constructing hash functions. The classical approach is the Merkle–Damgård construction [28]. However, hash functions are relevant to this work primarily in the context of sponge-based constructions. For that reason, we focus on the sponge construction below, which plays a central role in modern symmetric protocols discussed in this thesis.

The **Sponge Construction**, introduced in 2007 by Bertoni et al. [12], is parameterized by a *rate* r and a *capacity* c, where $r+c$ determines the internal state size. A well-chosen capacity ensures the desired security level of the hash function. The sponge construction generalizes hash function design by operating in two distinct phases:

1. *Absorption phase:* The message is divided into blocks, and each block is XOR-ed into the first r bits of an internal state before applying a permutation on the full internal state.

2. *Squeezing phase:* The final digest is extracted in blocks, applying the permutation between extractions.

Keccak, the algorithm behind SHA-3 [66], follows this structure. Figure 1.10 illustrates the general construction.

Pseudorandom Functions. A pseudorandom function (PRF) is a keyed deterministic function that, given a secret key, maps inputs to outputs that appear indistinguishable from truly random values. Formally, this is captured by the following thought experiment: an adversary is given black-box (or *oracle*) access to a function—either a PRF $F_k(\cdot)$ (for some fixed secret key k) or a truly random function $R(\cdot)$—allowing them to

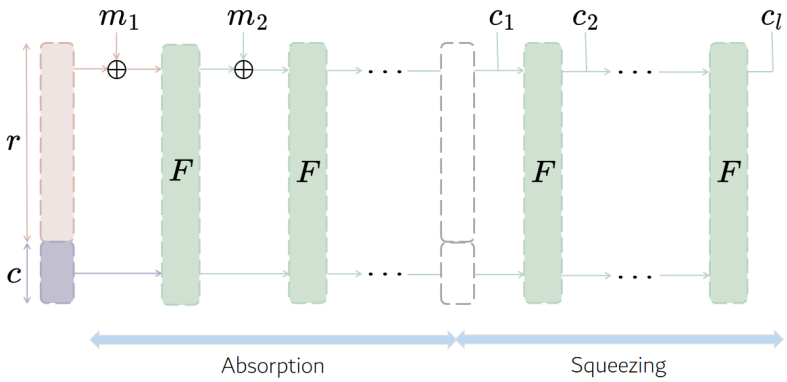


Figure 1.10: General structure of the sponge construction, where input data is absorbed into an internal state and later squeezed to produce the output.

query the function on inputs of their choice and observe the outputs. The adversary's task is then to determine which one of the functions it is interacting with, which for a good PRF, they should be unable to do.

More precisely, for all efficient adversaries $\mathcal{A}$, the distinguishing *advantage* (i.e., how much better the adversary performs than random guessing):

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{PRF}} := \left|\Pr[\mathcal{A}^{F_k} = 1] - \Pr[\mathcal{A}^{R} = 1]\right|$$

should be negligible in the security parameter, so that $\mathcal{A}$ cannot meaningfully distinguish a PRF from a truly random function. Here, the probabilities are taken over the random choice of the key k, the randomness of the truly random function R, and any internal randomness used by $\mathcal{A}$.

PRFs play a fundamental role in symmetric cryptography due to their flexibility and composability [36, Chapter 3], serving as building blocks for a wide range of cryptographic protocols, including encryption, message authentication codes (MACs), key derivation functions (KDFs), and secure multiparty computation (MPC). They are also commonly used as round functions in Feistel networks, contributing to the security of block ciphers [53]. Moreover, PRFs are tightly connected to the notion of pseudorandom generators (PRGs), and often serve as the conceptual foundation for more complex cryptographic objects like pseudorandom permutations (PRPs).

Security requirements for PRFs vary depending on their application. In some cases, a weak notion of security, known as a *weak PRF*, is sufficient. A weak PRF only guarantees security when the inputs queried by the adversary are chosen uniformly at random, rather than adaptively chosen ones [75]. This relaxation still ensures unpredictability

while allowing more efficient or simpler constructions. In contrast, a *strong PRF* must provide pseudorandomness even against adversaries that can choose their query inputs adaptively, which is a stronger security notion required in many cryptographic applications where inputs may be partially predictable or controllable by an attacker.

1.2.2 Attacks on Symmetric Primitives

While symmetric cryptography provides strong security guarantees, it is not impervious to attack. Over the decades, cryptanalysts have developed a rich arsenal of techniques to identify and exploit weaknesses in modern symmetric primitives.

In an ideal scenario, a symmetric cipher with an n-bit key should provide n-bit security. However, historical advancements in cryptanalysis as well as hardware speed-ups have repeatedly lowered the effective security of certain schemes, prompting the need for stronger primitives. While earlier cryptographic standards considered 80-bit security sufficient, modern security requirements typically demand at least 128-bit security to withstand threats.

Because one of the central goals of this thesis is to analyze the security of some advanced symmetric primitives, it is crucial to understand the types of attacks that may be mounted against them. Below, we provide an overview of some of the most common attack techniques, each targeting different properties of the cipher. These attacks often leverage statistical patterns, algebraic structures, or structural flaws in the cryptographic design. The resistance of symmetric primitives to classical and modern forms of cryptanalysis is a recurring theme throughout this work.

Statistical Attacks. Statistical cryptanalysis exploits deviations from ideal randomness in encryption algorithms. Differential and linear attacks are two prominent techniques that analyze how small differences in inputs influence the encryption process. These attacks identify predictable biases that deviate from the behavior of an ideal random permutation, enabling cryptanalysts to infer key-related information.

- **Differential cryptanalysis** [14] examines how specific differences between pairs of plaintexts propagate through a cipher. If certain differences in the cipher state before the last round occur with a probability higher than random chance, an attacker can use this information to deduce parts of the secret key. However, in practice, computing the exact probability across multiple encryption rounds is challenging. Instead, cryptanalysts focus on *differential trails*, which track how

differences evolve at each encryption round. By analyzing these trails, attackers can uncover non-random patterns in the cipher's behavior, making key recovery more feasible. This method has proven particularly effective against block ciphers.

- **Linear cryptanalysis** [56, 55] exploits biased linear relationships between plaintext, ciphertext, and key bits. If an attacker can discover Boolean linear equations that hold with probability significantly different from 1/2, they can gather information about the key through repeated observations. This attack becomes particularly effective when the cipher's structure introduces unintended statistical dependencies.

Algebraic Attacks. Algebraic attacks [9] exploit the fact that all ciphers can be expressed as a system of algebraic equations involving the plaintext, ciphertext, and key, often involving operations such as addition, multiplication, or Boolean logic.

The goal of an algebraic attack is to construct such a system of equations in a way that reduces the complexity of solving to recover the key—potentially below the bounds claimed by the cipher's security guarantees. The feasibility of this approach depends on the structure of the resulting equations: if they are sufficiently low-degree, sparse, or otherwise highly structured, solving the system may become practical—even when the number of variables is large. Ciphers with simple algebraic representations are particularly vulnerable, as their encryption process can be described using simple polynomial equations that may be solvable through algebraic techniques.

Among the most powerful tools for conducting algebraic cryptanalysis is the concept of a Gröbner basis [20], which allows for the systematic simplification of multivariate polynomial systems. Formally, given a set of polynomials over a finite field, a Gröbner basis is a canonical generating set of the ideal they generate, with respect to a chosen monomial ordering. Computing a Gröbner basis transforms the original problem into an equivalent one that is often easier to solve algorithmically—for instance, by reducing it to a triangular or even diagonal form. Gröbner basis methods (or even other methods such as SAT solvers [77]) can thus reveal structural weaknesses in the cipher if the algebraic expressions for its round functions or key schedule result in a tractable system, for example, if the cipher's internal operations can be described using low-degree polynomials, or if the number of equations exceeds the number of unknowns in a manageable way.

Given their analytical depth and increasing applicability, algebraic attacks play a central role in the work presented in this thesis. Throughout, we explore this topic in greater detail, focusing on the algebraic properties that make symmetric constructions either vulnerable or resilient to such attacks.

Structural Attacks. Structural attacks exploit inherent properties of a cipher that persist throughout encryption, making it distinguishable from a random permutation. Two prominent types are *Invariant Attacks* [52], which rely on subsets of inputs that preserve certain properties under encryption, and *Integral Attacks* [27], which study how carefully specific sets of plaintexts propagate collectively through multiple rounds of encryption. The latter are particularly relevant against SPNs, and represent the best known attacks against AES [81].

Other Cryptanalytic Techniques. Beyond these major categories, researchers have developed a range of cryptanalysis methods, including hybrid techniques that combine multiple attack strategies. Notable examples include **Boomerang and slide attacks** [82, 15], which extend differential techniques to break multiple encryption rounds; and **Meet-in-the-middle attacks** [29], which reduce key search complexity by computing forward from the plaintext and backward from the ciphertext, checking for a match in the middle.

Despite these threats, modern symmetric cryptographic algorithms are designed with rigorous security analysis to withstand known attack techniques. However, cryptanalysis remains an evolving field, constantly challenging the resilience of existing ciphers. The ongoing challenge for cryptographers is to anticipate and mitigate new attack strategies, ensuring the continued security of symmetric cryptographic systems.

Chapter 2

Beyond Traditional Symmetric Primitives: Symmetric Techniques for Advanced Protocols

The rapid evolution of cryptographic applications has driven the need for specialized symmetric primitives tailored to complex frameworks such as *Fully Homomorphic Encryption* (FHE), *Zero-Knowledge Proofs* (ZKPs), and *Multi-Party Computation* (MPC). While traditional symmetric primitives are well-suited for conventional encryption and decryption tasks, they often struggle to meet the computational constraints imposed by these advanced settings [4]. Many modern cryptographic protocols introduce new efficiency challenges that conventional symmetric primitives were not designed to address, demanding the development of alternative solutions optimized for these environments.

To address such growing demands, cryptographers have introduced a specialized class of symmetric primitives known as *Symmetric Techniques for Advanced Protocols* (STAP), explicitly designed for efficiency in frameworks such as FHE, ZKPs, and MPC [78]. The term STAP was first introduced at *STAP'23*, a workshop affiliated with *Eurocrypt'23*, and it broadly refers to symmetric cryptographic algorithms optimized for performance in arithmetic-intensive settings, particularly those operating over algebraic structures such as large prime fields $\mathbb{F}_p$ or the finite field $\mathbb{F}_{2^n}$. In these settings, conventional symmetric primitives often become inefficient or structurally incompatible, as they are typically designed to operate efficiently over binary circuits and bit-oriented data representations like $\mathbb{F}_2^n$—a natural fit for classical processors and software—and their bitwise-oriented design does not translate well to algebraic computations, rendering them impractical. STAP primitives, by contrast, are crafted to favor structures that align with the algebraic constraints of the protocols they support, each of which imposes distinct design

requirements that directly influence STAP construction.

The increasing adoption of STAPs reflects their growing significance in modern cryptographic research. This chapter explores the motivation behind STAPs, their key efficiency and security challenges, their role in advanced cryptographic protocols, and the key design challenges that set them apart from classical symmetric primitives.

2.1 Applications in Advanced Protocols

STAPs play a crucial role in cryptographic protocols that impose unique efficiency constraints on symmetric constructions. In the following, we examine each of these domains and the role of STAPs in them, analyzing how their tailored design principles address the efficiency and security challenges posed by these advanced cryptographic protocols.

2.1.1 Fully Homomorphic Encryption

Fully Homomorphic Encryption (FHE) is a cryptographic technique that enables computations to be performed directly on encrypted data. Given ciphertexts corresponding to plaintext values, an FHE scheme allows one to compute a function on the ciphertexts, resulting in a new ciphertext that, when decrypted, matches the outcome of performing the same function to the original plaintexts. In formal terms, for a function F and plaintexts $x_1, \ldots, x_n$, an FHE scheme satisfies:

$$\mathrm{Enc}(F(x_1, \ldots, x_n)) = F(\mathrm{Enc}(x_1), \ldots, \mathrm{Enc}(x_n)),$$

where the right-hand side represents the homomorphic evaluation of F over encrypted inputs. A crucial property of FHE is that it supports both homomorphic addition and multiplication on ciphertexts, and, since any computable function can be expressed as a composition of additions and multiplications (at the bit-level if necessary), this means that FHE schemes preserve the semantics of *any* computation, allowing computations to be carried out on encrypted data without requiring intermediate decryption or ever exposing the underlying information [5, 35].

One of the core challenges in homomorphic encryption is managing *noise*, a form of computational error that is deliberately introduced during encryption to ensure security. As homomorphic operations are applied to ciphertexts, the noise grows. Importantly, multiplications contribute significantly more to noise growth than additions. This leads to

the concept of *multiplicative depth*, which refers to the maximum number of sequential multiplications on any input-to-output path in a circuit. The higher the multiplicative depth, the greater the noise accumulation, which can quickly render a ciphertext undecryptable. To mitigate this, a technique called *bootstrapping* is used, which refreshes a ciphertext by encrypting it, and then homomorphically evaluating the decryption circuit itself—reducing the noise and restoring the ciphertext to a usable state.

To alleviate the cost of frequent bootstrapping and enable more efficient computation in practical settings, a class of schemes known as *approximate homomorphic encryption* has gained prominence. These schemes, most notably the CKKS scheme [23], allow approximate evaluation of real-number computations over encrypted data. Unlike traditional FHE schemes that operate on discrete bitstrings or modular integers, approximate FHE encodes real- or complex-valued vectors into ciphertexts and supports approximate arithmetic operations such as addition, multiplication, and scalar scaling. Instead of striving for exact correctness, approximate FHE tolerates a bounded level of numerical error in its outputs, which is acceptable in many applications such as signal processing and machine learning. However, designing cryptographic primitives for use within approximate FHE frameworks introduces new challenges, including the need to control numerical stability, ensure bounded error propagation, and maintain efficiency across large vectorized operations.

Applications. A key application of FHE is privacy-preserving computation, enabling secure data processing without exposing sensitive information. It is particularly valuable for verifiable computation, privacy-preserving machine learning (PPML), and electronic voting. One of the most significant use cases of FHE is securely outsourcing computations on sensitive data to a cloud service while maintaining confidentiality. However, directly encrypting entire datasets with FHE may be computationally prohibitive due to its high overhead, limiting large-scale applications.

To address the high computational overhead of FHE on client-side devices, a more practical alternative is *Hybrid Homomorphic Encryption* (HHE) [24, 3], which offloads the costly FHE computations to the cloud while preserving strong security guarantees. In this approach, the client first encrypts their data using a symmetric encryption scheme (e.g., a block cipher) with a key K_s, generating a ciphertext C_s. This is precisely where STAPs play a central role, as the symmetric scheme in this setting is typically instantiated by a STAP primitive specifically optimized for HHE. Then, instead of applying costly FHE operations directly to the entire dataset, the client encrypts only the symmetric key K_s using FHE, producing $Enc(K_s)$.

Both C_s and $Enc(K_s)$ are then sent to the cloud. Upon receiving the encrypted data, the cloud first homomorphically encrypts the symmetric ciphertext C_s, resulting in $Enc(C_s)$. Then, using the homomorphically encrypted key $Enc(K_s)$, the cloud performs a homomorphic decryption of the symmetric cipher, applied to $Enc(C_s)$. The effect is that the cloud obtains the underlying plaintexts, but now encrypted under FHE. Essentially, the symmetric encryption is removed, and the cloud ends up with plaintexts encrypted solely with FHE—simulating a scenario in which the client had encrypted the data directly using FHE. From this point, the cloud can perform arbitrary homomorphic computations on the data. After processing, the cloud returns the computed result—still encrypted—to the client, who can then decrypt it using their private key.

While HHE does not reduce the overall computational cost of FHE-based processing, it shifts the intensive workload from the client to the cloud, making it more suitable for resource-constrained clients. However, this comes at the cost of increased system complexity, as the cloud must now perform both homomorphic decryption of the symmetric cipher and the subsequent computation on encrypted data. The overall workflow of HHE is illustrated in Figure 2.1.

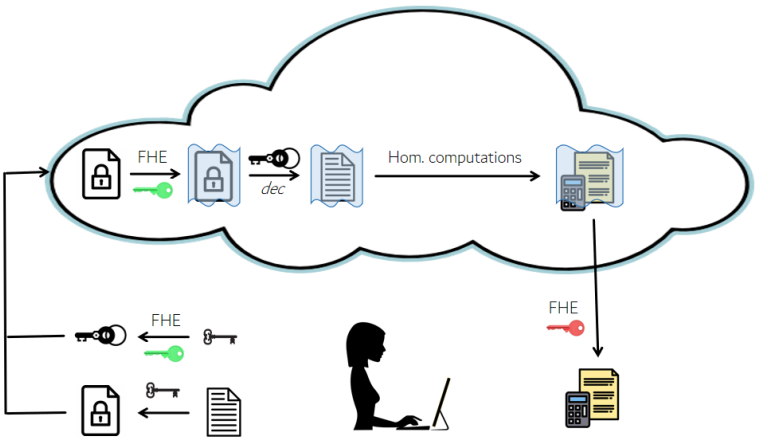


Figure 2.1: Hybrid Homomorphic Encryption (HHE) workflow. The client encrypts data symmetrically and transmits the encrypted data along with an FHE-encrypted key to the cloud. The cloud homomorphically encrypts the symmetric ciphertext, then homomorphically runs the symmetric decryption operation using the FHE-encrypted symmetric key. The result is the plaintext, now encrypted under FHE, which can then be processed homomorphically.

Efficiency Challenges. FHE schemes support arbitrary sequences of additions and multiplications over encrypted data, but their efficiency is severely constrained by the growth of noise introduced during multiplications. In contrast to addition, which con-

tributes very little to noise, multiplication causes significant noise amplification. Over time, this accumulation can render the ciphertext undecryptable unless costly noise-reduction procedures like bootstrapping are performed.

A key factor in this trade-off is the multiplicative depth, which reflects the worst-case noise growth and effectively limits the complexity of functions that can be computed before decryption fails. Because bootstrapping is expensive, minimizing multiplicative depth is a core design goal when building circuits for homomorphic evaluation. FHE-optimized symmetric primitives therefore aim to keep the multiplicative depth of the cipher low.

FHE-Optimized Symmetric Primitives. Several symmetric primitives have been specifically designed to align with FHE's efficiency constraints. These include LowMC [3], various Rasta variants (Dasta, Fasta, Masta, Pasta, and Rasta [49, 25, 46, 33, 31]), as well as FLIP [58], FiLIP [57], and Kreyvium [22]. These ciphers are tailored to minimize the use of non-linear operations while maintaining robust security properties within homomorphic frameworks.

Traditionally, FHE schemes operate over polynomial rings, but most FHE-friendly symmetric primitives have been designed over fields. However, two recently proposed symmetric schemes, Elisabeth [26] and Rubato [47], deviate from this convention by operating directly over rings. We will take a closer look at Rubato later in this thesis.

2.1.2 Zero-Knowledge Proofs

Originally introduced by Goldwasser, Micali, and Rackoff in 1985 [37], Zero-Knowledge Proofs (ZKPs) are cryptographic protocols that enable a *prover* to demonstrate the validity of a statement to a *verifier* without revealing any additional information beyond the truth of the statement itself. This fundamental property ensures that sensitive data remains private while still allowing for secure verification [36, Chapter 4]. Formally, ZKPs satisfy three key properties:

- **Completeness:** If the statement is true and the prover follows the protocol, the verifier will accept the proof with high probability.
- **Soundness:** A dishonest prover cannot convince the verifier of a false statement except with negligible probability.

- **Zero-Knowledge:** The verifier gains no additional information beyond the validity of the statement.

While originally defined in interactive settings where the verifier sends a challenge to the prover, most modern cryptographic applications rely on Non-Interactive Zero-Knowledge Proofs (NIZKPs). These eliminate the need for communication between prover and verifier by using a cryptographic hash function applied to the statement and possibly other public data to simulate the verifier's challenge. Since the hash function behaves like a random function, its output is unpredictable and effectively acts as a randomly chosen challenge. This convinces an offline verifier that the proof was generated honestly, as the prover could not have chosen the challenge in their favor. Additionally, because the challenge is derived deterministically from public data, the verifier can independently recompute it to confirm that the prover followed the protocol correctly. This makes NIZKPs especially well-suited for offline or decentralized verification scenarios, and they now form the backbone of many privacy-preserving technologies.

A visual comparison of interactive and non-interactive proofs is shown in Figure 2.2

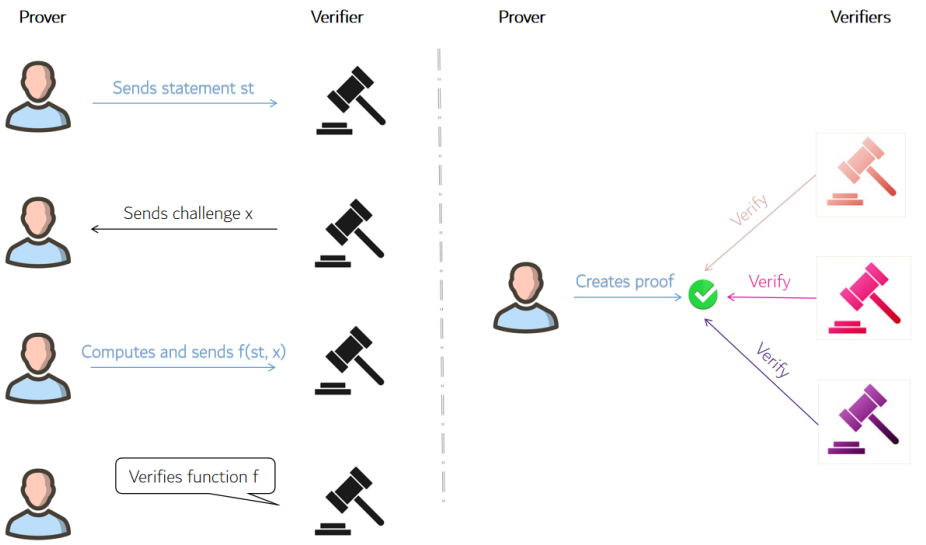


Figure 2.2: Comparison of Interactive Zero-Knowledge Proofs (left) and Non-Interactive Zero-Knowledge Proofs (right).

Applications. ZKPs are widely used in scenarios where authentication and verification are required without exposing underlying data. They play a crucial role in privacy-preserving technologies across various fields [83]. In electronic voting, they ensure the correctness of election results while keeping each individual's vote secret, maintaining

both transparency and voter privacy. Secure authentication systems leverage ZKPs to allow users to prove knowledge of a password or secret key without revealing it, reducing risks like password leaks and phishing attacks. Similarly, digital signatures benefit from ZKPs by enabling message authentication without exposing private keys, ensuring integrity and non-repudiation.

Another significant application is in privacy-preserving cryptocurrency transactions, where ZKPs validate transactions without revealing sender, recipient, or amount details, ensuring confidentiality in decentralized financial systems. Additionally, confidential identity verification uses ZKPs to prove attributes like age or citizenship without disclosing unnecessary personal data, enhancing privacy in online services and regulatory compliance.

A central building block in all of these applications is a hash function, which is often constructed via a sponge construction instantiated with an internal permutation—a STAP primitive—that repeatedly absorbs and squeezes data. Hash functions are commonly used in two ways: either to compress multiple inputs or messages into a short digest—often modeled as the output of a random oracle—or to serve as the compression function within cryptographic accumulators like Merkle trees, enabling efficient membership proofs. In most cases, the prover must demonstrate knowledge of a preimage x of a given hash output $y = H(x)$, without revealing x. Instead of recomputing the full hash function H inside the proof—which can be prohibitively expensive—ZK systems often verify an equivalent relation $F(x, y) = 0$ that is satisfied if and only if $y = H(x)$, using a cheaper algebraic representation. Since the cost of generating and verifying the proof depends heavily on the structure of this relation, the efficiency of the entire protocol is directly influenced by the hash function and thus the design of the underlying permutation.

Efficiency Challenges. ZKPs are typically constructed by translating the computational statement to be proved into an arithmetic circuit composed of addition and multiplication operations over a finite field. The computational cost of a ZKP depends heavily on the structure of this circuit, primarily its *multiplicative complexity* (MC)—the number of multiplication operations it contains. Additions are generally inexpensive, so minimizing the number of multiplications is crucial for improving prover efficiency, proof size, and overall system performance [4].

ZK-friendly symmetric encryption schemes and hash functions thus aim to optimize multiplicative complexity while maintaining security. Such constructions typically operate over large finite fields $\mathbb{F}_q$, where q is either a large prime or a power of 2 greater than or

equal to 2^{128}, and rely on low-degree polynomial representations for performance optimization.

ZK-Friendly Symmetric Primitives. To accommodate the computational constraints of ZKPs, specialized symmetric primitives have been developed. These fall into three main categories [67]:

1. **Low-Degree Primitives:** These primitives limit the number of non-linear operations using low-degree polynomial functions in their round functions, improving efficiency in proof generation and verification. Examples include Poseidon variants (HadesMiMC [42], Neptune [44], Poseidon [40], and Poseidon2 [41]) and MiMC [2].
2. **Equivalence Relation-Based Primitives:** These designs enable high-degree evaluation while maintaining low-degree verification, balancing expressiveness and efficiency. Notable examples include the Marvellous family (Rescue variants [4, 79, 7], Vision variants [4, 8], and XHash [6]), Anemoi [18], Arion [70] and Griffin [38].
3. **Look-Up Table Primitives:** A more recent approach, these primitives leverage precomputed look-up tables to optimize performance. However, this method can reduce compatibility across different proving systems. Examples include Monolith [39] and Tip5 [80].

As research progresses, new cryptographic protocols continue to emerge. In line with this, part of this thesis is devoted to the cryptanalysis of the Griffin, Arion, and Anemoi permutations.

2.1.3 Multi-Party Computation

Multi-Party Computation (MPC) is a cryptographic protocol that enables multiple parties to jointly compute a function over their private inputs without revealing any information beyond the final output [34, Chapter 2]. This ensures that no individual participant gains access to another party's private data while still collaboratively executing computations. Formally, an MPC protocol allows parties $P_1, P_2, \ldots, P_n$ with private inputs $x_1, x_2, \ldots, x_n$ to jointly compute $f(x_1, x_2, \ldots, x_n)$ while preserving input confidentiality. Notably, the output of the computation may not be the same for all participants—each party may receive an individual result y_i, such that $(y_1, y_2, \ldots, y_n) = f(x_1, x_2, \ldots, x_n)$. This allows for more flexible computations where outputs are personalized or distributed among parties (see Figure 2.3).

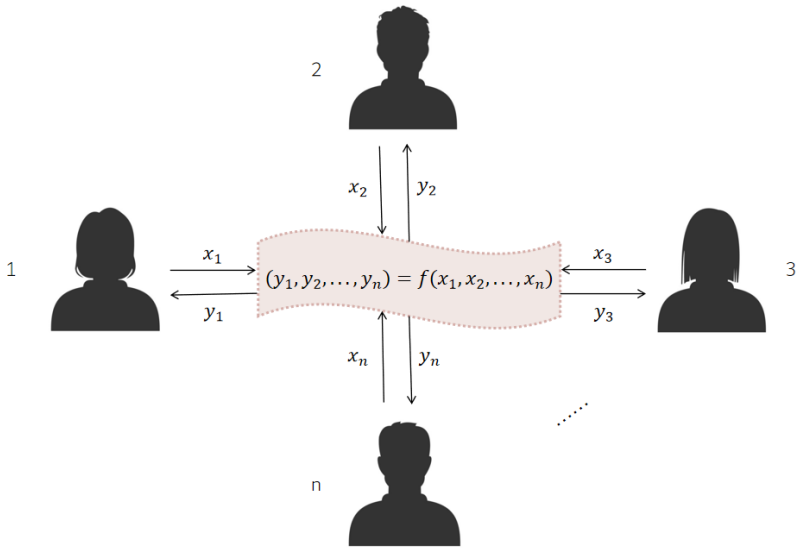


Figure 2.3: Multi-Party Computation: each party holds a private input and collaboratively computes a function without revealing their input to others. Communication occurs among parties to securely evaluate the function and reveal only the output.

MPC protocols are designed to tolerate adversarial behavior by ensuring that malicious parties cannot significantly disrupt or bias the computation. Different adversarial models capture the types of corruption a protocol can withstand:

- **Semi-Honest (Passive) Adversaries:** Corrupted parties follow the protocol correctly but try to learn extra information from the messages they receive. These protocols are often more efficient but provide weaker security guarantees.
- **Malicious (Active) Adversaries:** Corrupted parties may deviate arbitrarily from the protocol to gain information or sabotage the computation. Defenses involve cryptographic techniques like ZKPs or consistency checks.
- **Honest Majority:** A majority of the parties are assumed to be honest [34, Chapter 7]. This allows for simpler and more efficient protocols, often tolerating up to $\lfloor (n-1)/2 \rfloor$ corruptions.
- **Dishonest Majority:** More than half of the parties may be corrupted. Security in this model is harder to achieve and typically relies on stronger cryptographic assumptions, resulting in higher overhead.

The choice of adversarial model significantly affects the efficiency, communication complexity, and security guarantees of an MPC protocol, and must align with the threat

model of the application at hand.

Applications. MPC has broad applications in privacy-preserving computation scenarios where multiple stakeholders must collaborate without compromising confidentiality [34, Chapter 1]. Key use cases include electronic voting, where votes are tallied without exposing individual choices; auctions with private bids, ensuring the computation of winning bids while maintaining bidder privacy; secure storage mechanisms where multiple entities jointly manage sensitive data without unilateral access; and server-side authentication methods such as one-time password schemes and key generation, without exposing user secrets. Additionally, MPC is integral to Oblivious Pseudorandom Functions (OPRFs), which enable secure computations in various cryptographic protocols while hiding input-output correlations [34, Chapter 3].

Efficiency Challenges. The primary bottleneck in MPC stems from the cost of performing multiplicative operations, which require communication between parties [4]. In contrast, additions are typically free or incur negligible overhead, as they can be executed locally. As a result, the efficiency of an MPC protocol is tightly linked to the multiplicative complexity of the underlying computation.

MPC protocols can be broadly categorized based on how they handle communication and round complexity. In this context, a round refers to a complete phase of communication between all parties, typically consisting of one message from each party. Some protocols aim for a constant number of rounds, where the total communication grows linearly with the MC. Others prioritize reducing total communication bandwidth but incur more rounds, with the number of rounds proportional to the multiplicative depth of the circuit f. In both models, minimizing MC is critical to reducing interaction, lowering latency, and improving throughput.

This efficiency concern is particularly relevant when deploying symmetric cryptographic primitives in MPC. Rather than appearing as stand-alone operations, these primitives typically function as internal subroutines that enable the higher-level goal of securely computing a joint function over private inputs. For example, block ciphers or PRFs may be used to expand short seeds into long pseudorandom tapes shared between parties, encrypt intermediate values before redistribution in multi-round computations, or implement oblivious primitives such as OPRFs. In these contexts, every non-linear gate in the primitive's circuit contributes to the overall communication and computation cost of the MPC protocol. MPC-friendly primitives are therefore designed with minimal non-linear operations and, ideally, single-round evaluation—the theoretical minimum—thereby enabling practical, high-performance secure computation without weakening security.

MPC-Optimized Symmetric Primitives. Building on this motivation, dedicated symmetric cryptographic primitives have been developed that reduce the reliance on multiplicative operations while preserving cryptographic strength. One such primitive type is Correlated Pseudorandom Functions (cPRFs), which output values satisfying a predefined correlation pattern yet remain indistinguishable from random to an adversary [19]. cPRFs assume inputs are uniformly random rather than adversarially chosen, and they are especially useful in generating correlated randomness during MPC setup.

Symmetric primitives specifically developed to align with MPC constraints include the block ciphers Ciminion [32] and MiMC [2], and PRFs such as the Dark Matter PRF [17] and Hydra [45].

As MPC gains traction in real-world applications, the need for optimized cryptographic primitives and efficient protocol designs continues to grow. This thesis contributes to this line of work by analyzing one such cryptographic construction, particularly a weak PRF successor of the Dark Matter PRF [1].

2.2 Reimagining Symmetric Design: From Classical to STAP

Symmetric cryptographic primitives have traditionally been optimized for general-purpose computing environments, prioritizing efficiency in terms of CPU cycles, memory usage, and compatibility with standard hardware architectures. These designs are shaped by well-understood threat models and implementation contexts, favoring structures that are both fast and resistant to known attacks.

In contrast, STAPs are designed with very different constraints and goals [4], shaped by the needs of protocols such as FHE, ZKP, and MPC. These differences span across both the underlying algebraic structures and the performance metrics that drive design decisions, reflecting the specialized requirements of the protocols they support.

2.2.1 Algebraic Foundations and Mathematical Structures

The algebraic structures underpinning symmetric designs play a critical role in protocol compatibility. As previously discussed, classical symmetric primitives are universally defined over $\mathbb{F}_2$, which is inherently compatible with digital hardware, where field operations like addition and multiplication translate directly into efficient bitwise operations

(e.g., XOR and AND).

STAPs, however, often deviate from this norm. While some of them are defined over binary extension fields $\mathbb{F}_{2^n}$, most STAPs are defined over large prime fields $\mathbb{F}_p$, where the choice of p is dictated by the arithmetic of the host protocol. This alignment allows the symmetric primitive to be natively evaluated within the same domain as the protocol itself, avoiding costly and error-prone conversions between algebraic structures. Some designs even forgo field-based frameworks entirely, opting instead for modular rings $\mathbb{Z}_q$ or structured polynomial rings, thereby expanding the design space—but also introducing new challenges in terms of security analysis.

2.2.2 Cost Metrics and Security Trade-offs

In classical symmetric designs, performance is measured primarily in terms of bit-level operations: clock cycles, memory footprint, and parallelizability on standard CPUs and embedded devices. Accordingly, designers emphasize strong diffusion and non-linearity, and choose components based on their resistance to standard attack vectors, including differential, linear, and algebraic cryptanalysis.

As seen in previous sections, protocols like FHE, ZKP, and MPC shift the performance bottleneck from bit-level operations to arithmetic ones—especially multiplications. As a result, STAP designers aim to minimize both multiplicative complexity and multiplicative depth, often by favoring linear operations or structured constructions that align with protocol requirements. This leads to a fundamentally different design philosophy. In these settings, precomputed non-linear components like classical S-boxes are inefficient, so non-linearities must be computed directly and efficiently. As a result, STAPs often rely on algebraic functions such as low-degree monomials or sparse polynomials, which are both computationally cheap and compatible with the structure of the host protocol.

However, this algebraic simplicity comes at a cost. While these constructions facilitate protocol performance, they also narrow the range of available defenses, potentially weakening resistance to certain types of attacks—particularly algebraic cryptanalysis. A common countermeasure is increasing the number of rounds in the primitive's design, but this alone does not always suffice, as it also raises the multiplicative complexity, potentially undermining efficiency. This demonstrates the delicate trade-off between efficiency and security in STAP design.

2.2.3 Implications

The contrasting design philosophies of classical symmetric primitives and STAPs illustrate a fundamental shift in the goals and assumptions of modern symmetric cryptography. Rather than targeting standalone deployment in hardware, embedded systems, or general-purpose software, STAPs are tailored components embedded within larger arithmetic-heavy protocols. This specialization shapes not only how they are implemented, but also how their security should be analyzed and argued for.

Critically, the algebraic simplicity that enables efficient protocol integration also opens the door to new forms of cryptanalysis. While binary fields are well-understood within existing cryptanalytic frameworks, there is neither a clear consensus nor a solid theoretical foundation for evaluating the resistance of STAPs against algebraic attacks. Security arguments in this context are not yet well understood. This knowledge gap highlights the need for new methods to evaluate the algebraic robustness of primitives deliberately designed to be simple.

Against this backdrop, the objective of this thesis is twofold: first, to investigate the algebraic vulnerability of a few emerging STAP constructions, using both classical and novel algebraic attack techniques; and second, to explore how one might develop sound and transparent arguments for their security. In doing so, this thesis aims to contribute to the foundations of algebraic cryptanalysis in the context of domain-specific symmetric primitives.

Chapter 3

Paper Overview

This chapter provides an overview of the included papers and summarizes their main contributions.

3.1 Paper I

Context and Motivation

As discussed in the previous chapter, STAP primitives are compatible with a broader range of algebraic structures, and of particular interest are primitives defined over rings $\mathbb{Z}_q$ for composite q. Rubato [47], introduced at Eurocrypt 2022, embodies this new trend. It combines low-degree polynomial operations over $\mathbb{Z}_q$ with Gaussian noise to obscure algebraic structure and resist known attacks. This design enables compatibility with approximate FHE, particularly in the RtF (Real-to-Finite-field) transciphering framework built upon CKKS. However, its low algebraic degree raises concerns about susceptibility to algebraic attacks, especially since the noise addition may not suffice to fully mask underlying weaknesses.

Figure 3.1 illustrates both the internal structure of Rubato's round function (top) and its overall encryption process (bottom), highlighting how linear and non-linear components interact before the final noise addition.

In our work [43], we investigate how classical algebraic cryptanalysis techniques—originally developed for field-based primitives—can be adapted to the ring setting, and we identify structural properties unique to rings that can be exploited to lower the complexity of some attacks. This analysis reveals how certain design choices, when translated from

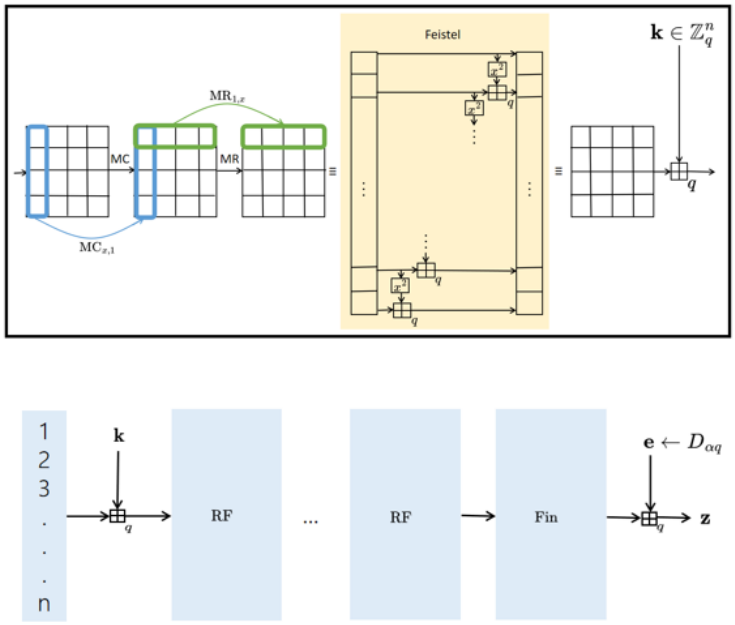


Figure 3.1: Overview of Rubato's structure. **Top:** The round function, consisting of linear and low-degree non-linear transformations. **Bottom:** The full encryption process, showing initialization, multiple round applications, and final noise injection.

fields to rings, may introduce subtle vulnerabilities that are not immediately apparent.

Main Contribution: Key Recovery Attack on Rubato

We develop and execute a full key recovery attack on five of the six proposed Rubato variants, specifically when q admits favorable factorizations. The attack leverages the fact that Rubato's state transformations are low-degree polynomial maps over $\mathbb{Z}_q$, and that the added noise is not sufficient to cover up this weakness. In particular, when q includes small prime factors, we exploit this structure to mount efficient modular brute-force attacks. The attack proceeds as follows:

1. **Partial key recovery modulo** m: We use small prime factors of q to recover the secret key modulo m, using a brute-force strategy guided by distinguishing properties of the reduced noise distribution.

2. **Noise-free index identification**: Using the partial key and properties of the Gaussian noise, we identify positions in the key stream where no noise was added.

Table 3.1: Lowest time complexities of key recovery attack, where q has particular factors.

Rubato variant	Assumption on q	Time	Data	Memory
Rubato-80S	$44\|q$	$2^{55.35}$	$2^{15.71}$	$2^{24.48}$
Rubato-80M	$12\|q$	$2^{57.06}$	$2^{17.91}$	$2^{32.96}$
Rubato-80L	$4\|q$	2^{65}	$2^{20.31}$	$2^{39.27}$
Rubato-128S	$q = 11 \cdot 2^{22}$	$2^{55.35}$	$2^{44.43}$	$2^{44.43}$
Rubato-128M	$20\|q$	$2^{83.59}$	$2^{29.44}$	$2^{39.27}$

Table 3.2: Percentage of choices of q vulnerable to the attack for the various Rubato variants.

Rubato variant	Fraction of vulnerable q's
Rubato-80S	42.05%
Rubato-80M	25%
Rubato-80L	25%
Rubato-128S	58.47%
Rubato-128M	37.25%
Rubato-128L	0%

3. **Full key recovery**: Using this noise-free subset, we recover the key by solving a linearized system of polynomial equations over $\mathbb{Z}_q$.

The overall complexity of this attack is significantly below the claimed security levels for most variants, which claim 80- or 128-bit security. Table 3.1 provides a detailed breakdown of the complexity for each variant under our attack.

We also quantify the proportion of weak q values: for five of the six Rubato variants, at least 25% of valid q choices result in ciphers vulnerable to our attack. Table 3.2 summarizes the vulnerability thresholds across different parameterizations.

Implications and Recommendations

This work emphasizes the importance of adapting cryptanalytic methods to the ring setting and calls for more careful analysis when designing symmetric primitives over $\mathbb{Z}_q$, as subtle design choices (like the factorization of q) can open up substantial vulnerabilities that undermine traditional security arguments.

We also highlight concrete recommendations to mitigate the attacks on Rubato:

- Restricting q to be a prime number.
- Increasing the number of rounds or the width of the noise distribution.
- Considering the use of non-polynomial S-box constructions to disrupt polynomial representations altogether.

In response to our results, the Rubato designers have revised their specification to require that q be a prime number, addressing one of the key vulnerabilities we identified.

3.2 Paper II

Context and Motivation

Prominent examples of recently proposed STAP designs specifically tailored for ZK settings include the hash functions Griffin [38], which is built around the Griffin-π permutation; Anemoi [18], a family of permutations; and ArionHash [70], which uses the Arion permutation as its core component.

Each of these primitives follows a similar high-level structure: the round function is composed of a non-linear layer and a linear layer that is typically realized via multiplication by an MDS matrix and addition of a round constant. A key design choice is the use of a single branch with a very high-degree polynomial (notably via inversion of low-degree monomials in a finite field), combined with low-degree polynomials for the remaining branches. This strategy aims to achieve high algebraic degree rapidly across rounds, therefore reducing the necessary number of rounds. Figure 3.2 illustrates the round functions of Griffin-π, Anemoi, and Arion, highlighting their common structural patterns exploited in our analysis.

The security of these designs fundamentally relies on the hardness of the *Constrained-Input Constrained-Output* (CICO) problem [13], or more precisely, its following variant where it is assumed that $\mathbb{F}$ is a very large finite field:

> Let $F : \mathbb{F}^t \to \mathbb{F}^t$ be a permutation. The CICO problem consists of finding $x \in \{0\} \times \mathbb{F}^{t-1}$ such that $F(x) \in \{0\} \times \mathbb{F}^{t-1}$.

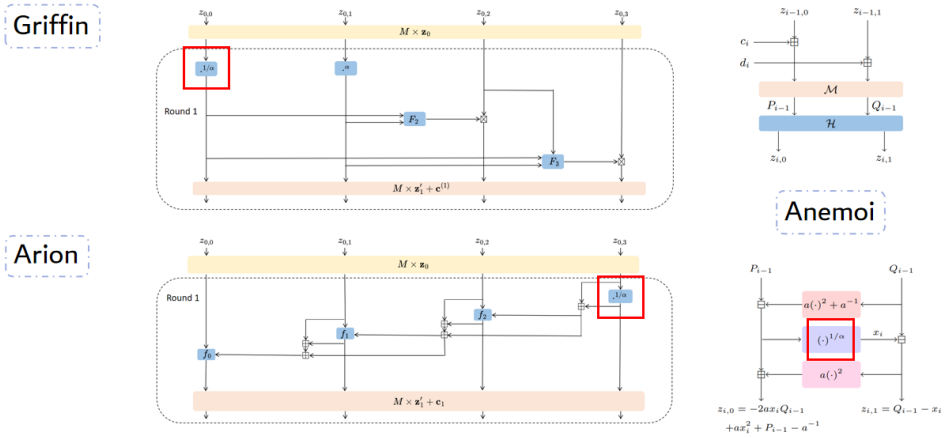


Figure 3.2: Round function structures for Griffin-π, Anemoi, and Arion primitives. The red square indicates the only high-degree component in each design.

An adversary capable of solving the CICO problem effectively gains partial control over both the input and output of the permutation—something a secure permutation is specifically designed to prevent.

Main Contribution: The FreeLunch Attack

In our work [10], we introduce the *FreeLunch* attack—a novel cryptanalytic technique that builds upon traditional Gröbner basis methods while significantly reducing their computational overhead. By strategically bypassing some of the most computationally expensive steps, FreeLunch offers a more efficient pipeline for attacking certain STAP primitives. The attack workflow consists of:

1. **Modeling:** We model the CICO problem as a system of polynomial equations by choosing suitable state variables and defining constraints corresponding to the round function structure.

2. **Bypassing Gröbner Basis Computation:** Traditional attacks require computing a Gröbner basis of the system, which is highly expensive. Instead, we generate the polynomial system using a monomial order where it already forms a Gröbner basis, eliminating this costly step. This "free" Gröbner basis is precisely what gives the attack its name, FreeLunch.

3. **Efficient Change of Order:** Classical attacks perform a change of order to obtain a Gröbner basis in lex order and extract a univariate polynomial. In our case, we

develop a dedicated reordering algorithm that exploits the system's sparsity and block structure to perform this change much more efficiently than for a general system.

4. **Root Finding:** Finally, we compute the roots of the resulting univariate polynomial via standard techniques.

The crucial innovation lies in steps 2 and 3: by custom-tailoring the monomial ordering and equation generation, we get the Gröbner basis for free and greatly accelerate step 3 compared to traditional approaches.

We successfully demonstrate the FreeLunch attack against full-round instances of Griffin-π, Anemoi, and Arion, all originally claiming 128-bit security. Our theoretical complexity estimates reveal drastic security degradation: attack complexities can drop as low as 2^{64} operations for the weakest instances—shaving off tens of bits compared to the intended security target.

Table 3.3 summarizes the complexity estimates for several attacked instances, highlighting the ones for which the claimed 128-bit security no longer holds.

Name	Number of branches				Name	Number of branches					Name	
	3	4	8	≥12		3	4	5	6	8		2
Griffin ($\alpha = 3$)	120	112	76	64	**Arion** ($e = 3$)	128	134	114	119	98	**Anemoi**-3	118
Griffin ($\alpha = 5$)	141	110	81	74	α-**Arion** ($e = 3$)	104	84	88	92	98	**Anemoi**-5	156

Table 3.3: Bit complexity for three out of four steps of the FreeLunch attack for variants of Griffin-π, Arion and Anemoi with claimed 128-bit security. Red text marks broken instances.

Implications and Recommendations

The FreeLunch attack exposes previously underestimated vulnerabilities in STAP designs, urging caution in future primitive development. Specifically, our results highlight that:

- Future designs should aim to avoid FreeLunch-friendly structures, possibly by incorporating more high-degree components.
- Classical security arguments based solely on round counts, degree growth, or generic Gröbner basis hardness assumptions are insufficient in STAP contexts. They should also account for specific system structures and monomial orders.

3.3 Paper III

Context and Motivation

Recent years have witnessed increasing demand for symmetric primitives tailored to secure MPC, particularly in post-quantum settings. One promising line of research involves the use of wPRFs constructed under the *alternating moduli paradigm*.

A recent contribution in this space is the work of Alamati et al. [1], which proposes new wPRFs optimized for MPC. Building on the foundational ideas introduced by Boneh et al. [17], their constructions alternate linear operations over two finite fields, typically $\mathbb{F}_2$ and $\mathbb{F}_3$. Their Standard ($\mathbb{F}_2$, $\mathbb{F}_3$)-wPRF is defined as:

$$F(k, x) := \mathbf{B} \cdot_3 (\mathbf{A} \cdot_2 [k \odot_2 x]) ,$$

where $k, x \in \mathbb{F}_2^n$ are the key and input vectors, $\mathbf{A}$ is a random matrix over $\mathbb{F}_2$, and $\mathbf{B}$ is a compressing matrix over $\mathbb{F}_3$. The notation $\odot_2$ denotes component-wise multiplication over $\mathbb{F}_2$. Figure 3.3 illustrates the structure of this construction. A variant called the Reversed Moduli wPRF reverses the order of moduli, using component-wise multiplication over $\mathbb{F}_3$ followed by compression over $\mathbb{F}_2$.

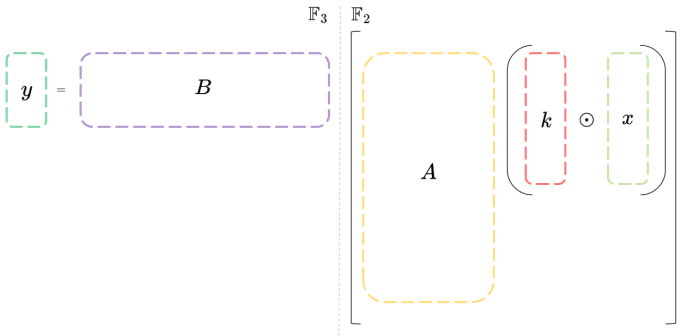


Figure 3.3: Construction of the Standard ($\mathbb{F}_2$, $\mathbb{F}_3$)-wPRF as proposed by Alamati et al.

They propose two classes of parameter sets: *One-to-One*, where the input and output spaces are of comparable size and intended to yield bijective behavior under random keys; and *Many-to-One*, which allow for input compression and naturally exhibit output collisions.

While these designs are promising, they remain relatively new and have not yet undergone extensive cryptanalysis. In this work [54], we critically examine their One-to-One parameter sets and find that certain algebraic properties of the construc-

tion—particularly the effect of zero entries in the key—introduce structural weaknesses exploitable by attackers.

Main Contribution: Cryptanalysis of One-to-One Alternating-Moduli wPRFs

We present a key recovery attack, and a modification thereof, that break the claimed λ-bit security One-to-One alternating-moduli wPRFs introduced by Alamati et al. Our attacks exploit the fact that the pointwise multiplication used in the constructions allows zero-entries in the key to *zero out* corresponding input positions. This interaction results in input–output mappings that are substantially more compressing than intended.

1. **Standard One-to-One wPRF:** We develop a collision-based key recovery attack that exploits the observation that when $k_i = 0$, the corresponding input bit x_i has no effect on the output. By collecting collision pairs—distinct inputs that produce the same output—we iteratively identify zero positions in the key. After each collision, we learn more about the key, progressively narrowing down the candidate space. Once the number of undetermined positions is sufficiently small, we switch to an exhaustive search over keys within a low Hamming distance of our current guess. The attack completes with complexity $\mathcal{O}(2^{\lambda/2} \log \lambda)$, far below the claimed 2^λ security level.

2. **Reversed Moduli One-to-One wPRF:** In this variant, the key values belong to $\mathbb{F}_3$, increasing the search space. However, our adapted attack still recovers all positions where $k_i = 0$ using collisions. Once the zero positions are fixed, we perform an exhaustive search over the remaining positions, each of which can take one of two values. The final complexity of the attack is $\mathcal{O}(2^{0.84\lambda})$, again falling short of the claimed security level.

The attacks are supported by theoretical analysis and validated through experiments, which confirms alignment with predicted complexities and reveals that the number of collisions required for successful key recovery is consistently small.

Implications and Recommendations

Our work demonstrates that the One-to-One parameter sets in the alternating-moduli wPRFs are more fragile than anticipated, as the zero entries in the key reduce input en-

tropy, thereby collapsing the image size and making the function vulnerable to collision-based attacks.

To mitigate these vulnerabilities, we recommend:

- Restricting key entries to non-zero values, i.e., $k \in \mathbb{F}_p^*$ (which naturally requires $p > 2$), to avoid complete zeroing of input components.
- Replacing pointwise multiplication with operations like addition that retain contribution from all input coordinates regardless of key values.

We note that our attacks do not apply to the Many-to-One parameter sets, where frequent output collisions are inherent and not informative about the key. Nonetheless, further cryptanalysis of these parameter sets is warranted. Overall, our results emphasize the need for careful evaluation when structural properties interact with algebraic operations in non-trivial ways.

Chapter 4

Conclusion

This thesis explores the algebraic security of symmetric primitives tailored for advanced cryptographic protocols such as FHE, MPC, and ZK—so-called STAPs. These primitives often rely on algebraic simplicity to enable efficient integration, exposing them to novel forms of algebraic cryptanalysis that are not well addressed by traditional frameworks. Through three case studies, we examine how different structural features can interact subtly with attack techniques and reveal unanticipated vulnerabilities.

The first paper [43] shows that security assumptions do not seamlessly transfer from field-based to ring-based settings. Specifically, the factorization of the modulus can create structural weaknesses that are easily overlooked. The second paper [10] introduces the FreeLunch attack, revealing that many STAPs are susceptible to system-specific vulnerabilities that elude classical degree-based arguments. Finally, the third paper [54] demonstrates how small oversights can dramatically reduce entropy and enable effective collision-based attacks. Together, these examples show that minor design decisions can introduce significant vulnerabilities not captured by standard security arguments.

Our results highlight the importance of context-specific analysis and structural awareness, and emphasize that algebraic cryptanalysis must evolve alongside cryptographic design, particularly in emerging application domains. By contributing new attack techniques and exposing concrete weaknesses, this thesis takes a step toward a more rigorous and transparent foundation for evaluating the algebraic robustness of STAPs.

Looking ahead, this work suggests that our understanding of STAP security is still evolving. While significant progress has been made in identifying and analyzing such vulnerabilities, a comprehensive theory of algebraic resistance and design strategies remains elusive. As STAPs become increasingly adopted in real-world applications, ensuring their security will require deeper theoretical insights through continued cryptanalysis.

Bibliography

[1] N. Alamati, G.-V. Policharla, S. Raghuraman, and P. Rindal. Improved Alternating-Moduli PRFs and Post-quantum Signatures. In L. Reyzin and D. Stebila, editors, *Advances in Cryptology – CRYPTO 2024*, volume 14927 of *Lecture Notes in Computer Science*, pages 274–308, Cham, 2024.

[2] M. Albrecht, L. Grassi, C. Rechberger, A. Roy, and T. Tiessen. MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity. In J. H. Cheon and T. Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016*, volume 10031 of *Lecture Notes in Computer Science*, pages 191–219, Berlin, Heidelberg, 2016.

[3] M. R. Albrecht, C. Rechberger, T. Schneider, T. Tiessen, and M. Zohner. Ciphers for MPC and FHE. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, volume 9056 of *Lecture Notes in Computer Science*, pages 430–454, Berlin, Heidelberg, 2015.

[4] A. Aly, T. Ashur, E. Ben-Sasson, S. Dhooghe, and A. Szepieniec. Design of Symmetric-Key Primitives for Advanced Cryptographic Protocols. *IACR Transactions on Symmetric Cryptology*, 2020(3):1–45, Sep. 2020.

[5] F. Armknecht, C. Boyd, C. Carr, K. Gjøsteen, A. Jäschke, C. A. Reuter, and M. Strand. A Guide to Fully Homomorphic Encryption. Cryptology ePrint Archive, Paper 2015/1192, 2015.

[6] T. Ashur, A. S. Bhati, A. Kindi, M. Mahzoun, and L. Perrin. XHash: Efficient STARK-friendly Hash Function. Cryptology ePrint Archive, Paper 2023/1045, 2023.

[7] T. Ashur, A. Kindi, W. Meier, A. Szepieniec, and B. Threadbare. Rescue-Prime Optimized. Cryptology ePrint Archive, Paper 2022/1577, 2022.

[8] T. Ashur, M. Mahzoun, J. Posen, and D. Šijačić. Vision Mark-32: ZK-Friendly Hash Function Over Binary Tower Fields. Cryptology ePrint Archive, Paper 2024/633, 2024.

[9] G. V. Bard. *Algebraic Cryptanalysis*. Springer, New York, 2009.

[10] A. Bariant, A. Boeuf, A. Lemoine, I. Manterola Ayala, M. Øygarden, L. Perrin, and H. Raddum. The Algebraic FreeLunch: Efficient Gröbner Basis Attacks Against Arithmetization-Oriented Primitives. In L. Reyzin and D. Stebila, editors, *Advances in Cryptology – CRYPTO 2024*, volume 14923 of *Lecture Notes in Computer Science*, pages 139–173, Cham, 2024.

[11] D. J. Bernstein. ChaCha, a variant of Salsa20. Technical report, University of Illinois at Chicago, 2008.

[12] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche. Sponge functions. In *ECRYPT Hash Workshop*, 2007.

[13] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche. Cryptographic sponge functions. `http://sponge.noekeon.org/CSF-0.1.pdf`, 2011.

[14] E. Biham and A. Shamir. Differential Cryptanalysis of DES-like Cryptosystems. *J. Cryptology*, 4:3–72, 1991.

[15] A. Biryukov and D. Wagner. Slide Attacks. In L. Knudsen, editor, *Fast Software Encryption*, volume 1636 of *Lecture Notes in Computer Science*, pages 245–259, Berlin, Heidelberg, 1999.

[16] Bitcoin Developers. Bitcoin protocol specification. `https://en.bitcoin.it/wiki/Protocol_Specification`, 2021.

[17] D. Boneh, Y. Ishai, A. Passelègue, A. Sahai, and D. J. Wu. Exploring Crypto Dark Matter: New Simple PRF Candidates and Their Applications. In A. Beimel and S. Dziembowski, editors, *Theory of Cryptography*, volume 11240 of *Lecture Notes in Computer Science*, pages 699–729, Cham, 2018.

[18] C. Bouvier, P. Briaud, P. Chaidos, L. Perrin, R. Salen, V. Velichkov, and D. Willems. New Design Techniques for Efficient Arithmetization-Oriented Hash Functions: Anemoi Permutations and Jive Compression Mode. In H. Handschuh and A. Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023*, volume 14083 of *Lecture Notes in Computer Science*, pages 507–539, Cham, 2023.

[19] E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, and P. Scholl. Correlated Pseudorandom Functions from Variable-Density LPN. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1069–1080, 2020.

[20] B. Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenrings nach einem nulldimensionalen Polynomideal [An Algorithm for Finding the*

Basis Elements of the Residue Class Ring of a Zero Dimensional Polynomial Ideal]. PhD thesis, University of Innsbruck, 1965. English translation by M. Abramson in *Journal of Symbolic Computation*, 41 (2006): 475–511.

[21] V. Buterin. Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. `https://ethereum.org/en/whitepaper/`, 2014.

[22] A. Canteaut, S. Carpov, C. Fontaine, T. Lepoint, M. Naya-Plasencia, P. Paillier, and R. Sirdey. Stream Ciphers: A Practical Solution for Efficient Homomorphic-Ciphertext Compression. volume 31 of *J Cryptol*, pages 885–916, 2018.

[23] J. H. Cheon, A. Kim, M. Kim, and Y. Song. Homomorphic Encryption for Arithmetic of Approximate Numbers. In T. Takagi and T. Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, volume 10624 of *Lecture Notes in Computer Science*, pages 409–437, Cham, 2017.

[24] J. H. Cheon and J. Kim. A Hybrid Scheme of Public-Key Encryption and Somewhat Homomorphic Encryption. *IEEE Transactions on Information Forensics and Security*, 10(5):1052–1063, 2015.

[25] C. Cid, J. P. Indrøy, and H. Raddum. FASTA – A Stream Cipher for Fast FHE Evaluation. In S. D. Galbraith, editor, *Topics in Cryptology – CT-RSA 2022*, volume 13161 of *Lecture Notes in Computer Science*, pages 451–483, Cham, 2022.

[26] O. Cosseron, C. Hoffmann, P. Méaux, and F.-X. Standaert. Towards Globally Optimized Hybrid Homomorphic Encryption - Featuring the Elisabeth Stream Cipher. In *ASIACRYPT 2022*, Taipei, Taiwan, 2022.

[27] J. Daemen, L. Knudsen, and V. Rijmen. The block cipher Square. In E. Biham, editor, *Fast Software Encryption*, volume 1267 of *Lecture Notes in Computer Science*, pages 149–165, Berlin, Heidelberg, 1997.

[28] I. B. Damgård. A Design Principle for Hash Functions. In *Advances in Cryptology — CRYPTO' 89 Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427, New York, NY, 1990.

[29] W. Diffie and M. Hellman. Exhaustive Cryptanalysis of the NBS Data Encryption Standard. In *Computer*, volume 10, pages 74–84, 1977.

[30] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[31] C. Dobraunig, M. Eichlseder, L. Grassi, V. Lallemand, G. Leander, E. List, F. Mendel, and C. Rechberger. Rasta: A Cipher with Low ANDdepth and Few

ANDs per Bit. In H. Shacham and A. Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018*, volume 10991 of *Lecture Notes in Computer Science*, pages 662–692, Cham, 2018.

[32] C. Dobraunig, L. Grassi, A. Guinet, and D. Kuijsters. Ciminion: Symmetric Encryption Based on Toffoli-Gates over Large Finite Fields. In A. Canteaut and F.-X. Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021*, volume 12697 of *Lecture Notes in Computer Science*, pages 3–34, Cham, 2021.

[33] C. Dobraunig, L. Grassi, L. Helminger, C. Rechberger, M. Schofnegger, and R. Walch. Pasta: A Case for Hybrid Homomorphic Encryption. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(3):30–73, Jun. 2023.

[34] D. Evans, V. Kolesnikov, and M. Rosulek. A Pragmatic Introduction to Secure Multi-Party Computation. *Found. Trends Priv. Secur.*, 2(2–3):70–246, Dec. 2018.

[35] C. Gentry. *A Fully Homomorphic Encryption Scheme*. PhD thesis, Stanford University, 2009. `https://crypto.stanford.edu/craig`.

[36] O. Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge University Press, 2001.

[37] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC '85, page 291–304, New York, NY, USA, 1985. Association for Computing Machinery.

[38] L. Grassi, Y. Hao, C. Rechberger, M. Schofnegger, R. Walch, and Q. Wang. Horst Meets Fluid-SPN: Griffin for Zero-Knowledge Applications. In H. Handschuh and A. Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023*, volume 14083 of *Lecture Notes in Computer Science*, pages 573–606, Cham, 2023.

[39] L. Grassi, D. Khovratovich, R. Lüftenegger, C. Rechberger, M. Schofnegger, and R. Walch. Monolith: Circuit-Friendly Hash Functions with New Nonlinear Layers for Fast and Constant-Time Implementations. Cryptology ePrint Archive, Paper 2023/1025, 2023.

[40] L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger. Poseidon: A New Hash Function for Zero-Knowledge Proof Systems. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 519–535. USENIX Association, Aug. 2021.

[41] L. Grassi, D. Khovratovich, and M. Schofnegger. Poseidon2: A Faster Version of the Poseidon Hash Function. In N. El Mrabet, L. De Feo, and S. Duquesne,

editors, *Progress in Cryptology - AFRICACRYPT 2023*, volume 14064 of *Lecture Notes in Computer Science*, pages 177–203, Cham, 2023.

[42] L. Grassi, R. Lüftenegger, C. Rechberger, D. Rotaru, and M. Schofnegger. On a Generalization of Substitution-Permutation Networks: The HADES Design Strategy. In A. Canteaut and Y. Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020*, volume 12106 of *Lecture Notes in Computer Science*, pages 674–704, Cham, 2020.

[43] L. Grassi, I. Manterola Ayala, M. N. Hovd, M. Øygarden, H. Raddum, and Q. Wang. Cryptanalysis of Symmetric Primitives over Rings and a Key Recovery Attack on Rubato. In H. Handschuh and A. Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023*, volume 14083 of *Lecture Notes in Computer Science*, pages 305–339, Cham, 2023.

[44] L. Grassi, S. Onofri, M. Pedicini, and L. Sozzi. Invertible Quadratic Non-Linear Layers for MPC-/FHE-/ZK-Friendly Schemes over Fnp: Application to Poseidon. *IACR Transactions on Symmetric Cryptology*, 2022(3):20–72, Sep. 2022.

[45] L. Grassi, M. Øygarden, M. Schofnegger, and R. Walch. From Farfalle to Megafono via Ciminion: The PRF Hydra for MPC Applications. In C. Hazay and M. Stam, editors, *Advances in Cryptology – EUROCRYPT 2023*, volume 14007 of *Lecture Notes in Computer Science*, pages 255–286, Cham, 2023.

[46] J. Ha, S. Kim, W. Choi, J. Lee, D. Moon, H. Yoon, and J. Cho. Masta: An HE-Friendly Cipher Using Modular Arithmetic. *IEEE Access*, 8:194741–194751, 2020.

[47] J. Ha, S. Kim, B. Lee, J. Lee, and M. Son. Rubato: Noisy Ciphers for Approximate Homomorphic Encryption. In O. Dunkelman and S. Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022*, volume 13275 of *Lecture Notes in Computer Science*, pages 581–610, Cham, 2022.

[48] D. Hankerson and A. Menezes. *Encyclopedia of Cryptography, Security and Privacy*. Springer US, Boston, MA, 2011.

[49] P. Hebborn and G. Leander. Dasta – Alternative Linear Layer for Rasta. *IACR Transactions on Symmetric Cryptology*, 2020, Issue 3:46–86, 2020.

[50] S. M. Hosseini and H. Pilaram. A Comprehensive Review of Post-Quantum Cryptography: Challenges and Advances. Cryptology ePrint Archive, Paper 2024/1940, 2024.

[51] D. Kahn. *The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet*. Scribner, New York, 1996.

[52] G. Leander, B. Minaud, and S. Rønjom. A Generic Approach to Invariant Subspace Attacks: Cryptanalysis of Robin, iSCREAM and Zorro. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, volume 9056 of *Lecture Notes in Computer Science*, pages 254–283, Berlin, Heidelberg, 2015.

[53] M. Luby and C. Rackoff. How to Construct Pseudo-random Permutations from Pseudo-random Functions. In *Advances in Cryptology — CRYPTO '85 Proceedings*, volume 218 of *Lecture Notes in Computer Science*, pages 447–447, Berlin, Heidelberg, 1986.

[54] I. Manterola Ayala and H. Raddum. Zeroed Out: Cryptanalysis of Weak PRFs in Alternating Moduli. *IACR Transactions on Symmetric Cryptology*, 2025(2):1–15, Jun. 2025.

[55] M. Matsui. Linear Cryptanalysis Method for DES Cipher. In T. Helleseth, editor, *Advances in Cryptology — EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397, Berlin, Heidelberg, 1994.

[56] M. Matsui and A. Yamagishi. A New Method for Known Plaintext Attack of FEAL Cipher. In R. A. Rueppel, editor, *Advances in Cryptology — EUROCRYPT' 92*, volume 658 of *Lecture Notes in Computer Science*, pages 81–91, Berlin, Heidelberg, 1993.

[57] P. Méaux, C. Carlet, A. Journault, and F.-X. Standaert. Improved Filter Permutators for Efficient FHE: Better Instances and Implementations. In F. Hao, S. Ruj, and S. Sen Gupta, editors, *Progress in Cryptology – INDOCRYPT 2019*, volume 11898 of *Lecture Notes in Computer Science*, pages 68–91, Cham, 2019.

[58] P. Méaux, A. Journault, F.-X. Standaert, and C. Carlet. Towards Stream Ciphers for Efficient FHE with Low-Noise Ciphertexts. In M. Fischlin and J.-S. Coron, editors, *Advances in Cryptology – EUROCRYPT 2016*, volume 9665 of *Lecture Notes in Computer Science*, pages 311–343, Berlin, Heidelberg, 2016.

[59] A. J. Menendez, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. Boca Raton: CRC Press, 1997.

[60] National Bureau of Standards. Data Encryption Standard (DES). Technical Report FIPS PUB 46, U.S. Department of Commerce, 1977.

[61] National Institute of Standards and Technology. Advanced Encryption Standard (AES). Technical Report FIPS PUB 197, U.S. Department of Commerce, 2001.

[62] National Institute of Standards and Technology. Post-Quantum Cryptography Standardization, 2017. Available at `https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization`.

[63] National Institute of Standards and Technology. Digital Signature Standard (DSS). Technical Report FIPS PUB 186-5, U.S. Department of Commerce, 2023.

[64] National Institute of Standards and Technology. FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard. `https://csrc.nist.gov/pubs/fips/203/final`, 2024. Standardization of CRYSTALS-Kyber.

[65] National Institute of Standards and Technology. FIPS 204: Module-Lattice-Based Digital Signature Algorithm Standard. `https://csrc.nist.gov/pubs/fips/204/final`, 2024. Standardization of CRYSTALS-Dilithium.

[66] National Institute of Standards and Technology (NIST) and M. J. Dworkin. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, 2015. Available at `https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=919061`.

[67] C. Rechberger. On the History of FHE/MPC/ZK-Friendly Symmetric Crypto. `https://who.paris.inria.fr/Leo.Perrin/rescale/slides/Christian-STAP-2023.pdf`, 2023. Talk at STAP (Symmetric Techniques for Advanced Protocols).

[68] E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, 2018.

[69] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Association for Computing Machinery*, 21(2):120–126, 1978.

[70] A. Roy, M. J. Steiner, and S. Trevisani. Arion: Arithmetization-Oriented Permutation and Hashing from Generalized Triangular Dynamical Systems. arXiv, 2303.04639, `https://arxiv.org/abs/2303.04639`, 2023.

[71] K. Seo and S. Kent. Security Architecture for the Internet Protocol. RFC 4301, Dec. 2005.

[72] P. W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[73] Signal Foundation. Signal Protocol Specifications. `https://signal.org/docs/`, 2013.

[74] S. Singh. *The Codebook: The Evolution of Secrecy from Mary, Queen of Scots to Quantum Cryptography*. Doubleday, New York, 2000.

[75] J. Sjödin. *Weak Pseudorandomness and Unpredictability*. PhD thesis, ETH Zurich, 2007. ETH Series in Information Security and Cryptography, vol. 8, Hartung-Gorre Verlag, ISBN 3-86628-088-2.

[76] N. P. Smart. *Cryptography: An Introduction*. 3rd edition, 2013. Available at `https://nigelsmart.github.io/Crypto_Book/`.

[77] M. Soos, K. Nohl, and C. Castelluccia. Extending SAT Solvers to Cryptographic Problems. In O. Kullmann, editor, *Theory and Applications of Satisfiability Testing - SAT 2009*, volume 5584 of *Lecture Notes in Computer Science*, pages 244–257, Berlin, Heidelberg, 2009.

[78] STAP Workshop Committee. STAP'23: Symmetric Techniques for Advanced Protocols. `https://who.paris.inria.fr/Leo.Perrin/rescale/stap-23.html`, 2023.

[79] A. Szepieniec, T. Ashur, and S. Dhooghe. Rescue-Prime: a Standard Specification (SoK). Cryptology ePrint Archive, Paper 2020/1143, 2020.

[80] A. Szepieniec, A. Lemmens, J. F. Sauer, B. Threadbare, and Al-Kindi. The Tip5 Hash Function for Recursive STARKs. Cryptology ePrint Archive, Paper 2023/107, 2023.

[81] Y. Todo and K. Aoki. FFT Key Recovery for Integral Attack. In D. Gritzalis, A. Kiayias, and I. Askoxylakis, editors, *Cryptology and Network Security*, volume 8813 of *Lecture Notes in Computer Science*, pages 64–81, Cham, 2014.

[82] D. Wagner. The Boomerang Attack. In L. Knudsen, editor, *Fast Software Encryption*, volume 1636 of *Lecture Notes in Computer Science*, pages 156–170, Berlin, Heidelberg, 1999.

[83] J. Wishwasara. Zero Knowledge Proofs: A Comprehensive Review of Applications, Protocols, and Future Directions in Cybersecurity, 08 2023.

[84] T. Ylonen and C. Lonvick. The Secure Shell (SSH) Protocol Architecture. RFC 4251, 2006.

Chapter 5

Scientific Results

Article I

5.1 Cryptanalysis of Symmetric Primitives over Rings and a Key Recovery Attack on Rubato

Lorenzo Grassi, Irati Manterola Ayala, Martha Norberg Hovd, Morten Øygarden, Håvard Raddum, and Qingju Wang

Published in *Advances in Cryptology – CRYPTO 2023, Lecture Notes in Computer Science (LNCS)*, vol 14083, pages 305–339. Presented at *CRYPTO* in August 2023 by the candidate.
https://doi.org/10.1007/978-3-031-38548-3_11

Version reproduced here in Cryptology ePrint Archive, Report 2023/822, 2023, with minor corrections.
https://eprint.iacr.org/2023/822

Cryptanalysis of Symmetric Primitives over Rings and a Key Recovery Attack on Rubato *

Lorenzo Grassi[1], Irati Manterola Ayala[2], Martha Norberg Hovd[2], Morten Øygarden[2], Håvard Raddum[2], and Qingju Wang[3]

[1] Ruhr University Bochum, Bochum, Germany
[2] Simula UiB, Bergen, Norway
[3] Telecom Paris, Institut Polytechnique de Paris, France
lorenzo.grassi@ruhr-uni-bochum.de
{irati,martha,haavardr,morten.oygarden}@simula.no
qingju.wang@telecom-paris.fr

Abstract. Symmetric primitives are a cornerstone of cryptography, and have traditionally been defined over fields, where cryptanalysis is now well understood. However, a few symmetric primitives defined over *rings* $\mathbb{Z}_q$ for a composite number q have recently been proposed, a setting where security is much less studied. In this paper we focus on studying established algebraic attacks typically defined over fields and the extent of their applicability to symmetric primitives defined over the ring of integers modulo a composite q. Based on our analysis, we present an attack on full Rubato, a family of symmetric ciphers proposed by Ha et al. at Eurocrypt 2022 designed to be used in a transciphering framework for approximate fully homomorphic encryption. We show that at least 25% of the possible choices for q satisfy certain conditions that lead to a successful key recovery attack with complexity significantly lower than the claimed security level for five of the six ciphers in the Rubato family.

Keywords: Algebraic cryptanalysis, composite modulus, Rubato, Key recovery attack, Arithmetization oriented primitives

1 Introduction

Symmetric cryptography is the most fundamental form of encryption and its history goes back thousands of years. In modern times, the first cipher to be standardized was the symmetric encryption algorithm DES in 1977 [63], and since then many other symmetric ciphers have been proposed and standardized. The continued development of other areas of cryptography has often required symmetric ciphers with particular properties, which prompts the proposals of new schemes. This cycle of demand and proposal continues to this day.

As symmetric cryptography evolves, so does its cryptanalysis. Security claims and notions have been formalized, and it has long been standard to assess the

* Author list in alphabetical order.

security of new primitives by examining their susceptibility to known attacks, e.g., linear [61] and differential [17,18] attacks as well as refined and generalized versions of them [15,16,19,55,73]. Another class of attacks is algebraic attacks, such as interpolation [50], higher-order differential [58,55], or computing Gröbner bases for a set of polynomials representing the encryption function.

However, the procedure and success of many attacks depend on the ring or field over which the symmetric primitive is defined, especially algebraic attacks. This dependency is the main topic of our paper, as we assess how attacks on primitives defined over *fields* may carry over to primitives defined over *rings*.

1.1 From Traditional Symmetric Primitives to Symmetric Primitives over Integer Rings Modulo Composites

Traditional Symmetric Primitives. Since computers work with bits, the traditional symmetric ciphers (from DES and onwards) have been built on bit-strings, which must be embedded with mathematical operations in order to perform any cryptographic algorithm. The natural algebraic structure for these ciphers has therefore been the binary field $\mathbb{F}_2$, or one of its extensions $\mathbb{F}_{2^n}$. These fields are very convenient for computers, as addition is simply the XOR operation, and multiplication is simply the AND for $\mathbb{F}_2$, or a particular matrix/vector multiplication over $\mathbb{F}_2$ for multiplying elements of $\mathbb{F}_{2^n}$.

The non-linear operations in these types of ciphers may be performed using S-boxes: permutations on short bit-strings that can easily be implemented via a look-up table. S-boxes should be designed such that describing them with polynomials over the base field produces polynomials of the highest possible degree which the S-box size will permit, as security may be compromised if this is not the case. There are alternative ways of performing non-linear operations, for example the method of ARX ciphers, which uses addition modulo 2^n as an operation in the cipher (see [11,12,13] for examples). However, S-boxes are by far the most common way to perform non-linear operations.

The linear operations in symmetric ciphers should combine the outputs from different non-linear operations as a means to thwart attacks. Iterating the non-linear and linear operations over several rounds quickly makes the polynomials describing encryption depend on all unknown key variables and have the maximum possible degree for the given number of unknowns and the base field. The fields $\mathbb{F}_2$ and the more general $\mathbb{F}_{2^n}$ are well understood for algebraic cryptanalysis, and it has become increasingly easy to argue convincingly that ciphers defined over either these fields are secure against algebraic attacks.

Arithmetization-Oriented Symmetric Primitives. The traditional ciphers work very well for simple encryption/decryption of binary data. However, with the evolution of more sophisticated cryptographic constructions like multi-party computation (MPC), fully homomorphic encryption (FHE), and zero-knowledge (ZK) protocols there has been an increasing demand for *arithmetization-oriented* symmetric primitives to be used together with MPC, ZK, or FHE. While traditional primitives have been designed to be efficient in software and hardware, the

MPC-/ZK-/FHE-friendly primitives are subject to a different efficiency metric. Instead of minimizing the number of bitwise operations, these designs aim to minimize the cost related to the number of non-linear operations. Roughly:

- MPC-friendly schemes aim to minimize the number of non-linear operations necessary to evaluate them;
- ZK-friendly schemes aim to minimize the number of non-linear operations necessary to verify them;
- FHE-friendly schemes aim to minimize the multiplicative depth of their representation when encryption and decryption are expressed as circuits.

Several specialized MPC-/ZK-/FHE-friendly symmetric primitives have recently been proposed, for example MiMC [4], *Vision/Rescue* [6], Chaghri [7], Rasta [32], Ciminion [33], `Reinforced Concrete` [43], HadesMiMC/Poseidon [46,44]. All these primitives are characterized by the following:

- they are usually defined over a prime field $\mathbb{F}_p$ for a large prime p (usually $\log_2(p) \geq 64$), whereas traditional schemes are defined over binary fields;
- they can be described by a simple algebraic expression over their natural field, whereas classical schemes require a more complex algebraic expression.

The first point is motivated by the fact that MPC/ZK/FHE protocols often also rely on primitives from public-key cryptography, which are usually defined over prime fields. It is therefore more convenient to deal with a symmetric primitive that works directly over a prime field, rather than one instantiated over a binary field, which would require conversion to/from the prime field.

The second point is related both to the cost metric of MPC/ZK/FHE protocols, and to the fact that any sub-component (such as the non-linear S-boxes) that defines the symmetric primitive must be computed on the fly. Indeed, due to the huge size of the field these primitives are typically defined over, functions such as the S-box cannot be pre-computed and stored as a look-up table. Some simple non-linear algebraic function is therefore used instead of a look-up table, which leads to a simpler algebraic description, making the scheme potentially vulnerable to algebraic attacks [3,14,34,45,51].

Symmetric Primitives over Quotient Rings with Composite Modulus. The areas of MPC, ZK, FHE, and their associated symmetric primitives are constantly evolving. While traditionally defined over fields such as $\mathbb{F}_p$ and $\mathbb{F}_{p^n}$, there has recently been a surge of new MPC-protocols defined over a ring $\mathbb{Z}_{2^n}$ [28,30,57,62,72]. One method for creating such a ring-based protocol is to construct a MAC over a ring, then apply it in an adapted framework [28].

As with MPC, the vast majority of ZK protocols are also based over fields, but there has recently been a handful of suggestion over rings here as well, e.g., proof systems based on VOLEs over rings [9,10], and the SNARK Rinocchio [38].

The story is slightly different for FHE, as these schemes have most often been defined over polynomial rings, but also here the associated primitives have

typically been defined over fields. There are, however, two recently proposed symmetric schemes defined over rings: Elisabeth [27] and Rubato [49]. Furthermore, these schemes are not used to construct an FHE scheme, but rather combined with already existing FHE schemes as part of a larger framework.

Elisabeth is a family of stream ciphers proposed by Cosseron et al. at Asiacrypt 2022, designed and optimized to be used in combination with the TFHE scheme in a Hybrid Homomorphic Encrypton (HHE) framework. Whereas previous ciphers designed for HHE are defined over fields, Elisabeth is defined over the ring $\mathbb{Z}_{16}$. This definition impacts the design of the scheme from a security perspective, which we discuss briefly in Section 4 in the bigger picture of how to design a non-linear function over a ring. However, defining the cipher over the ring $\mathbb{Z}_{16}$ also has positive impacts on efficiency, as it allows Elisabeth to exploit the various subroutines of TFHE to the fullest, and runs significantly faster than comparable FHE-friendly ciphers for TFHE.

Rubato is a family of ciphers proposed by Ha et al. at Eurocrypt 2022 designed to be used in a transciphering framework for approximate FHE. The cipher is based on the novel idea of introducing noise to a symmetric cipher of a low algebraic degree, which the authors use to argue that very few rounds is sufficient for achieving security. The design of Rubato is very similar to HERA [25], another FHE-friendly cipher. A critical difference is that HERA is defined over a field $\mathbb{F}_p$ for p a prime, while this condition is relaxed to a ring $\mathbb{Z}_q$ for any 25- or 26-bit integer q in the design of Rubato. We refer to Section 5 for a detailed description of the cipher and the framework wherein it is designed to be used.

1.2 Our Contributions

Even though symmetric primitives over rings have been proposed, the cryptanalysis used to argue for their security is developed for primitives over fields. Since symmetric primitives over rings are rather new in the literature, knowledge of cryptanalysis specific to the ring setting is limited. It is therefore timely that the cryptanalysis of symmetric primitives is developed to also assess their security when they are defined over rings, not just fields. In this paper, we aim to start filling this gap.

Security of Symmetric Primitives over the Ring $\mathbb{Z}_q$. First of all, we aim to better understand the differences in the security of a symmetric primitive defined over a ring $\mathbb{Z}_q$ with respect to one defined over a field $\mathbb{F}_p$. We focus first on adapting the brute force attack in Section 2, whilst the algebraic attacks based on linearizaton, Gröbner bases, interpolation, and higher-order differential are discussed in Section 3.

The reason we focus on algebraic attacks is twofold. First, the main focus of this paper is arithmetization-friendly symmetric schemes. These schemes admit a simple algebraic expression, which in general implies that algebraic attacks are much more powerful than statistical attacks. Second, many statistical attacks (e.g., differential [17]) only exploit the property that $(\mathbb{F}_q, +)$ or $(\mathbb{Z}_q, +)$

are groups, and they work whether the analyzed primitive admits a polynomial representation or not. Hence, it is very likely that such attacks work in a similar way over both rings and fields. We leave the problem of analyzing this aspect in more detail for future work.

Most of the mentioned algebraic attacks can be adapted to work when the cryptographic function admits a polynomial representation over $\mathbb{Z}_q$, though they are not as straightforward as in the finite field case. We report the following:

- Brute force attack: perhaps surprisingly, we note that an adaptation of the brute force attack can be significantly cheaper than the straightforward $\mathcal{O}(q^n)$ for a primitive over $\mathbb{Z}_q$ with n secret elements. For instance, if $q = p^y$, the cost is $\mathcal{O}(y \cdot p^n)$, as opposed to $\mathcal{O}(p^{y \cdot n})$.
- Linearization: this attack works similarly to that of the finite field case, though there are subtle difference. In particular, if other linear algebra methods than (an adaptation of) Gaussian elimination is to be used, the solving procedure for a linear equation system over $\mathbb{Z}_q$ must be repeated for every prime factor of q.
- Gröbner bases: if the polynomial system is overdetermined and admits a unique solution, it can be solved through Gröbner basis techniques, albeit at a higher cost than what we expect when solving it over finite fields. Whether solutions to more general polynomial systems over $\mathbb{Z}_q$ can be found by Gröbner basis methods is an open problem;
- Interpolation: there exist dedicated methods for interpolating polynomials over $\mathbb{Z}_q$. Moreover, for some compositions of q, the maximal degree of polynomials can be significantly smaller than q, which could make interpolation attacks competitive.
- Higher-order differential: beyond restricting to prime factors of q, we have not been able to find good generalizations of zero sums. We therefore do not expect higher-order differential attacks to pose much of a threat to ciphers designed over $\mathbb{Z}_q$.

Based on this analysis we discuss how to design the non-linear components of a symmetric scheme over $\mathbb{Z}_q$ for preventing these attacks in Section 4.

Key Recovery Attack on Full Rubato. We present Rubato in Section 5, and give an attack on full Rubato in Section 6. We exploit the fact that Rubato can, in fact, be described by polynomials of low degree in $\mathbb{Z}_q$. As already mentioned, the designers of Rubato introduced adding random noise drawn from a Gaussian distribution to the Rubato key stream to make algebraic attacks much harder. We show how to overcome the addition of random noise by making use of a brute force attack on the key modulo small factors of q. If some factors are small enough the brute force attack has complexity much lower than the claimed security level and allows the attacker to identify positions in the key stream where no noise has been added, leaving the cipher open to a full key recovery with a linearization attack. We provide experimental data verifying that the brute-force method we introduce works as intended and can be used to remove the noise.

We further discuss the assumptions underlying the attack in Section 7 and show that for all but one Rubato variant, at least 25% of the possible choices for q leads to a cipher that can be broken with time complexity less than the claimed security level. For example, if q contains the factor 12, the secret key in Rubato-80M can be successfully recovered with time complexity $2^{57.06}$ using less than 250.000 known key stream elements and less than 25 GB of memory.

Restoring the Security of Rubato. Lastly, in Section 8 we discuss some countermeasures that allow to reestablish the security of Rubato, and which could be crucial for the design of new symmetric primitives over rings $\mathbb{Z}_q$. These include increasing the width of the noise distribution, increasing the number of rounds, and using non-polynomial S-boxes.

A Note on Sections Dependency. Sections 2 through 4 are in large parts general discussion on cryptanalysis and security, whilst Sections 5 through 8.1 deal specifically with Rubato. However, the attack on Rubato does not rely on all the material discussed in the former sections. A reader mainly interested in Rubato may therefore turn to Section 5 after having read Section 3.1, and similarly a reader mainly interested in more general findings on cryptanalysis may find themselves content with skimming Sections 5 through 8.1.

2 General Security of Symmetric Primitives: Fields Versus Integers Modulo q

In this section, we recall fundamental properties of polynomial functions over $\mathbb{Z}_q$, and discuss their immediate security impact. We show that polynomials over $\mathbb{Z}_q$ are generally more restricted in their degrees than polynomials over finite fields. On top of that, we shall see that a symmetric primitive that can be written out as a polynomial in $\mathbb{Z}_q$ will offer less resistance against a brute force attack when compared to a primitive defined over a finite field of similar size.

2.1 Notation and Preliminaries

Notation. We fix the following notation for the rest of the paper. Let q denote a composite integer with prime factorization $q = p_1^{y_1} \cdots p_a^{y_a}$, where p_i is prime and $y_i \geq 1$ for $i = 1, \ldots, a$. Lowercase letters refers to single integers, and boldface lowercase letters refers to vectors or sequences of integers. Uppercase letters indicate functions, including matrices. A table of frequently used notation is found in Appendix A.

Polynomial Functions. It is well known that not every function over $\mathbb{Z}_q$ admits a polynomial representation when q is composite. The existence of null polynomials, i.e., non-trivial elements in $\mathbb{Z}_q[x]$ that evaluate to 0 for all $x \in \mathbb{Z}_q$, then follows from a quick counting argument. Univariate polynomial functions and null

polynomials have been well-studied in the literature, see e.g., [52,67] for general $\mathbb{Z}_q$, and [39] for the important case of $\mathbb{Z}_{p^y}$. Let $\rho = \rho(q)$ be the smallest integer such that $\rho! \equiv 0 \mod q$. Then there are $\prod_{i=0}^{\rho} q/\gcd(i!, q)$ distinct polynomial functions $\mathbb{Z}_q \to \mathbb{Z}_q$, each of which has a canonical representation as a polynomial in $\mathbb{Z}_q[x]$ of degree at most ρ [67, Corollary 9 and Theorem 10]. Note that ρ can be significantly smaller than q. Indeed, we have $\rho(q) = \max\{\rho(p_i^{y_i}) | 1 \leq i \leq a\}$, and $y(p-1)+1 \leq \rho(p^y) \leq py$ [40, Lemma 8]. Finally, a classification of null polynomials is also known [67, Theorem 6],[39, Theorem 1]. While we are not aware of similar studies of *multivariate* polynomial functions over $\mathbb{Z}_q$, a coordinate-wise application of the aforementioned result implies an upper bound of degree $n\rho$ for polynomials in n variables.

Univariate permutation polynomials have also been studied in the literature. An exact characterization is known for $q = 2^y$ [64]. For more general values of q, a formula counting the number of permutation polynomials is given in [70].

A necessary condition for a function F defined over $\mathbb{Z}_q$ to admit a polynomial representation is preservation of congruence. We state the multivariate version in the following, with the proof given in Appendix B.

Lemma 1. *Let u be a divisor of q, and F a polynomial function $\mathbb{Z}_q^n \to \mathbb{Z}_q$.*

i) For any $\mathbf{x} \in \mathbb{Z}_q^n$: $F(\mathbf{x}) \mod u \equiv F(\mathbf{x} \mod u) \mod u$.
ii) $\forall \mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3 \in \mathbb{N}^n$: $F(\mathbf{n}_1 \cdot u + \mathbf{n}_2) \mod u \equiv F(\mathbf{n}_3 \cdot u + \mathbf{n}_2) \mod u$.

The Chinese Remainder Theorem. Many problems in $\mathbb{Z}_q$ can be simplified by working over the (powers of) prime factors of q. The classical tool for this is the Chinese Remainder Theorem (CRT), which we recall in the following.

Theorem 1. *Let $q = \prod_{i=1}^{a} p_i^{y_i}$ where $\gcd(p_i, p_j) = 1$ for all $i \neq j$. Let $b_1, \ldots, b_a \in \mathbb{Z}$ such that $0 \leq b_i < p_i^{y_i}$ for $1 \leq i \leq a$. Then there exists a unique integer x that satisfies the two following conditions:*

- *$0 \leq x < q$, and*
- *for all $1 \leq i \leq a$: $x \equiv b_i \mod p_i^{y_i}$.*

Suppose that we want to recover an element $x \in \mathbb{Z}_q$, for $q = p_1^{y_1} \cdot p_2^{y_2}$, from its values modulo $p_i^{y_i}$. By Bézout's identity, there exist $\mu_1, \mu_2 \in \mathbb{Z}$ such that $\mu_1 \cdot p_1^{y_1} + \mu_2 \cdot p_2^{y_2} = 1$, which can be computed via the extended Euclidean algorithm. Then, the solution x to $x \equiv b_1 \mod p_1^{y_1}$ and $x \equiv b_2 \mod p_2^{y_2}$ is given by $x = b_1 \cdot \mu_2 \cdot p_2^{y_2} + b_2 \cdot \mu_1 \cdot p_1^{y_1}$. This strategy can easily be generalized to values of q with more distinct prime factors.

2.2 Solving Polynomial Systems Modulo q

Let $F_1, \ldots, F_n : \mathbb{Z}_q^n \to \mathbb{Z}_q$ be n polynomial functions and consider the following system of equations

$$\begin{aligned} F_1(x_1, \ldots, x_n) &= b_1 \\ &\vdots \\ F_n(x_1, \ldots, x_n) &= b_n \,. \end{aligned} \tag{1}$$

The discussion in the previous subsection prompts the following strategy for solving such a system of equations.

1. For $i \in \{1, \ldots, a\}$, rewrite Eq. (1) modulo $p_i^{y_i}$.
2. Solve each of the systems modulo $p_i^{y_i}$.
3. Reconstruct a solution in $\mathbb{Z}_q^n$ using CRT.

In the case of $y_i = 1$, solving the system modulo p_i can be done using any algorithm for solving polynomial systems over finite fields. Greater care is needed if $y_i \geq 2$. In the following, we describe a way to further break down the problem.

It is well-known that any $x \in \mathbb{Z}_{p^y}$ can be written as $x = \sum_{i=0}^{y-1} x^{(i)} \cdot p^i$, where $x^{(0)}, \ldots, x^{(y-1)} \in \mathbb{Z}_p$. Based on this, one strategy for solving the equation system modulo p^y is the following:

i) rewrite the equations in Eq. (1) modulo p and solve the system (by exhaustive search if necessary), finding $x_1^{(0)}, \ldots, x_n^{(0)}$;
ii) rewrite the equations modulo p^2. It is simple to note that the only variables appearing in the system are those with superscript (0) and (1), i.e. $x_1^{(0)}, \ldots, x_n^{(0)}$ and $x_1^{(1)}, \ldots, x_n^{(1)}$. Since $x_1^{(0)}, \ldots, x_n^{(0)}$ are known from the previous step, one only needs to solve for $x_1^{(1)}, \ldots, x_n^{(1)}$;
iii) more generally, given $x_1^{(0)}, \ldots, x_n^{(0)}, x_1^{(1)}, \ldots, x_n^{(1)}, \ldots, x_1^{(i-1)}, \ldots, x_n^{(i-1)}$, rewrite the equations modulo p^i, and solve the system in order to find $x_1^{(i)}, \ldots, x_n^{(i)}$.

By working iteratively, one finds a solution $(x_1, \ldots, x_n) \in \mathbb{Z}_{p^y}^n$ to the system of equations modulo p^y. There is the possibility that the reduced systems contain parasitic solutions, i.e., solutions modulo p^i for some $i < y$, that do not lift to a solution modulo p^y. In this case, one can always go back to a smaller modulus and look for a different solution.

While this approach puts a restriction on what values $x_1^{(i)}, \ldots, x_n^{(i)}$ can take, there is still the problem that the system solving routine must be done in the ring $\mathbb{Z}_{p^i}$, where the usual field-based algorithms cannot be readily applied. For now we have mentioned exhaustive search as one possible solving method; more sophisticated methods will be discussed in Section 3.

2.3 Impact on Security

It is well known that the cost of a brute force attack on a symmetric primitive defined over a field $\mathbb{F}_{p^y}$ with a secret key consisting of n field elements should be $\mathcal{O}(p^{n \cdot y})$ if the primitive is well-designed. The following result shows that the cost of breaking any symmetric primitive defined over $\mathbb{Z}_{p^y}$ with a secret key of n elements is significantly lower, $\mathcal{O}(y \cdot p^n)$, *if* the primitive can be described by a system of polynomial equations

Theorem 2. *Let $q = p_1^{y_1} \cdots p_a^{y_a}$ and consider a symmetric primitive over $\mathbb{Z}_q$ relying on the secrecy of n elements. If the primitive can be described by a system of polynomials that admits a constant number of solutions modulo p_i^j, for $1 \leq$*

$i \leq a$ and $1 \leq j \leq y_i$, then the number of evaluations of the primitive needed to perform a brute force attack is

$$\mathcal{O}\left(\sum_{i=1}^{a} y_i \cdot p_i^n\right)$$

Proof. The proof follows the procedure proposed in Section 2.2. We focus on finding a solution modulo p^y; the final complexity statement is obtained by summing over all cases on this form.

By Lemma 1, it is not necessary to write the polynomials representing the primitive out in full. Rather, the solving procedure used in $i)-iii)$ in Section 2.2, at step i_0 for $0 \leq i_0 \leq y-1$, is done by evaluating the primitive for all possible values $(x_1^{(i_0)}, \ldots, x_n^{(i_0)}) \in \mathbb{Z}_p^n$. For $i_0 \geq 1$, this search has to be repeated for each solution that was found modulo p^{i_0-1}. Since we assume a constant number of solutions at every step, the cost of finding all solutions modulo p^y is $\mathcal{O}(yp^n)$. Finally, we note that the last step of combining the solutions with CRT will never be a dominant step, as the run time of the extended Euclidean algorithm is logarithmic in the p_i's. □

We emphasize that it is not necessary for an attacker to know the polynomial representation of the primitive in order to apply the attack. Moreover, we do not expect the restriction on solutions for the various moduli to pose much of a practical limitation. For instance, in the case of a block cipher (resp. stream cipher), any would-be parasitic solution is likely to disappear by including a few extra plaintext-ciphertext pairs (resp. key stream elements).

3 Algebraic Methods over $\mathbb{Z}_q$ for Composite q

We now study the applicability of algebraic attacks on symmetric primitives defined over $\mathbb{Z}_q$. As we are unaware of a generalization of algebraic attacks for primitives that do not admit a polynomial representation, we only concern ourselves with the cases where such a representation exists. As we shall see, there are several differences between applying algebraic attacks to a primitive over a ring and over a field, both with regards to efficiency and success. In fact, some of these algebraic attacks may not work at all *even if* the targeted symmetric primitive admits a polynomial representation over the ring.

We start by discussing linearization and Gröbner basis techniques. Both of these are polynomial system solving methods, and can thus be used in the framework described in Section 2.2. We then go on to investigate attacks based on interpolation and higher-order differentials.

3.1 Linearization Attacks

Linearization is a well-known class of techniques used to solve multivariate polynomial systems of equations over finite fields (see, e.g., [54]). The core idea is

to turn a system of non-linear equations into a linear system by treating each monomial as a separate variable. In general, the method generates polynomials of some degree, up to the point where the number of equations exceeds the number of monomials so a solution can be found by linear algebra.

In symmetric cryptography, it is usually assumed that an attacker has access to sufficiently many equations to directly linearize the system. Recall that the number of possible monomials in a degree d polynomial in $\mathbb{F}[x_1, \ldots, x_n]$, where $|\mathbb{F}| > d$, is $b_{n,d} := \binom{n+d}{n}$. If the symmetric primitive admits a polynomial representation of degree d in n variables, the linearization attack requires $\mathcal{O}(b_{n,d}^{\omega})$ multiplications in $\mathbb{F}$, where $2 < \omega \leq 3$ is the linear algebra constant. The memory cost of the linearization attack is $\mathcal{O}(b_{n,d}^2)$, and the data complexity is $\mathcal{O}(b_{n,d})$.

Linear Algebra Modulo q. When the polynomial system is defined over $\mathbb{Z}_q$ for a composite $q = p_1^{y_1} \ldots p_a^{y_a}$ the usual linear algebra techniques cannot be readily applied. The straightforward Gaussian elimination method can, however, be adapted by restricting to multiplications by units in $\mathbb{Z}_q$ (as opposed to non-zero elements for the field case). When performing the ensuing reduction, one furthermore requires the involved rows to have a unit in their pivot position to guarantee a successful row echelon form. Note that this puts a stronger condition on which rows can contribute in a row-reduction process, but it is unlikely to pose much of a problem in the setting of a linearization attack where extra rows may be sampled. More advanced linear algebra techniques, like Strassen's algorithm [69], can also be applied under stronger assumptions on the underlying matrix.

The idea is to recover solutions over $\mathbb{Z}_{p^y}$ for the various prime factors p of q, and combine them using CRT. For $y = 1$, the solution is found by following the normal algorithm over fields. For $y > 1$, we suggest following the first half of a technique by Dixon, which uses p-adic expansion to recover exact rational solutions from systems of integer coefficients [31]. We recall the method below:

For an invertible matrix A over $\mathbb{Z}_{p^y}$ consider the problem of finding $\mathbf{x}$ so that

$$A\mathbf{x} \equiv \mathbf{b} \mod p^y . \tag{2}$$

Start by finding $C \equiv A^{-1} \mod p$, which is done by solving $CA \equiv I \mod p$, using any algorithm that works over the field $\mathbb{F}_p$. For $0 \leq i \leq y-1$ and $\mathbf{b}_0 = \mathbf{b}$, we then compute $\mathbf{x}_i \equiv C\mathbf{b}_i \mod p$ and $\mathbf{b}_{i+1} = (\mathbf{b}_i - A\mathbf{x}_i)/p$.

Note that, by construction, we have $\mathbf{b}_i - A\mathbf{x}_i \equiv \mathbf{0} \mod p$, so the coordinates in $\mathbf{b}_{i+1}$ are well-defined elements in $\mathbb{Z}_{p^y}$. The solution to Eq. (2) is now given by $\mathbf{x} = \sum_{i=0}^{y-1} \mathbf{x}_i p^i$. This is verified by computing

$$A\mathbf{x} = \sum_{i=0}^{y-1} p^i A\mathbf{x}_i = \sum_{i=0}^{y-1} p^i(\mathbf{b}_i - p\mathbf{b}_{i+1}) = \mathbf{b}_0 - p^y\mathbf{b}_y \equiv \mathbf{b} \mod p^y.$$

3.2 Gröbner Basis Attack

Some of the most powerful techniques for finding a solution to a polynomial equation system involve computing a Gröbner basis of the associated polynomial

ideal. While the majority of work in this direction considers polynomial rings over fields, the theory of Gröbner basis computation has also been generalized to work over more general rings. An overview of this generalization can be found in [2, Section 4]. A reader familiar with the theory of polynomial rings over fields should note that there are several differences between the two cases. In fact, the definitions of fundamental concepts such as S-polynomials, polynomial reductions and even that of a Gröbner basis itself, must be adapted when working over rings, due to the existence of zero divisors and lack of multiplicative inverses. Still, with the proper adaptations in place, it can be shown that there exists a Gröbner basis for any ideal in a polynomial ring over $\mathbb{Z}_q$.

One of the most efficient algorithms for computing Gröbner bases, the F_4 algorithm [36], has also been extended to polynomial rings over $\mathbb{Z}_q$ in the computer algebra system Magma [21]. It is not clear whether the typical procedure for complexity estimation of the F_4 algorithm (c.f. [8]) can be generalized to polynomial rings over the integers modulo q. We have run several experiments with the F_4 algorithm on randomly generated polynomial systems over both $\mathbb{Z}_{p^y}$ and $\mathbb{F}_{p^y}$, and report the results in Appendix C. In all experiments we observe that both time and memory costs are significantly larger for the polynomial systems over $\mathbb{Z}_{p^y}$, than it is for their finite field counterpart. Further investigations of the complexity of Gröbner basis computation over $\mathbb{Z}_q$, beyond this qualitative comparison, are out of scope for this work.

Solutions from Gröbner Bases. If the polynomial system is sufficiently overdetermined and has a unique solution, we expect to be able to read the solution directly from the Gröbner basis when the coefficients are in a field. We also observed this in all the $\mathbb{Z}_{p^y}$-experiments in Appendix C. The process of recovering a solution from a Gröbner basis of more general polynomials systems, however, is more involved (see, e.g., [23]).

When working over a field, the typical strategy is to change the monomial order of the Gröbner basis with the FGLM-algorithm [37] into an order where a univariate polynomial can be found. A solution to one of the variables is then found by factoring this univariate polynomial, and the remainder of the (multivariate) solution is found by back-substitution and repeated solving of univariate polynomial equations. There are several reasons why the same strategy cannot be applied to polynomials over $\mathbb{Z}_q$. Firstly, we are not aware of any work that has adapted the FGLM-algorithm to Gröbner bases over rings. Secondly, factorization in $\mathbb{Z}_q$ is not as well-behaved as in the finite field case, and there are polynomials where no better factorization method than brute-force is known [71]. Finally, it is not even clear whether the theoretical foundations of this strategy (c.f. [23, Section 2]) can be extended to rings.

3.3 Interpolation Attack

The goal of the interpolation attack [50] is to construct a polynomial that describes a cryptographic function. Given the interpolation polynomial, the attacker can use it to set up distinguishers, forgeries, or key recovery attacks.

If the cryptographic function is described by a univariate polynomial of degree d over a finite field, then this polynomial can be constructed from the Lagrange interpolation formula using d distinct input-output pairs. This formula relies on the existence of inverses of non-zero elements, and thus cannot be readily applied to polynomials over $\mathbb{Z}_q$. That said, the problem of interpolating polynomials modulo q has been studied in several papers, and some of these techniques can be applied in an attack.

Interpolation of Univariate Polynomials Modulo q. Recall from Section 2.1 that univariate polynomial functions have a canonical representation of degree $d \leq \rho = \rho(q)$. This representation can be recovered from the evaluations of the values $0, 1, \ldots, d-1$, by following the procedure described in the proof of [40, Corollary 7]. Another interpolation method, based on Newton interpolation, is described in [39] for $\mathbb{Z}_{p^y}$. While this is a different approach, it still requires the evaluation of all inputs $0, 1, \ldots, d-1$. We remark that only knowing the polynomial function modulo factors $p_i^{y_i}$ of q does not pose much of a drawback for an attacker. Indeed, Lemma 1 ensures that an attacker can evaluate any $x \in \mathbb{Z}_q$ modulo these factors, and find the correct output using the CRT.

As noted in Section 2.1, the upper degree bound $\rho(q)$ can be significantly smaller than q. Therefore, in order to ensure that interpolation attacks will not pose a problem, any cryptographic function with a polynomial representation over $\mathbb{Z}_q$ should be careful in its choice of q.

3.4 Higher-Order Differential Attack

Given a vectorial Boolean function F over $\mathbb{F}_2^n$ of degree d, the higher-order differential attack [58,55] traditionally exploits the fact that $\bigoplus_{\mathbf{x}\in\mathcal{V}} F(\mathbf{x}) = 0$ for any affine subspace $\mathcal{V} \subseteq \mathbb{F}_2^n$ of dimension strictly larger than d. A generalization of the attack to any prime field $\mathbb{F}_p$ has recently been proposed in [14]. For this version, it is shown that if $F : \mathbb{F}_p^n \to \mathbb{F}_p$ is of degree $\deg(F) < h(p-1)$, then

$$\sum_{\mathbf{x}\in\mathcal{W}} F(\mathbf{x}) = 0 \tag{3}$$

where $\mathcal{W} \subseteq \mathbb{F}_p^n$ is an affine subspace of dimension at least h [14, Corollary 1]. The result can be generalized further to polynomials over $\mathbb{F}_{p^n}$ using the existence of a vector space isomorphism $\mathbb{F}_{p^n} \cong \mathbb{F}_p^n$.

Differentials of Polynomials over $\mathbb{Z}_q$. For polynomials over $\mathbb{Z}_q$, a zero-sum similar to that of Eq. (3) can be set up by restricting to a prime factor modulus in the following manner.

Proposition 1. *Let p be a prime divisor of q, and $F \in \mathbb{Z}_q[x_1, \ldots, x_n]$ be a polynomial of degree $< h(p-1)$, and let $\mathcal{V} \subseteq \mathbb{F}_p^n \cong \mathbb{Z}_p^n \subseteq \mathbb{Z}_q^n$ be an affine subspace of dimension at least h. Then:*

$$\sum_{\mathbf{x}\in\mathcal{V}} F(\mathbf{x}) \equiv 0 \mod p.$$

Proof. Due to Lemma 1 and the result in [14], we have that

$$\sum_{\mathbf{x}\in\mathcal{V}} F(\mathbf{x}) \mod p \equiv \sum_{\mathbf{x}\in\mathcal{V}} F(\mathbf{x} \mod p) \mod p \equiv \sum_{\mathbf{x}\in\mathcal{V}\subseteq\mathbb{F}_p^n} F(\mathbf{x}) = 0.$$

□

Unlike the finite field case, this result cannot be generalized to prime powers, since there is no vector space isomorphism between $\mathbb{Z}_p^n$ and $\mathbb{Z}_{p^n}$. Indeed, we have performed small-scale experiments on low degree polynomials F over $\mathbb{Z}_{2^n}$ which generally does not sum to zero, even when the sum is taken over all of $\mathbb{Z}_{2^n}$.

The zero-sum in Eq. (3) crucially relies on the fact that $\sum_{x\in\mathbb{F}_p} x^i = 0$ for each $i < p-1$. One may ask whether it is possible to obtain a similar result, and thus a better generalization than Proposition 1, when working over $\mathbb{Z}_q$ directly. In Appendix D we answer this question in the negative when q is the product of distinct primes, by giving an exact characterization of $\sum_{x\in\mathbb{Z}_q} x^i$, for any i.

4 Designing a Non-Linear (S-box) Function over $\mathbb{Z}_q$

We discussed possible algebraic attacks on symmetric primitives over rings $\mathbb{Z}_q$ in the previous section. Based on this, we now discuss three possible strategies for designing the S-boxes and/or non-linear functions with the goal of making algebraic attacks as hard as possible. A similar discussion for the linear layer is presented in Appendix E.

4.1 Polynomial Non-Linear Function over $\mathbb{Z}_q$

As in the field case, one possible design strategy is to simply define the non-linear invertible S-box function via an invertible polynomial function. Note that it is well-known how to design invertible polynomial functions over a ring $\mathbb{Z}_q$, see e.g. [64,66,74] for some concrete examples.

The advantage of this design is the possibility to define the S-box function in a very efficient way, especially when the polynomial function is sparse. The obvious downside is that it is possible to describe the complete encryption function as a polynomial system, making the brute force attack described in Theorem 2 possible. The algebraic attacks described in the previous section should also be considered in this case.

4.2 Learning from Elisabeth: Look-up Tables

Another possible way of designing the non-linear function is via a look-up table, which is exactly what is proposed for the Elisabeth stream cipher [27]. Its non-linear layer is defined using 8 different S-box functions $S_1, S_2, \ldots, S_8$ that are defined over $\mathbb{Z}_{16}$ via look-up tables (not invertible in Elisabeth's case), such that they do not admit any polynomial representation over $\mathbb{Z}_{16}$.

The advantage of using look-up tables is the possibility to set up a non-linear function that does not admit any polynomial representation over the ring $\mathbb{Z}_q$, which immediately makes the cipher immune to any algebraic attack working over $\mathbb{Z}_q$. The disadvantage of this strategy is that the ciphers are less applicable, for example in the HHE setting, which combines a symmetric cipher with an FHE scheme. Although FHE schemes are defined to evaluate any polynomial homomorphically, there is no guarantee that a symmetric cipher which does *not* admit a polynomial representation is possible to evaluate, much less that it will be efficient. TFHE, the FHE scheme Elisabeth is designed to be combined with, is able to evaluate a look-up table very efficiently for the parameter choices set by Elisabeth, but it is currently the only FHE scheme able to do so, and hence the only FHE scheme Elisabeth may practically be combined with.

4.3 "Cut and Sew" Approach

Either of the two strategies just proposed have their own pros and cons. Defining an S-box as a simple polynomial allows one to evaluate large S-boxes efficiently, while a non-polynomial S-box is immune to direct algebraic attacks. The best scenario would be to have a design approach that incorporates the advantages of both methods, and the "cut and sew" approach we propose, inspired by ideas from [43], aims to do this.

In the following, we consider two concrete examples, one where $q = p_1 \cdot p_2$ with $p_1 \neq p_2$ and one where $q = p^2$. By combining and generalizing them, it is possible to design a non-linear function for any composite q. Given $x \in \mathbb{Z}_q$, the "cut and sew" approach works as follows:

1. decompose $x \in \mathbb{Z}_q$ to its components with respect to the factors of q;
2. apply a non-linear function on each component of x;
3. recompose the new components together.

Let us consider the two cases in more detail.

Case: $q = p_1 \cdot p_2$. Let us decompose each $x \in \mathbb{Z}_q$ as

$$x = x_2 \cdot p_2 + x_1$$

where $x_1 \in \{0, 1, \ldots, p_2 - 1\}$ and $x_2 \in \{0, 1, \ldots, p_1 - 1\}$. An S-box S over $\mathbb{Z}_q$ can be then defined as

$$S(x) = S_2(x_2) \cdot p_2 + S_1(x_1),$$

where $S_1 : \mathbb{F}_{p_2} \to \mathbb{F}_{p_2}$ and $S_2 : \mathbb{F}_{p_1} \to \mathbb{F}_{p_1}$. It is easy to see that if both S_1 and S_2 are invertible, then S is invertible as well.

Both S_1 and S_2 can be instantiated with either a look-up table or a polynomial function, keeping in mind that both S_1 and S_2 are defined over fields. In particular, by instantiating S_1 and S_2 with polynomials over $\mathbb{F}_{p_2}$ and $\mathbb{F}_{p_1}$, it is possible to efficiently evaluate these functions even if p_1 and p_2 are large.

In order to prevent the algebraic attacks previously discussed, it makes sense to choose S_1 and S_2 such that S does not admit any polynomial representation over $\mathbb{Z}_q$. By Lemma 1, S admits a polynomial representation only if

$$\forall i \in \{1,2\}: \qquad S(x \cdot p_i + y) \mod p_i \equiv S(z \cdot p_i + y) \mod p_i \tag{4}$$

for all relevant tuples (x, y, z). It is easy to verify that this equality always holds for $i = 2$. Indeed, $S(x \cdot p_2 + y) \equiv S(z \cdot p_2 + y) \equiv S_1(y) \mod p_2$ by the definition of S.

For the case $i = 1$, one has to prove that such an equality is not satisfied for at least one relevant tuple (x, y, z), depending on the details of S_1 and S_2. For instance, if S_1 and S_2 are chosen as random permutations, then Eq. (4) is not satisfied with probability $1 - 1/p_1$ for any given tuple (x, y, z). Since $1 - 1/p_1 \geq 1/2$, a few tests should be sufficient for verifying that an S-box S does not admit a polynomial representation. We show how to construct an S-box S that does not admit a polynomial representation given an orthomorphism over $\mathbb{F}_{p_2}$ and only in the case $p_1 > p_2$ in Appendix F. We give this construction for completeness, and leave the problem to generalize such strategy, or to propose new ones, open for future research.

While the method described above ensures that S cannot be described as a polynomial over $\mathbb{Z}_q$, we note that S still reduces to S_1 modulo p_2. Thus, some care is needed in the construction to ensure that this cannot be exploited in an attack. A possible way to prevent this exploitation is by using two different S-boxes S, S' over $\mathbb{Z}_q$ defined as follows

$$\begin{aligned} x = x_2 \cdot p_2 + x_1 &\mapsto S(x) = S_2(x_2) \cdot p_2 + S_1(x_1) \\ x = x'_1 \cdot p_1 + x'_2 &\mapsto S'(x) = S'_1(x'_1) \cdot p_1 + S'_2(x'_2) \end{aligned}$$

where $x_2, x'_2 \in \mathbb{F}_{p_1}$, $x_1, x'_1 \in \mathbb{F}_{p_2}$, $S_2, S'_2 : \mathbb{F}_{p_1} \to \mathbb{F}_{p_1}$, and $S_1, S'_1 : \mathbb{F}_{p_2} \to \mathbb{F}_{p_2}$. Hence, S admits a polynomial representation modulo p_2, while S' admits it modulo p_1. As a result, a symmetric primitive depending on both S and S' will not admit a polynomial representation modulo any of p_1 or p_2. Note that many MPC-/ZK-/FHE-friendly symmetric primitives (e.g., [6,22,42,43,47,48]) are all defined via multiple S-boxes.

Case: $q = p^2$. Let $x = x_2 \cdot p + x_1$ as before for $x_1, x_2 \in \{0, 1, \ldots, p-1\}$. Here, we suggest to define

$$S(x) = S_2(x_1) \cdot p + S_1(x_2)\,,$$

where $S_1, S_2 : \mathbb{F}_p \to \mathbb{F}_p$, and where we note that x_1 and x_2 are "swapped", in the sense that the output element that is multiplied by p depends only on x_1, while the input element multiplied by p depends only on x_2.[4] As before, such an S-box is invertible if and only if both S_1, S_2 are invertible. Moreover:

[4] Note that the subspace $\{x \cdot p + x \in \mathbb{Z}_{p^2} \mid \forall x \in \mathbb{F}_p\}$ is invariant if $S_1 = S_2$. However, it is possible to break such invariant subspace via a proper choices of round constants (see e.g. [59,60] for details).

Lemma 2. *Let p be a prime integer. The function S over $\mathbb{Z}_{p^2}$ defined as $S(x = x_2 \cdot p + x_1) = S_2(x_1) \cdot p + S_1(x_2)$, where $x_1, x_2 \in \{0, 1, \ldots, p-1\}$ and S_1 is invertible, never admits a polynomial representation over $\mathbb{Z}_{p^2}$.*

Proof. If S has a polynomial representation, then it must satisfy Lemma 1, that is, $S(y \cdot p + x) \mod p = S(z \cdot p + x) \mod p$, which implies $S_1(y) = S_1(z)$ for each $x, y, z \in \{0, 1, \ldots, p-1\}$. Obviously, this condition is never satisfied if S_1 is bijective and $y \neq z$. □

The statistical properties of an S-box constructed using the cut-and-sew approach may very well be sub-optimal. This should not cause a big problem when the S-box is large since probabilities of differential or linear trails should still be easy to make small enough to rule out differential or linear attacks. However, it is something a designer should keep in mind and check if using this approach for any particular construction.

5 Rubato

An HHE framework involves the homomorphic evaluation of some cryptographic function, e.g., encryption of a symmetric cipher, and it is therefore desirable that this function has a low multiplicative depth so the evaluation can be done efficiently. However, a low depth is not advisable from a security perspective, as it makes the cipher susceptible to the attacks described in Section 3. Furthermore, the strategies described in Section 4 do not combine well with FHE, except in specialized circumstances.

Rubato [49] is an attempt to strike a balance between low multiplicative depth and security, as it is a family of symmetric cipher which admits a polynomial representation of low degree, but with the addition of Gaussian noise to the key stream to prevent algebraic attacks. We describe the ciphers in this section, as well as the transciphering framework it is intended for. The notation of the original paper is mostly adapted to ours.

5.1 Description of Rubato

For an integer $q \geq 2$, let $\mathbb{Z}_q := \mathbb{Z} \cap (-q/2, q/2]$ and $\mathbb{Z}_q^\times$ be the multiplicative group of $\mathbb{Z}_q$. We view the state X of Rubato as a $v \times v$ matrix over $\mathbb{Z}_q$, where $x_{i,j}$ denotes the entry in the i-th row and in the j-th column. Let the block size n be the square of some $v \in \mathbb{Z}_{>0}$.

For λ-bit security Rubato takes a symmetric key $\mathbf{k} \in \mathbb{Z}_q^n$, a nonce $\mathbf{nc} \in \{0,1\}^\lambda$ and a counter $i \in \mathbb{Z}_{\geq 0}$ as input, and returns a block of key stream

$$\mathbf{z} = \mathsf{Rubato}[\mathbf{k}, \mathbf{nc}, i](\mathbf{is}) \in \mathbb{Z}_q^\ell$$

for some $\ell < n$, where $\mathbf{is} = (1, 2, \ldots, n) \in \mathbb{Z}_q^n$ denotes an initial (fixed) state. Encryption of a message vector $\boldsymbol{\mu} \in \mathbb{R}^\ell$ by Rubato is defined by

$$\mathbf{c} = \lfloor \Delta \cdot \boldsymbol{\mu} \rceil + \mathbf{z} \mod q,$$

where $\Delta \in \mathbb{R}$ is a scaling factor dependent on the norm of the message.

Components. We introduce the following components of Rubato:

Add-Round Key and the Key-Schedule: the Add-Round Key function (ARK) over $\mathbb{Z}_q^n$ is defined as

$$\mathrm{ARK}[\mathbf{k}, i](\mathbf{x}) = \mathbf{x} + \mathbf{k} \bullet \mathbf{rc_i},$$

where $\bullet$ denotes component-wise multiplication modulo q and $\mathbf{rc_i} \in (\mathbb{Z}_q^\times)^n$ are round constants defined via an XOF that takes the nonce $\mathbf{nc}$ and the counter i as input.

Mix Columns and Mix Rows: The linear transformation in Rubato is composed of two consecutive operations: MixColumns and MixRows. Let $X \in \mathbb{Z}_q^{v \times v}$ be the state of Rubato. The linear layer is simply defined as

$$X \xrightarrow{\text{MixColumns}} M_v \times X \xrightarrow{\text{MixRows}} (M_v \times X) \times M_v^T$$

where M_v^T denotes the transpose of a particular matrix $M_v \in \mathbb{Z}_q^{v \times v}$. For the particular cases $v \in \{4, 6, 8\}$, M_v is defined as

$$M_v = \begin{bmatrix} \mathbf{y_v} \\ \mathbf{y_v} \lll 1 \\ \vdots \\ \mathbf{y_v} \lll v-1 \end{bmatrix},$$

where $\mathbf{y_4} = [2, 3, 1, 1]$, $\mathbf{y_6} = [4, 2, 4, 3, 1, 1]$ and $\mathbf{y_8} = [5, 3, 4, 3, 6, 2, 1, 1]$, and $\mathbf{y_v} \lll j$ denotes the cyclic rotation of $\mathbf{y_v}$ by j positions.

Feistel: A quadratic type-III Feistel [75] is applied on the state. Given the input $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{Z}_q^n$, the output is

$$\mathrm{Feistel}(\mathbf{x}) = (x_1, x_2 + x_1^2, x_3 + x_2^2, \ldots, x_n + x_{n-1}^2).$$

Rubato. Using the components described above, we illustrate the round function of Rubato in Fig. 1 and define the function as follows:

$$\mathrm{RF}[\mathbf{k}, i] = \mathrm{ARK}[\mathbf{k}, i] \circ \mathrm{Feistel} \circ \mathrm{MixRows} \circ \mathrm{MixColumns}.$$

The final round differs slightly from the rest in that a second linear transformation is applied, together with the truncation function $\mathrm{Tr}_{n,\ell}$, which simply cuts away the last $n - \ell$ entries of the state (i.e., $\mathrm{Tr}_{n,\ell}(x_1, \ldots, x_n) = (x_1, \ldots, x_\ell)$):

$$\begin{aligned} \mathrm{Fin}[\mathbf{k}, i + r] = &\mathrm{Tr}_{n,\ell} \circ \mathrm{ARK}[\mathbf{k}, i + r] \circ \mathrm{MixRows} \circ \mathrm{MixColumns} \circ \\ &\mathrm{Feistel} \circ \mathrm{MixRows} \circ \mathrm{MixColumns}\,, \end{aligned}$$

This final round is followed by the last function AGN, which adds Gaussian noise. Let $\mathbf{x} = (x_1, \ldots, x_\ell) \in \mathbb{Z}_q^\ell$ and $e_1, \ldots, e_\ell \leftarrow D_{\alpha q}$ be sampled independently

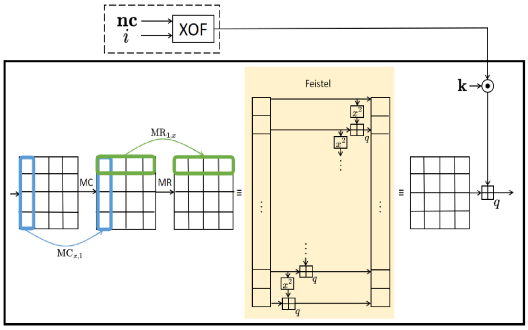


Fig. 1: The round function of Rubato.

Table 1: Proposed parameters of Rubato. λ is the security level, n is the block size, ℓ is the length of the keystream, $\lceil \log_2 q \rceil$ is the bit length of q with $\mathbb{Z}_q$ being the ring Rubato instances operate on, $(\alpha q)^2/2\pi$ is the variance of the Gaussian distribution the noise is sampled from, r is the total number of rounds.

Parameter	λ	n	ℓ	$\lceil \log_2 q \rceil$	αq	r
Par-80S	80	16	12	26	11.1	2
Par-80M	80	36	32	25	2.7	2
Par-80L	80	64	60	25	1.6	2
Par-128S	128	16	12	26	10.5	5
Par-128M	128	36	32	25	4.1	3
Par-128L	128	64	60	25	4.1	2

according to an one-dimensional discrete Gaussian distribution $D_{\alpha q}$ with zero mean and variance $(\alpha q)^2/2\pi$. Then,

$$\mathrm{AGN}(\mathbf{x}) = (x_1 + e_1, \ldots, x_\ell + e_\ell)\,.$$

All in all, the r-round stream cipher Rubato is defined as follows: [5]

$$\mathsf{Rubato}[\mathbf{k}, \mathbf{nc}, i] = \mathrm{AGN} \circ \mathrm{Fin}[\mathbf{k}, i{+}r] \circ \mathrm{RF}[\mathbf{k}, i{+}r{-}1] \circ \cdots \circ \mathrm{RF}[\mathbf{k}, i{+}1] \circ \mathrm{ARK}[\mathbf{k}, i].$$

The parameters of Rubato proposed by the authors are given in Table 1.

5.2 About the Value of q: Rubato in the RtF Framework

The choice of the parameter q greatly impacts the security of Rubato, and so to better understand the different aspects of this choice, we recall the RtF (Real-to-Finite field) transciphering framework, which is the greater context the Rubato

[5] For completeness, we present an equivalent version of Rubato in Appendix G.

ciphers are intended for. We stress, in particular, that there is no requirement for q to be prime from an applicability perspective of this framework.

The RtF framework is a type of HHE framework for the approximate homomorphic encryption scheme CKKS. The framework lets a client encrypt their data using a symmetric cipher, a (comparatively) cheap operation, and the encrypted result is sent to a server which performs the heavy, homomorphic encryption and further cloud computation. The framework was originally proposed by Cho et al. with the symmetric cipher HERA [25], which is a more traditional stream cipher than Rubato. HERA consists of several rounds of linear and non-linear operations, it does not add Gaussian noise to the key stream, and it is explicitly defined over a prime field for security. However, HERA is used in the RtF framework in the same way as Rubato is in the description below.

On the client side of the RtF framework, the client will feed a key $\mathbf{k}$ into Rubato, use the resulting key stream to encrypt a message, and finally send this encrypted message to the server. The client will also encode and encrypt the key $\mathbf{k}$ using the homomorphic encryption scheme FV and send the resulting ciphertext to the server. Upon receiving this encryption of $\mathbf{k}$, the server runs Rubato *homomorphically* to produce an FV-encryption of the key stream, whilst the encryption of the message is transformed into an FV ciphertext. The FV-encryption of the key stream is then subtracted from the FV-encryption of the symmetrically encrypted message, producing an FV-encryption of just the message. Finally, an operation termed 'half bootstrapping' is performed to transform the FV ciphertext into a CKKS ciphertext. After this step is completed, the RtF framework has served its purpose, and the server may evaluate the ciphertext further using only the CKKS scheme.

Since the RtF framework uses Rubato in combination with the FV and CKKS schemes, there are some overlaps in the parameters of the three schemes. Of most importance to us is that the modulus q of Rubato has to match the plaintext modulus of FV, as the key $\mathbf{k}$ is encrypted using FV, and the plaintext modulus of FV must therefore accommodate for this. There is no restriction on this plaintext modulus other than requiring it to be an integer larger than 1 [35]. In practice, however, it is usually taken to be a prime congruent to 1 modulo $2N$, where N is the dimension of the ring FV is defined over, but this choice is made *purely* for efficiency reasons, as the choice of plaintext modulus has no impact on the security of the FV scheme [35,1]. This is in great contrast to Rubato, where the choice of q may severely compromise the security.

5.3 Non-Invertible and/or Non-MDS Matrices for Rubato

Before presenting the attack on Rubato, we point out that the matrices that define the linear layer of Rubato are not always invertible and/or not always MDS for several values of q. We recall that a matrix $M \in \mathbb{Z}_q^{n \times n}$ is *invertible* if and only if its determinant $\det(M)$ is co-prime with q, i.e., $\gcd(\det(M), q) = 1$.

Definition 1 (MDS [29]). *The* branch number *of* $M \in \mathbb{Z}_q^{n \times n}$ *is defined as* $\mathcal{B}(M) = \min_{x \in \mathbb{Z}_q^n \setminus \{0\}} \{\mathrm{hw}(x) + \mathrm{hw}(M(x))\}$*, where* $\mathrm{hw}(\cdot)$ *is the bundle weight*

in wide trail terminology. A matrix $M \in \mathbb{Z}_q^{n \times n}$ is called a Maximum Distance Separable (MDS) matrix if and only if $\mathcal{B}(M) = n + 1$.

In the case of Rubato, we check all the possible integer values for q that are 25 or 26 bits. The number of q's such that M_v for $v \in \{4, 6, 8\}$ is invertible or MDS and the corresponding frequencies are provided in Table 2. In the 'Invertible' part, where M_v is invertible over $\mathbb{Z}_q$, the column 'Total' gives the total number of such q's, the column 'Prime' gives the number of such prime q's, and the column 'Composite' gives the number of such composite q's. The corresponding frequencies among all the possible $3 \cdot 2^{24}$ q values are given below the numbers. The 'MDS' columns are similar. We discuss these results in detail in Appendix H.

Table 2: The number of invertible matrices and MDS matrices of Rubato matrices M_v ($v = 4, 6, 8$) over all possible q-values of 25 or 26 bits.

Property / Matrix	Invertible			MDS		
	Total	Prime	Composite	Total	Prime	Composite
$v = 4$	$2^{25.04}$	$2^{21.46}$	$2^{24.91}$	$2^{23.23}$	$2^{21.46}$	$2^{22.73}$
	68.57%	5.72%	62.85%	19.56%	5.72%	13.85%
$v = 6$	$2^{23.68}$	$2^{21.46}$	$2^{23.33}$	$2^{22.62}$	$2^{21.46}$	$2^{21.77}$
	26.67%	5.72%	20.95%	12.83%	5.72%	7.11%
$v = 8$	$2^{25.0}$	$2^{21.46}$	$2^{24.87}$	$2^{22.11}$	$2^{21.46}$	$2^{20.64}$
	66.72%	5.72%	61.00%	8.96%	5.72%	3.24%

Impact on the Security. At the current state of the art, we are not aware of any attack on Rubato (or Rasta-like schemes) that exploits the possible non-invertibility of the linear layers that instantiate Rubato. For example, both Masta and the Rasta-like variant designs proposed in [41] are defined using non-invertible components. Still, no attacks have been proposed on them. This is related to the fact that the encryption function changes at every evaluation for these ciphers. Hence, even if an internal collision is found, different round functions are applied on the same state, with the results of different outputs.

However, using the same non-MDS matrix twice in one round of Rubato might lead to weaker diffusion than expected by the designers, especially due to the small number of rounds. We leave the open problem of exploiting non-invertible and/or non-MDS matrices for future work.

6 Key Recovery Attack on Rubato

We present a key recovery attack on Rubato, which breaks the claimed security level of five of the six proposed variants of Rubato when the modulus q belongs to a certain class. The steps of the attacks are as follows:

1. First, we recover the correct key and noise modulo m, when m is a factor of q lying in a particular interval.
2. Then, we recover the positions in the key stream where the noise added by $\mathrm{AGN}(\cdot)$ is exactly 0.
3. Finally, we recover the secret key by setting up a system of polynomial equations using the knowledge of positions with no noise, and solving the system by re-linearization.

For ease of exposition, we specify some further notation:

- We denote the Rubato algorithm without the final $\mathrm{AGN}(\cdot)$ operation as $\mathsf{Ru} = \mathsf{Ru}[\mathbf{k}, \mathrm{nc}, i]$.
- The stream of $\mathbb{Z}_q$-elements produced by running Ru is denoted as $\mathbf{w} = (w_1, w_2, \ldots)$.
- For either Rubato or Ru, we let Rubato_m or Ru_m denote that we are executing all the steps of the cipher in the ring $\mathbb{Z}_m$ rather than $\mathbb{Z}_q$, producing a stream of elements in $\mathbb{Z}_m$.

After presenting the attack, we will present the necessary assumptions q must meet in order to have an attack with complexity less than 2^λ given the parameter sets of the different Rubato variants, and the fraction of the valid choices for q that results in weak instances of Rubato.

6.1 Recovering Key and Noise Modulo a Small Factor of q

First, we describe how to recover the correct key values and noise values modulo m, where m is a factor of q lying in a particular interval. The upper and lower bounds on the interval depend on the Rubato variant and will be determined in Section 7.1.

Assume the attacker is given s elements of known key stream $z_1, \ldots, z_s$ generated by an unknown secret key $(k_1, \ldots, k_n) \in \mathbb{Z}_q^n$, where

$$s := \left\lceil \binom{n + 2^r}{2^r} \cdot \alpha q \right\rceil .$$

We then have the equations

$$z_i = w_i + e_i \mod q, \text{ for } 1 \leq i \leq s\,,$$

where the noise values e_i are drawn from $D_{\alpha q}$.

Let m be a non-trivial factor of q, and let $\tilde{\mathbf{k}} = (\tilde{k}_1, \ldots, \tilde{k}_n) \in \mathbb{Z}_m^n$ denote a guess for the values of the secret key modulo m. Note that if m satisfies $m^n < 2^\lambda$ it is possible to do an exhaustive search over all possible $(\tilde{k}_1, \ldots, \tilde{k}_n)$ and compute the Rubato_m key stream with complexity lower than the claimed security level. Furthermore, from Lemma 1 we have the equality

$$\mathsf{Rubato}_m[\mathbf{k} \mod m, \mathrm{nc}, i] = \mathsf{Rubato}[\mathbf{k}, \mathrm{nc}, i] \mod m.$$

For each guess $(\tilde{k}_1, \ldots, \tilde{k}_n)$, let $\tilde{w}_1, \ldots, \tilde{w}_s$ be the stream generated by $\mathsf{Ru}_m[\tilde{\mathbf{k}}]$.

In order to check the correctness of a guess, we note the following. If the guess is wrong we expect the values $\tilde{w}_i$ to be distributed uniformly at random over $\mathbb{Z}_m$, and in particular, we expect the candidate noise values computed as

$$\tilde{e}_i = (z_i \mod m) - \tilde{w}_i \text{ for } i = 1, \ldots, s$$

to be distributed uniformly at random over $\mathbb{Z}_m$. This assumption stems from the common expectation that a good cipher behaves like a random permutation. If the guess $(\tilde{k}_1, \ldots, \tilde{k}_n)$ is equal to $(k_1 \mod m, \ldots, k_n \mod m)$ where $(k_1, \ldots, k_n)$ is the correct secret key, we have

$$\tilde{e}_i = (e_i \mod m) \text{ for } i = 1, \ldots, s,$$

where the e_i-values are the actual noise values drawn from $D_{\alpha q}$ when producing the key stream $\mathbf{z}$.

If m is large enough relative to the αq parameter, we can distinguish between a correct and incorrect guess. In other words, the non-uniformity of the Gaussian distribution shines through even if the numbers drawn from $D_{\alpha q}$ are only given modulo m. In Section 7.1 we establish the exact bounds on m for five of the six Rubato variants that result in brute force attacks on $\tilde{\mathbf{k}}$ where we can distinguish the correct guess from the wrong ones with complexity smaller than 2^λ. As we shall see, this bound cannot be established in the case of Rubato-128L. After performing this part of the attack, we learn the correct values of $e_i \mod m$ for $i = 1, \ldots, s$, and of $k_j \mod m$ for $j = 1, \ldots, n$.

6.2 Recovering the Key Modulo a Larger Factor of q and Positions in the Key Stream with no Noise

After recovering $e_i \mod m$ for $1 \leq i \leq s$ and $k_j \mod m$ for $1 \leq j \leq n$ for some factor m of q, we proceed to identify every position in the key stream where the noise added by $\mathrm{AGN}(\cdot)$ is 0. In the following, let f be a non-trivial factor of q/m.

Case: $f \leq m$. If $f \leq m$ we can repeat the attack from Section 6.1, this time running Rubato_{fm}. Similar to the attack described in Section 2, the attacker can use the knowledge of the correct key values modulo m to speed up the exhaustive search. For each $k_i \mod fm$, the attacker does not guess on all values $0, \ldots, fm-1$, but only on the values $(k_i \mod m) + j \cdot m$ for $0 \leq j < f$.

Note that there is no lower bound on the size of f. If the attacker is able to distinguish the $D_{\alpha q}$ distribution modulo m from the uniform distribution, the attacker is certainly able to distinguish $D_{\alpha q}$ from uniform modulo $2m$, or any higher multiple of m. The attacker learns the correct key values modulo fm, and the correct e_i-values modulo fm after doing the exhaustive search modulo fm, with a complexity that is no higher than the initial step.

Case: $f = f_1 \cdots f_b$ where all $f_i \leq m$ In this case, it is possible to repeat the exhaustive search for each factor f_i of q/m where $2 \leq f_i \leq m$. The complexity of this is at most $b \cdot m^n$. However, our aim in this step is not to maximize the modulus fm for which one can recover the correct key modulo fm. Rather, we are interested in just having a large enough f such that all noise values e_i for $i = 1, \ldots, s$ will satisfy the bound $|e_i| < fm$ with high probability. In Section 7.1 we determine a threshold t depending on αq such that when $fm \geq t$ and e_i is drawn from $D_{\alpha q}$, then $|e_i| < fm$ for all $i = 1, \ldots, s$ with probability higher than 99%. So when we find $e_i \equiv 0 \mod fm$, we have that $e_i \equiv 0 \mod q$ with high probability as well, and not $e_i = \pm fm$. In other words, when the attacker finds $e_i \equiv 0 \mod fm$ where e_i is drawn from $D_{\alpha q}$, the attacker knows that, with high probability, there has been no noise added by $\mathrm{AGN}(\cdot)$ for this particular index i. No added noise will be a rather common occurrence, as the noise value 0 will be sampled from $D_{\alpha q}$ at a rate of $1/\alpha q$.

We define $\mathcal{I}$ to be the set of indices where no noise has been added by $\mathrm{AGN}(\cdot)$:

$$\mathcal{I} = \{i \mid e_i \equiv 0 \mod q\}.$$

So when $fm | q$, $fm > t$ and all prime factors of f are smaller than or equal to m, the attacker can recover the correct $\mathcal{I}$ with probability higher than 99%.

6.3 Key-Recovery of the full Rubato Key

Assuming the attacker knows $\mathcal{I}$, the set of indices in the Rubato key stream where no noise has been added, it is fairly straightforward to set up a system of polynomial equations in the unknown key variables that can be solved by linearization. As Rubato is designed to have very low multiplicative complexity, and hence have very few iterations of the round function, we will see that the size of the polynomial equations in $k_1, \ldots, k_n$ and the complexity for solving them is small compared to the security parameter.

Treating the unknown $k_1, \ldots, k_n$ as variables, the attacker starts by evaluating all operations for producing the Ru stream in sequence. This yields the expressions $F_i(k_1, \ldots, k_n) = w_i$ for $1 \leq i \leq s$.

When $i \in \mathcal{I}$, the attacker knows that $w_i = z_i$, so they can extract exactly these equations to set up the system

$$\begin{aligned} F_{i_1}(k_1, \ldots, k_n) &= z_{i_1} \\ F_{i_2}(k_1, \ldots, k_n) &= z_{i_2} \\ \vdots \qquad & \qquad \vdots \\ F_{i_b}(k_1, \ldots, k_n) &= z_{i_b}, \end{aligned} \tag{5}$$

for all $i_j \in \mathcal{I}$. Recall that we assume the attacker knows s elements of key stream where $s = \left\lceil \binom{n+2^r}{2^r} \cdot \alpha q \right\rceil$. Since the noise value 0 is sampled at a rate of $1/\alpha q$ we expect the size of $\mathcal{I}$ to be $|\mathcal{I}| \geq \binom{n+2^r}{2^r}$.

Each polynomial in Eq. (5) has degree 2^r. For instance, since every Rubato variant with 80-bit security has $r = 2$, the degree of the polynomials in Eq. (5) is

Table 3: The time complexities for solving a linearizied system of equations modulo one factor of q. To recover the secret Rubato key solving the linearized systems must be repeated at most 26 times, depending on q.

Rubato variant	Degree	# of monomials	Solving complexity
Rubato-80S	4	4845	$2^{34.28}$
Rubato-80M	4	91390	$2^{46.14}$
Rubato-80L	4	814385	$2^{54.98}$
Rubato-128S	32	$2^{41.04}$	$2^{114.90}$
Rubato-128M	8	$2^{27.40}$	$2^{76.72}$
Rubato-128L	4	814385	$2^{54.98}$

4. The number of monomials appearing in F_i is given by $\binom{n+2^r}{2^r}$. Since we expect to have more equations than monomials in Eq. (5) we can solve the system by Gaussian elimination. Here we also keep in mind that we are working with a composite q, so we need to use the method explained in Section 3.1, and in particular, we must solve the linearized system once for every prime factor of q.

The complexity of solving Eq. (5) for one prime factor is $\mathcal{O}(\binom{n+2^r}{2^r}^\omega)$, where $\omega \leq 3$ is the linear algebra constant. A conservative (and realistic) choice for ω is $\omega = 2.8$. Table 3 gives the degrees, number of monomials, and complexities for solving one linearized system modulo $p|q$ for the six different variants.

As we can see from Table 3, all complexities for breaking noise-less Rubato by linearization are significantly smaller than the security bounds 2^{80} and 2^{128}, even when this step has to be repeated a small number of times. Assuming q satisfies the assumptions necessary for doing steps 1 and 2 of the attack, the attacker can do a full key recovery attack on Rubato with complexity lower than 2^λ. Pseudo-code for the complete key recovery attack on Rubato is given in Appendix I, where we also use the notation introduced in the next section.

7 Assumptions and Cost of the Attack on Rubato

7.1 Assumptions on q

The following assumptions on the integer q that defines the ring $\mathbb{Z}_q$ used in Rubato must hold in order for the attack in Section 6 to be successful.

Assumption 1 *There exists an integer m such that $m|q$ and $m_{min} \leq m \leq m_{max}$, where m_{min} and m_{max} will be determined below.*

For Rubato with claimed λ-bit security, m cannot be too large, as we need $m^n < 2^\lambda$ in order to have a valid attack. Moreover, m cannot be too small as this makes the noise modulo m impossible to distinguish from random, hence the bounds $m_{\min}$ and $m_{\max}$.

Assumption 2 *There exists an integer f such that all prime factors of f are at most m, $fm|q$, and $fm > t$, where the threshold t will be determined below.*

This condition is necessary to be able to recover the positions where we know the noise value is exactly 0.

There exist values of q such that both these assumptions hold. These q-values give weak instances of Rubato, and must be avoided in an actual use case. Before looking into the weak choices for q, we compute the bounds $m_{\min}, m_{\max}$, and the threshold t mentioned above for a general Rubato variant with claimed λ-bit security.

Determining t. In order to determine the threshold t, recall that the aim is to find the smallest value $t \in \mathbb{Z}$ for each Rubato variant such that

$$e \mod (fm) \equiv 0 \qquad \Rightarrow \qquad e \mod q \equiv 0$$

with overwhelming probability when $fm > t$ and $fm|q$. This reduces to finding the smallest integer t such that, with high probability, the error values satisfy $|e_i| \leq t$ for all $1 \leq i < s$. In the analysis below we specify "high probability" to mean above 99%.

Let $G(x) = \frac{1}{\alpha q} \cdot e^{-x^2/2\sigma^2}$ be the Gaussian function describing the discrete Gaussian distribution $D_{\alpha q}$ the noise in Rubato is drawn from, where $\sigma = \alpha q/\sqrt{2\pi}$. Then $G(x)$ gives the probability that we sample $x \leftarrow D_{\alpha q}$. Thus, the probability that we sample $e_i \leftarrow D_{\alpha q}$ such that $|e_i| \leq t$ can be computed as

$$\Pr(|e_i| \leq t) = \sum_{x=-t}^{t} G(x)\,.$$

We want to make sure that after sampling s noise values, the probability that all of them lie in the interval $[-t, t]$ is at least 0.99. This condition translates into finding the smallest $t \in \mathbb{Z}$ such that $0.99 \leq \left(\sum_{x=-t}^{t} G(x)\right)^s$. We then get the desired bounds by finding the smallest t that satisfies this inequality for the different Rubato variants. These values are listed in Table 4.

Determining $m_{\min}$ and $m_{\max}$. As already stated, we must have $m^n < 2^\lambda$ in order to have a valid attack. This inequality provides the upper bound $m_{\max}$:

$$m_{\max} := \lfloor 2^{\lambda/n} \rfloor\,.$$

The lower bound $m_{\min}$ is the smallest value where it is possible to distinguish the correct key guess $\tilde{\mathbf{k}}$ modulo $m_{\min}$ from all the wrong ones. To find this lower bound, we first compute the probability that $e \mod m = x$ for $0 \leq x < m$ when e is sampled from $D_{\alpha q}$:

$$\Pr_m(x) = \sum_{i=-\infty}^{\infty} G(im + x)\,.$$

Secondly, for a given modulus m we split the set $\{0, \ldots, m-1\}$ into two disjoint subsets $\mathcal{I}_1$ and $\mathcal{I}_2$ as

$$\mathcal{I}_1 := \{x \mid \Pr_m(x) \geq 1/m\} \qquad \text{and} \qquad \mathcal{I}_2 := \{x \mid \Pr_m(x) < 1/m\}\,. \quad (6)$$

For a given stream $\tilde{\mathbf{e}} = \tilde{e}_1, \ldots, \tilde{e}_s$ of candidate noise values (that may or may not be sampled from $D_{\alpha q}$) and $0 \leq i < m$, let $u_i(\tilde{\mathbf{e}})$ be the frequency of observing the value i in the stream $\tilde{\mathbf{e}}$ modulo m, that is,

$$u_i(\tilde{\mathbf{e}}) = \frac{|\{\tilde{e}_j \in \tilde{\mathbf{e}} | \tilde{e}_j \mod m = i\}|}{s}\,.$$

Note that when $\tilde{\mathbf{e}}$ is sampled from $D_{\alpha q}$, we expect $u_i(\tilde{\mathbf{e}}) \approx \Pr_m(i)$ for $0 \leq i < m$.

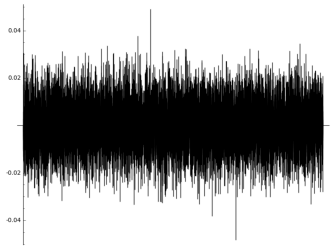
Score Value for $\tilde{\mathbf{k}}$. For a given key guess $\tilde{\mathbf{k}} = (\tilde{k}_1, \ldots, \tilde{k}_n)$ modulo m, we now define a score value for $\tilde{\mathbf{k}}$. First, execute $\mathsf{Ru}_m[\tilde{\mathbf{k}}]$ producing the stream $\tilde{w}_1, \ldots, \tilde{w}_s$. From the known key stream $z_1, \ldots, z_s$, compute the candidate noise value modulo m as $\tilde{e}_i = (z_i - \tilde{w}_i) \mod m$, for $1 \leq i \leq s$. We define the score for the key guess $\tilde{\mathbf{k}}$ as

$$\mathrm{Sc}(\tilde{\mathbf{k}}) = \sum_{i \in I_1} (u_i(\tilde{\mathbf{e}}) - 1/m) + \sum_{i \in I_2} (1/m - u_i(\tilde{\mathbf{e}}))\,.$$

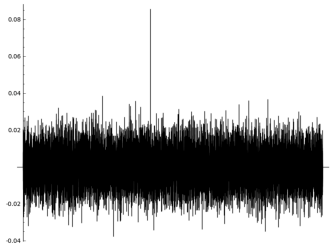
The score function gives a measure of how much the candidate noise value produced by $\tilde{\mathbf{k}}$ deviates from the uniform distribution in the same way as values drawn from $D_{\alpha q}$ modulo m will deviate from uniform. When $\tilde{\mathbf{k}}$ is the correct guess modulo m, we expect $\mathrm{Sc}(\tilde{\mathbf{k}}) = \sum_{i=0}^{m} |\Pr_m(i) - 1/m|$. This value will be significantly greater than 0, provided m is large enough relative to αq. When $\tilde{\mathbf{k}}$ is a wrong key guess, we expect the noise values in $\tilde{\mathbf{e}}$ to be distributed uniformly at random, and hence a score value of $\mathrm{Sc}(\tilde{\mathbf{k}}) = 0$.

If the assumption that all wrong key guesses give uniformly distributed noise values modulo m holds, it is possible to compute the probability that the correct key guess gives the unique highest score value of all guesses for the key modulo m. However, we have observed that wrong key guesses in 2-round Rubato do *not* produce noise values that are distributed uniformly at random (see Fig. 2e and Fig. 2f). Therefore we have found $m_{\min}$ heuristically, listed in Table 4, by checking the smallest m that produces a score value for the correct key guess that clearly stands out among many (at least 14640) wrong key guesses.

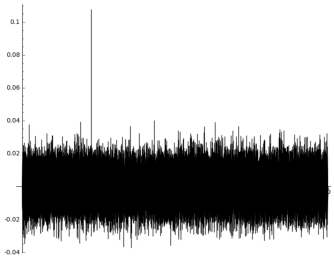
Set of Susceptible Values. We list the bounds $m_{\min}$ and $m_{\max}$ and the threshold t that allow attacks with complexity lower than 2^λ for each parameter set defined for Rubato in Table 4. We performed an exhaustive search on 26-bit numbers (for Rubato-80S and Rubato-128S) and 25-bit numbers (for the other variants) to find the percentage of q's satisfying Assumption 1 and 2. The last column of Table 4 shows the percentage of vulnerable choices of q. For Rubato-128L we have $m_{\min} > m_{\max}$, so we do not have an attack on this Rubato variant.



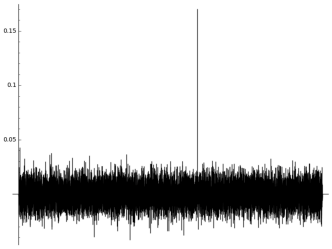
(a) **Rubato-80S**: distinguishing correct key guess modulo 11 using 14641 key samples.



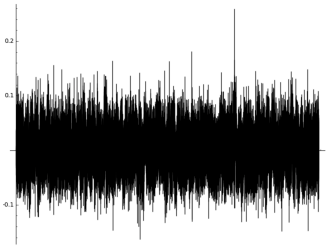
(b) **Rubato-128S**: distinguishing correct key guess modulo 11 using 14641 key samples.



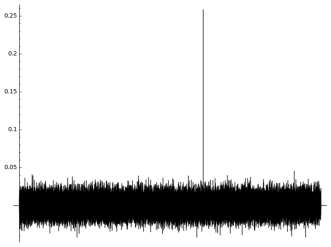
(c) **Rubato-80M**: distinguishing correct key guess modulo 3 using 59049 key samples.



(d) **Rubato-128M**: distinguishing correct key guess modulo 5 using 15625 key samples.



(e) **Rubato-80L**: distinguishing correct key guess modulo 2 using 65536 key samples.



(f) **Uniformly random noise**: score values for 65536 noise vectors modulo 2, produced by the `random()` function in C. The maximum score value from Fig. 2e is also inserted in the data set.

Fig. 2: Plots of score values computed for key guesses modulo m. The correct guess can be distinguished from all the wrong guesses. Comparing Fig. 2e and 2f shows that wrong key guesses in 2-round Rubato do not produce candidate noise that is uniformly random.

Table 4: Lower and upper bounds for the modulus m, threshold t and percentage of choices of q vulnerable to the attack for the various Rubato variants.

Rubato variant	$\mathbf{m}_{\min}$	$\mathbf{m}_{\max}$	**t**	**Fraction of vulnerable q's**
Rubato-80S	11	31	24	42.05%
Rubato-80M	3	4	7	25%
Rubato-80L	2	2	4	25%
Rubato-128S	11	255	35	58.47%
Rubato-128M	5	11	12	37.25%
Rubato-128L	-	-	-	0%

7.2 Practical Verification of the Attack

We have verified the attack described in Section 6 experimentally[6]. We also report on the experiments determining the smallest m for which we can distinguish a correct key guess modulo m from the wrong ones.

In all experiments, we selected a 25- or 26-bit q with some small factors, a key $\mathbf{k}$ at random, and produced 10000 elements of Rubato key stream. In an actual full key recovery attack, we need s to be higher for the relinearization part, but $s = 10000$ is sufficient for distinguishing the $D_{\alpha q}$ distribution from a (supposedly) uniform distribution modulo m.

Next, we fixed a value of m and made between $11^4 = 14641$ and $2^{16} = 65536$ guesses on the key modulo m, including the correct guess, and stored their score values in a file. Finally, we made plots of the score values in each file as a bar chart and verified that the maximum score value seen indeed corresponds to the correct key modulo m. The plots of the score values observed for the values $m_{\min}$ in Table 4 for the different Rubato variants are given in Fig. 2a-2e.

In Fig. 2f we have also included a plot of score values computed from noise values modulo 2, sampled by the `random()` function in C, together with the maximum score value from Fig. 2e. If the noise values produced by wrong key guesses in Rubato-80L were truly distributed uniformly at random, the plots of Fig. 2e and 2f should look the same. The fact that there is a significantly higher variance in Figure 2e shows that 2-round Rubato does not behave like a random permutation. This makes it somewhat harder to distinguish wrong key guesses from the correct one, but the attack still works for all values of m given in Table 4.

7.3 Attack Complexities

Finally, we investigate the lowest possible attack complexities of the Rubato attack in concrete numbers. For Rubato-80 and Rubato-128M, the lowest attack

[6] The code can be found at `https://github.com/Simula-UiB/RubatoAttack`

Table 5: Lowest time complexities of key recovery attack, where q has particular factors.

Rubato variant	**Assumption on q**	**Time**	**Data**	**Memory**
Rubato-80S	$44\|q$	$2^{55.35}$	$2^{15.71}$	$2^{24.48}$
Rubato-80M	$12\|q$	$2^{57.06}$	$2^{17.91}$	$2^{32.96}$
Rubato-80L	$4\|q$	2^{65}	$2^{20.31}$	$2^{39.27}$
Rubato-128S	$q = 11 \cdot 2^{22}$	$2^{55.35}$	$2^{44.43}$	$2^{44.43}$
Rubato-128M	$20\|q$	$2^{83.59}$	$2^{29.44}$	$2^{39.27}$

complexities occur when $m = m_{\min}$ and $f = 2^g$ for $g = \lceil \log_2(t/m) \rceil$. The time complexities are given as the number of times we guess on $\mathbf{k}$ and produce a sufficient amount of key stream to distinguish a correct key guess from the wrong ones. The total key recovery attack complexity C_{kr} is then given as

$$C_{kr} = m^n + g \cdot 2^n + C_{relin},$$

where C_{relin} is the complexity of doing the relinearization step. The complexities for relinearization in Table 3 are given in terms of number of multiplications and additions in $\mathbb{Z}_q$, and not as computing key stream for a particular key guess. When recomputing the complexities in Table 3 to make them comparable to the work done for each guess of $\tilde{\mathbf{k}}$, it becomes clear that apart from Rubato-128S, C_{relin} is negligibly small compared to doing steps 1 and 2 of the attack.

For Rubato-128S, the relinearization step is the dominant part of the attack. For $m = 11$ and the particular value $q = 11 \cdot 2^{22}$ (a 26-bit number) it is much faster to recover the complete key by guessing modulo 11 in step 1, followed by 22 successive key guesses modulo 2 in step 2, and skip solving the linearized system in step 3 altogether. So for this particular value of q the complexity of recovering the complete key in Rubato-128S is given as $C_{kr} = 11^{16} + 22 \cdot 2^{16}$.

Table 5 shows particular conditions on q that give attacks with the smallest possible time complexities. For completeness, we also list the memory and data complexities, where both of these are given as the number of $\mathbb{Z}_q$-elements the attacker needs to store.

8 Final Remarks

8.1 Restoring the Security of Rubato

There are several ways Rubato can be made secure against the attack presented in Section 6. Here we discuss some of them.

Restricting q to Prime Numbers. The easiest way to prevent our attack is to simply restrict q to be prime. Since our attack is based on the assumption that q contains small factors, this restriction immediately gives Rubato instances that

are immune to any small-factor attack. As already mentioned in Section 5.2, it is most common to choose the plaintext modulus of FV, the other part q plays in the RtF framework, to be prime. However, we stress that this choice is made for efficiency [24] *not* security in the FV scheme [1,35]. As our attack has demonstrated, restricting q to be prime is a choice made for security in Rubato, not convenience.

Increasing the Width of the Noise Distribution. Another way to protect the scheme against the small-factor attack is to increase the width of the noise distributions. If the value of αq is sufficiently high so that one cannot distinguish the correct key modulo m for $m \leq 2^{\lambda/n}$, then one cannot perform the initial exhaustive key search modulo m with a complexity that is lower than the claimed security level. This approach allows keeping Rubato defined for general values of q. The drawback of increasing the αq parameter is that there will be more noise added to the key stream, and hence less accuracy in the decrypted plaintext values. This loss of precision will also compound when doing further operations in the CKKS scheme.

Increasing the Number of Rounds. A third alternative is to increase the number of rounds used in Rubato such that the solving complexity of the relinearization step is high enough to make the scheme secure against our attack. Recall that the total complexity C_{kr} of the key recovery attack depends on the complexity C_{relin} of doing the relinearization step, which in turn depends on the number of monomials appearing in the polynomials defining the stream produced by Ru. More rounds will produce more monomials and therefore a higher solving complexity, as one can see in Table 3. The main disadvantage of this approach is the loss of efficiency, as applying more rounds would result in a higher multiplicative depth. This would be detrimental to Rubato's use case in a transciphering framework, where a low multiplicative depth of the decryption function is necessary for efficiency reasons.

Using Non-Polynomial S-boxes. Lastly, avoiding the use of polynomial S-boxes is yet another way to provide security against our attack, since it requires that the S-boxes in the scheme admit a polynomial representation. As mentioned in Section 4.2, this is the case for the stream cipher Elisabeth, whose S-box functions are defined using look-up tables. A full discussion on how to design such S-box functions can be found in Section 4. However, using a look-up table rather than a polynomial function in Rubato would result in a severe efficiency loss. The Elisabeth ciphers are defined specifically to be combined with the TFHE scheme, which can very efficiently evaluate a look-up table homomorphically *for free* during a bootstrapping operation [68]. The strategy of using a look-up table as the S-box is therefore very well suited for the TFHE scheme, but not for stream ciphers designed to be combined with any other FHE scheme, such as Rubato.

8.2 Conclusion

Symmetric primitives over rings is a rapidly growing area of cryptography, in part spurred on by the development in MPC, ZK, and FHE. Constructing primitives over rings instead of fields might prove advantageous in certain cases, exemplified by the efficiency of Elisabeth compared to other FHE-friendly ciphers. However, as our key recovery attack on Rubato shows, it is important to take care when choosing the ring the primitive is defined over, since the ring greatly affects how susceptible the primitive is to attacks. We stress that we do not mean to suggest that rings should be avoided as a base structure for symmetric primitives, since several of the proposed schemes have useful properties. Rather, we emphasize that a more thorough cryptanalysis over rings is needed to ensure that proposed primitives are secure, and hope to see more work in this direction.

Acknowledgments. Lorenzo Grassi is supported by the German Research Foundation (DFG) within the framework of the Excellence Strategy of the Federal Government and the States – EXC 2092 CaSa – 39078197.

References

1. Lattigo v4. Online: https://github.com/tuneinsight/lattigo, Aug. 2022. EPFL-LDS, Tune Insight SA.
2. W. W. Adams and P. Loustaunau. *An Introduction to Gröbner Bases*, volume 3. American Mathematical Society, 1994.
3. M. R. Albrecht, C. Cid, L. Grassi, D. Khovratovich, R. Lüftenegger, C. Rechberger, and M. Schofnegger. Algebraic Cryptanalysis of STARK-Friendly Designs: Application to MARVELlous and MiMC. In *Advances in Cryptology - ASIACRYPT 2019*, volume 11923 of *LNCS*, pages 371–397. Springer, 2019.
4. M. R. Albrecht, L. Grassi, C. Rechberger, A. Roy, and T. Tiessen. MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity. In *Advances in Cryptology - ASIACRYPT 2016*, volume 10031 of *LNCS*, pages 191–219, 2016.
5. J. Allsop and I. M. Wanless. Degree of orthomorphism polynomials over finite fields. *Finite Fields and Their Applications*, 75(101893).
6. A. Aly, T. Ashur, E. Ben-Sasson, S. Dhooghe, and A. Szepieniec. Design of Symmetric-Key Primitives for Advanced Cryptographic Protocols. *IACR Trans. Symmetric Cryptol.*, 2020(3):1–45, 2020.
7. T. Ashur, M. Mahzoun, and D. Toprakhisar. Chaghri - A FHE-friendly Block Cipher. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022*, pages 139–150. ACM, 2022.
8. M. Bardet, J.-C. Faugère, and B. Salvy. On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations. In *Proceedings of the International Conference on Polynomial System Solving*, pages 71–74, 2004.
9. C. Baum, L. Braun, A. Munch-Hansen, B. Razet, and P. Scholl. Appenzeller to Brie: Efficient Zero-Knowledge Proofs for Mixed-Mode Arithmetic and $\mathbb{Z}_{2^k}$. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 192–211, 2021.

10. C. Baum, L. Braun, A. Munch-Hansen, and P. Scholl. Moz$\mathbb{Z}_{2^k}$arella: Efficient Vector-OLE and Zero-Knowledge Proofs over $\mathbb{Z}_{2^k}$. In *Advances in Cryptology - CRYPTO 2022*, volume 13510 of *LNCS*, pages 329–358. Springer, 2022.
11. R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers. The SIMON and SPECK lightweight block ciphers. In *Proceedings of the 52nd Annual Design Automation Conference, 2015*, pages 175:1–175:6. ACM, 2015.
12. C. Beierle, A. Biryukov, L. C. dos Santos, J. Großschädl, L. Perrin, A. Udovenko, V. Velichkov, and Q. Wang. Lightweight AEAD and hashing using the SPARKLE permutation family. Submission to the NIST lightweight cryptographic standardization process (Finalist).
13. D. J. Bernstein. The Salsa20 Family of Stream Ciphers. In *New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 of *LNCS*, pages 84–97. Springer, 2008.
14. T. Beyne, A. Canteaut, I. Dinur, M. Eichlseder, G. Leander, G. Leurent, M. Naya-Plasencia, L. Perrin, Y. Sasaki, Y. Todo, and F. Wiemer. Out of Oddity - New Cryptanalytic Techniques Against Symmetric Primitives Optimized for Integrity Proof Systems. In *Advances in Cryptology - CRYPTO 2020*, volume 12172 of *LNCS*, pages 299–328. Springer, 2020.
15. E. Biham, A. Biryukov, and A. Shamir. Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. In *Advances in Cryptology - EUROCRYPT 1999*, volume 1592 of *LNCS*, pages 12–23. Springer, 1999.
16. E. Biham, O. Dunkelman, and N. Keller. The Rectangle Attack - Rectangling the Serpent. In *Advances in Cryptology - EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 340–357. Springer, 2001.
17. E. Biham and A. Shamir. Differential Cryptanalysis of DES-like Cryptosystems. *J. Cryptol.*, 4(1):3–72, 1991.
18. E. Biham and A. Shamir. *Differential Cryptanalysis of the Data Encryption Standard*. Springer, 1993.
19. A. Bogdanov, D. Khovratovich, and C. Rechberger. Biclique Cryptanalysis of the Full AES. In *Advances in Cryptology - ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 344–371. Springer, 2011.
20. N. Bordes, J. Daemen, D. Kuijsters, and G. V. Assche. Thinking Outside the Superbox. In *Advances in Cryptology - CRYPTO 2021*, volume 12827 of *LNCS*, pages 337–367. Springer, 2021.
21. W. Bosma, J. J. Cannon, C. Fieker, and A. Steel (eds.). Gröbner Bases over Euclidean Rings. In *Magma Handbook v.2.27*. Computational Algebra Group, School of Mathematics and Statistics, University of Sydney. https://magma.maths.usyd.edu.au/magma/handbook/text/1259#14396.
22. C. Bouvier, P. Briaud, P. Chaidos, L. Perrin, R. Salen, V. Velichkov, and D. Willems. New Design Techniques for Efficient Arithmetization-Oriented Hash Functions:Anemoi Permutations and Jive Compression Mode. Cryptology ePrint Archive, Paper 2022/840, 2022. https://eprint.iacr.org/2022/840.
23. A. Caminata and E. Gorla. Solving multivariate polynomial systems and an invariant from commutative algebra. In *Arithmetic of Finite Fields: 8th International Workshop, WAIFI 2020, Rennes, France, July 6–8, 2020, Revised Selected and Invited Papers 8*, pages 3–36. Springer, 2021.
24. H. Chen, K. Laine, and R. Player. Simple Encrypted Arithmetic Library - SEAL v2.1. Cryptology ePrint Archive, Paper 2017/224, 2017. https://eprint.iacr.org/2017/224.

25. J. Cho, J. Ha, S. Kim, B. Lee, J. Lee, J. Lee, D. Moon, and H. Yoon. Transciphering framework for approximate homomorphic encryption. In M. Tibouchi and H. Wang, editors, *Advances in Cryptology - ASIACRYPT 2021*, volume 13092 of *LNCS*, pages 640–669. Springer, 2021.
26. C. Cid, L. Grassi, A. Gunsing, R. Lüftenegger, C. Rechberger, and M. Schofnegger. Influence of the Linear Layer on the Algebraic Degree in SP-Networks. *IACR Trans. Symmetric Cryptol.*, 2022(1):110–137, 2022.
27. O. Cosseron, C. Hoffmann, P. Méaux, and F. Standaert. Towards Case-Optimized Hybrid Homomorphic Encryption - Featuring the Elisabeth Stream Cipher. In S. Agrawal and D. Lin, editors, *Advances in Cryptology - ASIACRYPT 2022*, volume 13793 of *LNCS*, pages 32–67. Springer, 2022.
28. R. Cramer, I. Damgård, D. Escudero, P. Scholl, and C. Xing. SPD$\mathbb{Z}_{2^k}$: Efficient MPC mod 2^k for Dishonest Majority. In *Advances in Cryptology - CRYPTO 2018*, volume 10992 of *LNCS*, pages 769–798. Springer, 2018.
29. J. Daemen and V. Rijmen. The Wide Trail Design Strategy. In *Cryptography and Coding - IMACC 2001*, volume 2260 of *LNCS*, pages 222–238. Springer, 2001.
30. A. P. Dalskov, D. Escudero, and M. Keller. Fantastic Four: Honest-Majority Four-Party Secure Computation With Malicious Security. In *USENIX Security Symposium*, pages 2183–2200, 2021.
31. J. D. Dixon. Exact Solution of Linear Equations Using P-Adic Expansions. *Numerische Mathematik*, 40(1):137–141, 1982.
32. C. Dobraunig, M. Eichlseder, L. Grassi, V. Lallemand, G. Leander, E. List, F. Mendel, and C. Rechberger. Rasta: A Cipher with Low ANDdepth and Few ANDs per Bit. In *Advances in Cryptology - CRYPTO 2018*, volume 10991 of *LNCS*, pages 662–692. Springer, 2018.
33. C. Dobraunig, L. Grassi, A. Guinet, and D. Kuijsters. Ciminion: Symmetric encryption based on toffoli-gates over large finite fields. In *Advances in Cryptology - EUROCRYPT 2021*, volume 12697 of *LNCS*, pages 3–34. Springer, 2021.
34. M. Eichlseder, L. Grassi, R. Lüftenegger, M. Øygarden, C. Rechberger, M. Schofnegger, and Q. Wang. An Algebraic Attack on Ciphers with Low-Degree Round Functions: Application to Full MiMC. In *Advances in Cryptology - ASIACRYPT 2020*, volume 12491 of *LNCS*, pages 477–506. Springer, 2020.
35. J. Fan and F. Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Paper 2012/144, 2012. https://eprint.iacr.org/2012/144.
36. J.-C. Faugère. A new efficient algorithm for computing Gröbner bases (F_4). *Journal of pure and applied algebra*, 139(1-3):61–88, 1999.
37. J.-C. Faugère, P. Gianni, D. Lazard, and T. Mora. Efficient Computation of Zero-Dimensional Gröbner Bases by Change of Ordering. *Journal of Symbolic Computation*, 16(4):329–344, 1993.
38. C. Ganesh, A. Nitulescu, and E. Soria-Vazquez. Rinocchio: SNARKs for Ring Arithmetic. Cryptology ePrint Archive, Paper 2021/322, 2021. https://eprint.iacr.org/2021/322.
39. R. Geelen, I. Iliashenko, J. Kang, and F. Vercauteren. On polynomial functions modulo p^e and faster bootstrapping for homomorphic encryption. In C. Hazay and M. Stam, editors, *Advances in Cryptology - EUROCRYPT 2023*, volume 14006 of *LNCS*, pages 257–286. Springer, 2023.
40. P. Gopalan. Query-Efficient Algorithms for Polynomial Interpolation over Composites. *SIAM Journal on Computing*, 38(3):1033–1057, 2008.

41. L. Grassi. Bounded Surjective Quadratic Functions over $\mathbb{F}_p^n$ for MPC-/ZK-/HE-Friendly Symmetric Primitives. Cryptology ePrint Archive, Paper 2022/1313, 2022. https://eprint.iacr.org/2022/1313.
42. L. Grassi, Y. Hao, C. Rechberger, M. Schofnegger, R. Walch, and Q. Wang. Horst Meets Fluid-SPN: Griffin for Zero-Knowledge Applications. Cryptology ePrint Archive, Paper 2022/403, 2022. https://eprint.iacr.org/2022/403.
43. L. Grassi, D. Khovratovich, R. Lüftenegger, C. Rechberger, M. Schofnegger, and R. Walch. Reinforced Concrete: A Fast Hash Function for Verifiable Computation. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 202*, pages 1323–1335. ACM, 2022.
44. L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger. Poseidon: A New Hash Function for Zero-Knowledge Proof Systems. In *30th USENIX Security Symposium, USENIX Security 2021*, pages 519–535. USENIX Association, 2021.
45. L. Grassi, D. Khovratovich, S. Rønjom, and M. Schofnegger. The Legendre Symbol and the Modulo-2 Operator in Symmetric Schemes over $\mathbb{F}_p^n$ Preimage Attack on Full Grendel. *IACR Trans. Symmetric Cryptol.*, 2022(1):5–37, 2022.
46. L. Grassi, R. Lüftenegger, C. Rechberger, D. Rotaru, and M. Schofnegger. On a Generalization of Substitution-Permutation Networks: The HADES Design Strategy. In *Advances in Cryptology - EUROCRYPT 2020*, volume 12106 of *LNCS*, pages 674–704. Springer, 2020.
47. L. Grassi, S. Onofri, M. Pedicini, and L. Sozzi. Invertible Quadratic Non-Linear Layers for MPC-/FHE-/ZK-Friendly Schemes over $\mathbb{F}_p^n$ Application to Poseidon. *IACR Trans. Symmetric Cryptol.*, 2022(3):20–72, 2022.
48. L. Grassi, M. Øygarden, M. Schofnegger, and R. Walch. From Farfalle to Megafono via Ciminion: The PRF Hydra for MPC Applications. In C. Hazay and M. Stam, editors, *Advances in Cryptology - EUROCRYPT 2023*, volume 14007 of *LNCS*, pages 255–286. Springer, 2023.
49. J. Ha, S. Kim, B. Lee, J. Lee, and M. Son. Rubato: Noisy Ciphers for Approximate Homomorphic Encryption. In *Advances in Cryptology - EUROCRYPT 2022*, volume 13275 of *LNCS*, pages 581–610. Springer, 2022.
50. T. Jakobsen and L. R. Knudsen. The Interpolation Attack on Block Ciphers. In *Fast Software Encryption – FSE 1997*, volume 1267 of *LNCS*, pages 28–40. Springer, 1997.
51. N. Keller and A. Rosemarin. Mind the Middle Layer: The HADES Design Strategy Revisited. In *Advances in Cryptology - EUROCRYPT 2021*, volume 12697 of *LNCS*, pages 35–63. Springer, 2021.
52. A. J. Kempner. Polynomials and their residue systems. *Transactions of the American Mathematical Society*, 22(2):240–266, 1921.
53. A. Kesarwani, S. K. Pandey, S. Sarkar, and A. Venkateswarlu. Recursive MDS matrices over finite commutative rings. *Discrete Applied Mathematics*, 304:384–396, 2021.
54. A. Kipnis and A. Shamir. Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization. In *Advances in Cryptology - CRYPTO 1999*, volume 1666 of *LNCS*, pages 19–30. Springer, 1999.
55. L. R. Knudsen. Truncated and Higher Order Differentials. In *Fast Software Encryption - FSE 1994*, volume 1008 of *LNCS*, pages 196–211. Springer, 1994.
56. L. R. Knudsen and D. A. Wagner. Integral Cryptanalysis. In *Fast Software Encryption – FSE 2002*, volume 2365 of *LNCS*, pages 112–127. Springer, 2002.

57. N. Koti, M. Pancholi, A. Patra, and A. Suresh. SWIFT: Super-fast and Robust Privacy-Preserving Machine Learning. In *USENIX Security Symposium*, pages 2651–2668, 2021.
58. X. Lai. *Higher Order Derivatives and Differential Cryptanalysis*. Springer US, 1994.
59. G. Leander, M. A. Abdelraheem, H. AlKhzaimi, and E. Zenner. A Cryptanalysis of PRINTcipher: The Invariant Subspace Attack. In *Advances in Cryptology - CRYPTO 2011*, volume 6841 of *LNCS*, pages 206–221. Springer, 2011.
60. G. Leander, B. Minaud, and S. Rønjom. A Generic Approach to Invariant Subspace Attacks: Cryptanalysis of Robin, iSCREAM and Zorro. In *Advances in Cryptology - EUROCRYPT 2015*, volume 9056 of *LNCS*, pages 254–283. Springer, 2015.
61. M. Matsui. Linear Cryptanalysis Method for DES Cipher. In *Advances in Cryptology - EUROCRYPT 1993*, volume 765 of *LNCS*, pages 386–397. Springer, 1993.
62. P. Mohassel and P. Rindal. ABY3: A Mixed Protocol Framework for Machine Learning. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 35–52, 2018.
63. National Institute of Standards and Technology. FIPS-46: Data Encryption Standard (DES), 1999. https://csrc.nist.gov/csrc/media/publications/fips/46/3/archive/1999-10-25/documents/fips46-3.pdf.
64. R. L. Rivest. Permutation Polynomials Modulo 2^w. *Finite Fields and Their Applications*, 2001(7):287–292, 2001.
65. C. J. Shallue and I. M. Wanless. Permutation polynomials and orthomorphism polynomials of degree six. *Finite Fields Their Appl.*, 20:84–92, 2013.
66. R. P. Singh and S. Maity. Permutation Polynomials modulo p^n. Cryptology ePrint Archive, Paper 2009/393, 2009. https://eprint.iacr.org/2009/393.
67. D. Singmaster. On Polynomial Functions (mod m). *Journal of Number Theory*, 6(5):345–352, 1974.
68. N. Smart. Bootstrapping for dummies. Zama Research Blog, 2022. https://www.zama.ai/post/what-is-bootstrapping-homomorphic-encryption.
69. V. Strassen. Gaussian Elimination is not Optimal. *Numerische mathematik*, 13(4):354–356, 1969.
70. N. N. Vasiliev and O. Kanzheleva. Polynomial Interpolation over the Residue Rings $\mathbb{Z}_n$. *Journal of Mathematical Sciences*, 209(6):845 – 850, 2015.
71. J. von zur Gathen and S. Hartlieb. Factoring Modular Polynomials. *Journal of Symbolic Computation*, 26:583–606, 1998.
72. S. Wagh, S. Tople, F. Benhamouda, E. Kushilevitz, P. Mittal, and T. Rabin. Falcon: Honest-majority maliciously secure framework for private deep learning. *Proceedings on Privacy Enhancing Technologies*, 2021(1):188–208, 2021.
73. D. A. Wagner. The Boomerang Attack. In *Fast Software Encryption – FSE 1999*, volume 1636 of *LNCS*, pages 156–170. Springer, 1999.
74. Y. Yu and M. Wang. Permutation Polynomials and Their Differential Properties over Residue Class Rings. Cryptology ePrint Archive, Paper 2013/251, 2013. https://eprint.iacr.org/2013/251.
75. Y. Zheng, T. Matsumoto, and H. Imai. On the Construction of Block Ciphers Provably Secure and Not Relying on Any Unproved Hypotheses. In *Advances in Cryptology - CRYPTO 1989*, volume 435 of *LNCS*, pages 461–480. Springer, 1989.

Table 6: Notations

$q = p_1^{y_1} \cdots p_a^{y_a}$	the prime factorization of an integer q
$D_{\alpha q}$	Gaussian distribution with squared variance αq
m	a factor of q sufficient to distinguish $D_{\alpha q}$ from random
d	degree of a polynomial
n	the number of elements in a Rubato state
$\mathbf{k}$	Rubato secret key
nc	Rubato nonce
M_v	fixed matrix in Rubato
r	number of Rubato rounds
ℓ	number of key stream elements output from one application of Rubato
λ	security parameter for Rubato
$\mathbf{e}, e_i$	noise sampled from $D_{\alpha q}$
$\mathbf{z}, z_i$	Rubato key stream
$\mathbf{w}, w_i$	output of $\mathrm{Tr}_{n,\ell}$, before adding noise
f	a factor of q/m
$\boldsymbol{\mu}$	Rubato plaintext
Δ	scaling factor for FHE scheme
s	number of known key stream elements used by attacker in Rubato attack
ω	linear algebra constant
t	threshold for identifying indices where $e_i \mod q = 0$
$G(x)$	the function $G(x) = \frac{1}{\alpha q} \cdot e^{-x^2/2\sigma^2}$
$C_\cdot$	complexity for doing $\cdot$
$\rho = \rho(q)$	The smallest integer such that $\rho! \equiv 0 \mod q$

SUPPLEMENTARY MATERIAL

A Notations

B Proof of Lemma 1

Proof of Lemma 1. i) Let u be a divisor of q and write $x_i = \hat{x}_i + x_i' u$. Any univariate monomial can then be written as $x_i^d = (\hat{x}_i + x_i' u)^d \equiv \hat{x}_i^d \mod u$. More generally, any multivariate monomial can then be written as $x_1^{d_1} x_2^{d_2} \dots x_n^{d_n} \equiv \hat{x}_1^{d_1} \hat{x}_2^{d_2} \dots \hat{x}_n^{d_n} \mod u$. The result now follows from the linearity of monomials.

Statement ii) is a direct consequence of i). □

C Comparison of Gröbner Basis Computation

This appendix presents a qualitative comparison of the running time and memory usage of Gröbner basis computations over $\mathbb{Z}_{p^y}$ and $\mathbb{F}_{p^y}$. In each experiment we randomly generate m quadratic polynomials in n variables, i.e., the coefficients of all monomials of degree ≤ 2 are sampled using the in-built random function over $\mathbb{Z}_{p^y}$ or $\mathbb{F}_{p^y}$. A random n-tuple $\mathbf{a}$ is picked as a common solution to the polynomials $f_1(\mathbf{x}), \dots, f_m(\mathbf{x})$ by considering the system $f_i(\mathbf{x}) - f_i(\mathbf{a}) = 0$, $1 \leq$

$i \leq m$. All systems are overdetermined with $\mathbf{a}$ as a unique solution, and the resulting Gröbner basis is always $\{x_1 - a_1, \ldots, x_n - a_n\}$. The time and memory usage given in Table 7 are from running the built-in F_4 routine with the computer algebra system Magma V2.27-1 [21].

Table 7: Time and Memory Consumption of Gröbner Basis Computation over $\mathbb{Z}_{p^y}$ and $\mathbb{F}_{p^y}$.

p^y	m	n	Time $\mathbb{F}_{p^y}$ (s)	Memory $\mathbb{F}_{p^y}$ (MB)	Time $\mathbb{Z}_{p^y}$ (s)	Memory $\mathbb{Z}_{p^y}$ (MB)
3^2	40	20	30.6	202	1107.4	974
2^2	40	20	4.0	302	196.6	470
5^3	60	20	2.1	67	42.2	101
97^2	120	30	611.2	317	3784.7	1291
2^{15}	120	30	150.7	702	4102.2	1376

D Details on Higher-Order Differential over $\mathbb{Z}_q$

Recall from Section 3.4 that we are interested in characterizing the sums $\sum_{x \in \mathbb{Z}_q} x^i \mod q$. To this end, we introduce some notation. Let $p_1, p_2, \ldots, p_a$ be distinct primes, and $\mu_1, \mu_1', \mu_2, \mu_2', \ldots, \mu_a, \mu_a' \in \mathbb{Z}$ integers that satisfy the Bézout identities:

$$\forall j \in \{1, 2, \ldots, a\} : \qquad \mu_j \cdot \frac{q}{p_j} + \mu_j' \cdot p_j = 1 \,. \tag{7}$$

For all $j \in \{1, 2, \ldots, a\}$, we define $\alpha_j \in \mathbb{F}_{p_j}$ as

$$\alpha_j := -\frac{q}{p_j} \mod p_j \,.$$

Finally, for each $i > 0$ and for each $j \in \{1, 2, \ldots, a\}$, we denote

$$\delta_{i,j} := \begin{cases} 0 & \text{if } i \not\equiv 0 \bmod (p_j - 1) \\ 1 & \text{otherwise} \end{cases} ,$$

and define $\delta_{0,j} = 0$.

Proposition 2. *Consider $q = p_1 p_2 \cdots p_a$, with $p_1, p_2, \ldots, p_a$ distinct primes, and let $\mu_i, \alpha_i, \delta_{i,j}$ be as defined above. Then, for all $i \geq 0$, we have*

$$\sum_{x \in \mathbb{Z}_q} x^i \mod q = \sum_{j=1}^{a} \left(\alpha_j \cdot \delta_{i,j} \cdot \mu_j \cdot \frac{q}{p_j} \right) \mod q \,.$$

Proof. If $i = 0$, the result holds since $\sum_{x \in \mathbb{Z}_q} x^0 = q \equiv 0 \mod q$.

Next, we consider $i \geq 1$, and compute $\sum_{x \in \mathbb{Z}_q} x^i \mod p$, where p is any of the prime divisor of q. Using Lemma 1, we have:

$$\sum_{x \in \mathbb{Z}_q} x^i \mod p \equiv \left(\sum_{x \in \mathbb{Z}_q} (x \mod p)^i \right) \mod p \equiv (q/p) \cdot \sum_{x \in \mathbb{Z}_p} x^i \mod p$$
$$\equiv \begin{cases} 0 & \text{if } i \not\equiv 0 \mod (p-1) \\ -q/p \mod p & \text{otherwise} \end{cases},$$

where in the last equality we have used that $\sum_{x \in \mathbb{Z}_p} x^i \equiv 0 \mod p$ if $i \not\equiv 0 \mod (p-1)$ by [14, Proposition 1], and $\sum_{x \in \mathbb{Z}_p} x^{p-1} \equiv -1 \mod p$ by Fermat's little theorem. We therefore have

$$\sum_{x \in \mathbb{Z}_q} x^i \equiv \alpha_j \cdot \delta_{i,j} \mod p_j,$$

for any $1 \leq j \leq a$ and $i \geq 0$. Since $p_1, p_2, \ldots, p_a$ are coprime, we can apply the Chinese Remainder Theorem 1 and derive the result:

$$\sum_{x \in \mathbb{Z}_q} x^i \mod q = \sum_{j=1}^{a} \left(\alpha_j \cdot \delta_{i,j} \cdot \mu_j \cdot \frac{q}{p_j} \right) \mod q$$

□

Based on this proposition, the following result follows immediately:

Corollary 1. *Consider $q = p_1 p_2 \cdots p_a$, with $p_1, p_2, \ldots, p_a$ distinct primes, and let $F : \mathbb{Z}_q \to \mathbb{Z}_q$ be given by $F(x) = \sum_{i=0}^{d} \varphi_i \cdot x^i$. Then*

$$\sum_{x \in \mathbb{Z}_q} F(x) \mod q \equiv \sum_{i=0}^{d} \varphi_i \cdot \sum_{j=1}^{a} \left(\alpha_j \cdot \delta_{i,j} \cdot \mu_j \cdot \frac{q}{p_j} \right) \mod q$$

where α_j, $\delta_{i,j}$, and μ_j are defined as for Proposition 2.

Depending on the composition of q, we note that $\delta_{i,j}$ can be zero for most values of i, and the sum $\sum_{x \in \mathbb{Z}_q} F(x) \mod q$ will only depend on relatively few coefficients φ_i. Still, the sum is generally non-zero and cannot be readily used in a higher-order differential attack unless the relevant coefficients φ_i are known. Since these coefficients typically depend on a secret key, we conclude that it is unlikely that the sum $\sum_{x \in \mathbb{Z}_q} F(x) \mod q$ can be exploited.

While we do not have an exact characterization of $\sum_{x \in \mathbb{Z}_q} F(x)$ when q has prime power divisors, we recall that the small-scale experiments mentioned in Section 3.4 over $\mathbb{Z}_{2^v}$ suggest that we do not get a zero-sum in this case either.

Remark 1. For completeness, we point out that the integral/square attack [56] (also based on the zero-sum property) works over rings in the same way as it works over field. However, we remark that it is based on a completely different property (related to the invertibility of the rounds functions) than the one exploited in a higher-order differential attack.

E Designing the Linear Layer for Symmetric Primitives over $\mathbb{Z}_q$

Here we briefly discuss the design approach for the linear layer in symmetric primitives over $\mathbb{Z}_q$. As in the field case, the simplest and most obvious idea is to define the linear layer as a multiplication by a matrix over $\mathbb{Z}_q^n$. We remark that, even if the majority of the works in the literature focus on the construction of matrices with particular statistical properties (e.g., a given branch number) over fields, some works have faced the problem of finding matrices with particular properties for the ring case. We refer to [53] for a concrete example.

Now we discuss the special case $q = p^y$. In this case, a possible idea is to define the linear transformation as a matrix multiplication over $\mathbb{F}_p^{y \cdot n}$. That is, given an element x over $\mathbb{Z}_{p^y}^n$:

1. first, rewrite x as an element over $\mathbb{F}_p^{y \cdot n}$;
2. then, apply the matrix multiplication with a matrix in $\mathbb{F}_p^{(y \cdot n) \times (y \cdot n)}$;
3. finally, rewrite the result as an element of $\mathbb{Z}_{p^y}^n$.

The main advantage of this approach is the possibility to work with matrices over fields. In such a case, the resulting scheme would be unaligned or weak-arranged following [20,26], in the sense that the linear layer and the S-box layer are defined over two different fields/rings. We refer to [20,26] for a complete discussion of the advantages and disadvantages of such schemes with respect to the ones of aligned or strong-arranged schemes, which is out the scope of this paper.

F Details of the "Cut and Sew" Design Strategy

We present a possible concrete construction of an S-box that does not admit a polynomial representation modulo $p_1 \cdot p_2$. From the setup in Section 4.3 we need to show that Eq. (4) does not hold modulo p_1 for some choice of (x, y, z). This corresponds to

$$\begin{aligned} &\left(S_2\left(\left\lfloor \frac{x \cdot p_1 + y}{p_2} \right\rfloor\right) \cdot p_2 + S_1\Big((x \cdot p_1 + y) \mod p_2\Big)\right) \mod p_1 = \\ &\left(S_2\left(\left\lfloor \frac{z \cdot p_1 + y}{p_2} \right\rfloor\right) \cdot p_2 + S_1\Big((z \cdot p_1 + y) \mod p_2\Big)\right) \mod p_1 . \end{aligned}$$

If there exists at least one tuple $x, z \in \{0, 1, \ldots, p_2 - 1\}$ and $y \in \{0, 1, \ldots, p_1 - 1\}$ for which such equality does *not* hold, then one can conclude that S does not admit a polynomial representation.

A possible way to construct S is given in the following Lemma.

Lemma 3. *Let p_1 and p_2 be two distinct primes such that $p_1 > p_2$, and let $q = p_1 \cdot p_2$. Let S_2 be the identity function over $\mathbb{F}_{p_1}$, and let S_1 be an orthomorphism over $\mathbb{F}_{p_2}$, that is, both $x \mapsto S_1(x)$ and $x \mapsto S_1(x) - x$ are invertible. Then*

the function S over $\mathbb{Z}_q$ defined as $S(x \equiv x_2 \cdot p_2 + x_1) = x_2 \cdot p_2 + S_1(x_1)$ for $x_1 \in \{0, 1, \ldots, p_2 - 1\}$ and $x_2 \in \{0, 1, \ldots, p_1 - 1\}$ does not admit a polynomial representation.

Proof. As we saw in Section 4.3, it is sufficient to prove that there exists a tuple $x, z \in \{0, 1, \ldots, p_2 - 1\}$ and $y \in \{0, 1, \ldots, p_1 - 1\}$ such that Eq. (4) for $i = 1$ is not verified. By simple computation, note that:

$$\begin{aligned}\left\lfloor \frac{x \cdot p_1 + y}{p_2} \right\rfloor \cdot p_2 &= x \cdot p_1 + y - ((x \cdot p_1 + y) \mod p_2) \\ &= (y - ((x \cdot p_1 + y) \mod p_2)) \mod p_1 \,,\end{aligned}$$

and similar for $\left\lfloor \frac{z \cdot p_1 + y}{p_2} \right\rfloor \cdot p_2$. Based on the previous considerations, it follows that Eq. (4) reduces to

$$(-x' + S_1(x')) \mod p_1 = (-z' + S_1(z')) \mod p_1$$

where $x' := (x \cdot p_1 + y) \mod p_2$ and $z' := (z \cdot p_1 + y) \mod p_2$. Since the function $x \mapsto -x + S_1(x)$ is invertible (by definition of orthomorphism), then such an equality is never satisfied for $x' \neq z'$ (remember that $p_1 > p_2$, so the final modulo reduction does not have any effect), which implies that S does not admit a polynomial representation. □

In particular, we point out that

- the identity map is the best choice from the implementation point of view, since it costs nothing;
- no condition is imposed on the orthomorphism. We refer to [5,65] for an analysis of orthomorphisms over $\mathbb{F}_p$.

G Equivalent Representation of Rubato

For completeness, we point out that – in the case in which *the linear layers are invertible* – an equivalent and simplified representation of Rubato is possible. In particular, recall that the final round of Rubato is slightly different from the rest, as a second final linear layer is applied. Here we show that it is possible to modify the description of Rubato such that a unique round function is used.

In the original description, the round function is defined by first applying a linear layer. However, since the linear layers are fixed (and invertible by assumption), and since no constraint is imposed on the inputs, it is possible to remove the initial linear layers of the stream cipher (in other words, it does not provide any additional security). Each round function (including the final one) can then be simply re-defined as

$$\mathrm{RF}'[k, i] = \mathrm{ARK}'[k, i] \circ \mathrm{MixRows} \circ \mathrm{MixColumns} \circ \mathrm{Feistel},$$

where for each $j \in \{0, 1, \ldots, r\}$:

$$\mathrm{ARK}'[k, i+j](x) = \begin{cases} x + \mathrm{MixRows} \circ \mathrm{MixColumns}(k \bullet rc_{i+j}) & \text{if } j \in \{0, 1, \ldots, r-1\}, \\ \mathrm{ARK}[k, i+r] & \text{if } j = r. \end{cases}$$

Then, the r-round stream cipher Rubato can be equivalently described as

$$\mathsf{Rubato}[k, \mathrm{nc}, i] = \mathrm{AGN} \circ \mathrm{Tr}_{n,\ell} \circ \mathrm{RF}'[k, i+r] \circ \cdots \circ \mathrm{RF}'[k, i+1] \circ \mathrm{ARK}'[k, i].$$

H Invertible and MDS Linear Layers for Rubato

As we have seen in Section 5, there are $985,818 \approx 2^{19.91}$ prime q values with bit-length 25 and $1,893,374 \approx 2^{20.85}$ with bit-length 26 (in total $2,879,192 \approx 2^{21.46}$, which is a fraction 5.72% of all q's of 25 and 26 bits) such that M_v ($v = 4, 6, 8$) are always invertible and MDS.

In the following, we further identify the conditions for q's such that M_v matrices are non-invertible and non-MDS.

Non-invertible conditions for q. According to the determinant of the matrix M_v for $v = 4, 6, 8$:

$$\det(M_4) = -35, \quad \det(M_6) = 6480 = 2^4 \cdot 3^4 \cdot 5, \quad \det(M_8) = 161875 = 5^4 \cdot 7 \cdot 37,$$

we prepare the set of small prime factors for each of them as:

$$S_4^{\mathrm{inv}} = \{-35\}, \quad S_6^{\mathrm{inv}} = \{2, 3, 5\}, \quad S_8^{\mathrm{inv}} = \{5, 7, 37\}.$$

Note that depending on the q values, there might be several small prime factors in S_4^{inv}. If any of the elements $s \in S_v^{\mathrm{inv}}$ are such that $\gcd(s, q) > 1$, then the matrix M_v is not invertible.

Non-MDS conditions for q. We compute the determinants of submatrices of M_v and get the corresponding set of small prime factors as follows:

$S_4^{\mathrm{mds}} = \{2, 3, 5, 7, 11, 17, -35\}$

$S_6^{\mathrm{mds}} = \{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 61, 67, 89, 97, 107, 109, 157\}$

$S_8^{\mathrm{mds}} = \{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89,$
$97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191,$
$193, 197, 199, 223, 227, 229, 233, 239, 241, 251, 257, 269, 271, 281, 293, 311, 313, 317, 331,$
$337, 347, 349, 359, 367, 373, 383, 401, 409, 421, 431, 433, 439, 443, 449, 457, 463, 491, 523,$
$541, 571, 607, 613, 631, 641, 647, 659, 673, 677, 683, 733, 743, 751, 773, 821, 839, 853, 857,$
$881, 883, 887, 907, 937, 967, 983, 1009, 1051, 1117, 1151, 1259, 1367, 1493, 1511, 1523, 1567,$
$1999, 2099, 2719, 2789, 3319, 3457, 3461\}$.

Similarly, depending on the q's, the element $-35 \in S_4^{\mathrm{mds}}$ might lead to several small prime factors, which makes the size of S_4^{mds} a variable. On the contrary, the size of S_6^{mds} and S_8^{mds} is deterministic: 22 and 124 respectively. If any of the elements $s \in S_v^{\mathrm{mds}}$ are such that $\gcd(s, q) > 1$, then the matrix M_v is not an MDS matrix.

I Pseudo-Code of Rubato Attack

Algorithm 1 gives a pseudo-code for the complete key recovery attack on Rubato presented in Section 6. The notation introduced earlier applies here as well.

Algorithm 1 Key-Recovery Attack on Rubato

Require: Key stream $\mathbf{z} \in \mathbb{Z}_q^s$ generated by Rubato, where
$m_{\min} \leq m \leq m_{\max}$,
$f = \prod_{i=1}^{b} f_i$,
$f_i \leq m$ for $i = 1, \ldots, b$,
$mf | q$, and
$mf \geq t$.

Ensure: The secret key $\mathbf{k}$ used to generate $\mathbf{z}$.

```
maxSc ← 0
for each k̃ ∈ Z_m^n do
    w̃ ← Ru_m[k̃]
    if Sc(k̃) > maxSc then
        maxSc ← Sc(k̃)
        k^(0) ← k̃
        ẽ ← (z̃ − w̃) mod m
    end if
end for
                                                    ▷ k^(0) = k mod m
for i = 1, ..., b do
    maxSc ← 0
    for v ∈ Z_{f_i}^n do
        k̃ ← k^(i−1) + (m ∏_{a=1}^{i−1} f_a) · v
        w̃ ← Ru_{m ∏_{a=1}^{i} f_a}[k̃]
        if Sc(k̃) > maxSc then
            maxSc ← Sc(k̃)
            k^(i) ← k̃
            ẽ ← (z̃ − w̃) mod m ∏_{a=1}^{i} f_a
        end if
    end for
end for
    ▷ mf sufficiently large that all |ẽ_i| < mf for ẽ = (ẽ_1, ..., ẽ_s) with probability at
least 99%
I ← {i | ẽ_i = 0}
j ← 1
for i ∈ I do
    F_i(k_1, ..., k_n) ← polynomial expressing w_i generated by Ru_q[k]
    A[j] ← coefficient vector of F_i
    u[j] ← z_i
    j ← j + 1
end for
x ← solution of Ax = u mod q
k ← values in x for the monomials k_1, ..., k_n.
```

Article II

5.2 The Algebraic FreeLunch: Efficient Gröbner Basis Attacks Against Arithmetization-Oriented Primitives

Augustin Bariant, Aurélien Boeuf, Axel Lemoine, Irati Manterola Ayala, Morten Øygarden, Léo Perrin, and Håvard Raddum

Published in *Advances in Cryptology – CRYPTO 2024, Lecture Notes in Computer Science (LNCS)*, vol 14923, pages 139–173. Presented at *CRYPTO* in August 2024.
https://doi.org/10.1007/978-3-031-68385-5_5

Version reproduced here in Cryptology ePrint Archive, Report 2024/347, 2024, with minor corrections.
https://eprint.iacr.org/2024/347

The Algebraic FreeLunch: Efficient Gröbner Basis Attacks Against Arithmetization-Oriented Primitives

Augustin Bariant[1,2], Aurélien Boeuf[2], Axel Lemoine[2,4], Irati Manterola Ayala[3]
Morten Øygarden[3], Léo Perrin[2], and Håvard Raddum[3]

[1] ANSSI, Paris, France
[2] Inria, Paris, France
[3] Simula UiB, Bergen, Norway
[4] DGA, France
{irati,morten.oygarden,haavardr}@simula.no
{augustin.bariant,aurelien.boeuf,axel.lemoine,leo.perrin}@inria.fr

Abstract. In this paper, we present a new type of algebraic attack that applies to many recent arithmetization-oriented families of permutations, such as those used in `Griffin`, `Anemoi`, `ArionHash`, and `XHash8`, whose security relies on the hardness of the constrained-input constrained-output (CICO) problem. We refer to the attack as the FreeLunch approach: the monomial ordering is chosen so that the natural polynomial system encoding the CICO problem already *is* a Gröbner basis. In addition, we present a new dedicated resolution algorithm for FreeLunch systems of complexity lower than current state-of-the-art resolution algorithms.
We show that the FreeLunch approach challenges the security of full-round instances of `Anemoi`, `Arion` and `Griffin`, and we experimentally confirm these theoretical results. In particular, combining the FreeLunch attack with a new technique to bypass 3 rounds of `Griffin`, we recover a CICO solution for 7 out of 10 rounds of `Griffin` in less than four hours on one core of AMD EPYC 7352 (2.3GHz).

Keywords: Algebraic attacks · Gröbner basis · FreeLunch · Symmetric cryptanalysis · `Griffin` · `Arion` · `Anemoi`

1 Introduction

Recent decades have seen the emergence of new directions in symmetric cryptography. The aim of symmetric primitives has always been to provide strong security guarantees along with stringent performance requirements. For instance, modern AES [1] implementations are able to process gigabytes of data in seconds. This has not changed, but the nature of the performance constraints considered is evolving. While the focus has traditionally been on software and hardware footprints, new use cases bring an entirely different set of relevant metrics.

Numerous such new settings exist, such as Homomorphic Encryption-friendly ciphers [17,18,41,20,8,36], ciphers designed to run efficiently in Multi Party Computation protocols [4,3,23], block ciphers defined over modular rings enabling

more efficient masking [40], and Arithmetization-Oriented [5] Permutations (AOP) operating on vectors over large field elements to better integrate with modern Zero-Knowledge proof systems. Despite their differences, there are general trends in the design of these primitives, which we group under a broad umbrella: Symmetric Techniques for Advanced Protocols (STAP).

One noticeable change the STAPs have brought is the underlying alphabet on which these primitives operate. Until recently, the overwhelming majority of symmetric primitives were designed based on operations either on the vector space $(\mathbb{F}_2)^n$, or using arithmetic over small binary fields of size 2^m, for $3 \leq m \leq 9$. For STAP, the used mathematical structures can be fields of characteristic 2, but with a much larger size (typically $m \geq 128$), large fields of prime characteristic (say, $p > 2^{128}$), or possibly not even a field (like Elisabeth [20] or Rubato [36]).

Moreover, the performance metrics now primarily involve the number of multiplications in the underlying structure. Some schemes are designed to have a low *multiplicative depth*, that is, few multiplications in sequence on any data path from input to output, while others are designed to merely have a low number of multiplications in total. These performance metrics have led designers to propose schemes that are light on multiplications but compensate by defining the multiplication operations over large fields. The hope is that schemes constructed this way may still be secure.

Resurgence of Algebraic Attacks. Despite its functionality, the constraints on multiplications significantly impact the security analysis of STAP primitives. Key-recovery attacks can often be simplified to the resolution of a (system of) non-linear equation(s). While this general approach has been applied successfully to $\mathbb{F}_2$-based stream ciphers, traditional block ciphers and hash functions have mostly been resistant to algebraic attacks.

However, since the constraints that STAP primitives must meet often affect their algebraic structure, algebraic attacks are one of the main threats to them. After an initial security analysis, most designers end up setting the number of rounds specifically so as to prevent algebraic attacks. Unfortunately, this is not always sufficient: to attack Jarvis [5], Albrecht *et al.* managed to re-write the equations considered by the designers in a simpler way, which made the resolution step much more efficient than initially thought [2]. When considering FHE- or MPC-oriented stream ciphers, FLIP [41] and its descendant Elisabeth [20] both fell [24,31] to linearization-based attacks: the system of equations to be solved ended up being simple enough that linear algebra-based approaches could be used.

Despite their crucial impact (particularly on setting the number of rounds), algebraic attacks against symmetric primitives are not nearly as well understood as classical attacks. In particular, there is no consensus among designers on how to provide solid and convincing arguments for the security of primitives against algebraic attacks.

Principles of an Algebraic Attack. Let us describe the main phases involved in the setup of an algebraic attack against a symmetric cipher. We provide a more thorough presentation in Section 2.

First, the problem of interest to the cryptanalyst is modelled as a system of polynomial equations: state variables are chosen, and equations linking them in a way that captures the round function constraints are defined.

Second, this system of equations is solved, using one of a few existing strategies. If the system can be represented as a unique univariate equation, an efficient FFT-based algorithm can be used to retrieve its roots, as performed in [10,9]. If, on the other hand, the system consists of several non-linear equations, one may need to convert the system into a form that can be solved efficiently.

A common strategy, adopted in this paper, consists in deriving a univariate polynomial equation that shares its solution with the original system, and in efficiently solving it afterwards. This is typically done via the computation of a *Gröbner basis* [21] for the system, using an algorithm such as F_4 [29], or its follow-up F_5 [30]. The Gröbner basis is then modified using a *change of order* algorithm, such as FGLM [28] or its variants [26,27,11], yielding a univariate polynomial equation sharing its solutions with the original system.

In the past, cipher designers have adopted different approaches to prevent such an attack: the authors of `Griffin` bounded the complexity of an algebraic attack with an estimate of the theoretical cost of F_5, but the authors of `Arion` [44] chose instead to bound the complexity of the change of order step[5].

Our Contributions. In this paper, we present a new type of algebraic attack that significantly outperforms all known methods. It requires revisiting all steps of the general approach outlined above: the encoding, the Gröbner basis computation and the change of order steps are different. First, using a custom monomial ordering and a specific equation generation procedure, we get the Gröbner basis for free. Combining this with known techniques is already sufficient to attack a few full-round instances of `Arion` and `Griffin` with a complexity lower than the security claim. We then further decrease the complexity of our attacks by improving the "reordering" step: we provide a novel and more efficient algorithm of our own design[6] that, in practice, significantly outperforms FGLM for our systems of equations. Unfortunately, the precise complexity analysis of one of the steps of this algorithm has remained beyond our reach. Nevertheless, we performed thorough experiments, implementing the full attack against several round-reduced primitives—we published the code used to verify these results on GitHub.[7] From these experimental results we can extrapolate the complexity of the dominating step, and get attack complexities as low as 2^{64}, 2^{98}, and 2^{118} for the weakest variants of `Griffin`, `Arion`, and `Anemoi`, respectively, where all

[5] This seems to be a choice of pragmatism, as it seems easier to get tight bounds; nothing in their experiments suggests that F_5 is faster.

[6] If the solving steps were meals of the day, the lunch would be free, hence FreeLunch.

[7] https://github.com/aurelbof/algebraic-freelunch

claim 128 bits of security. We can therefore confidently claim to shave off tens of bits of security from some full-round `Griffin` instances.

Outline of this Paper. We recall the necessary background on algebraic attacks based on Gröbner bases in Section 2. Our main contribution, the FreeLunch method, is introduced in a generic fashion in Section 3, and we show how it can be successfully applied to various primitives in Section 4. While it is a priori not possible to obtain a Gröbner basis for free in some cases, we show in Section 5 that it may still be possible to derive one at a negligible cost and apply this finding to the AOP `Anemoi`. We conclude in Section 6 with a discussion on the impact of the FreeLunch approach in terms *e.g.* of design.

2 Algebraic Background

We consider the case of an algebraic attack against a permutation intended for sponge [12] use. In this section, we will walk through the detailed inner workings of such an attack: it will allow us to introduce all the necessary mathematical background and state-of-the-art methods and to set the stage for our attacks by both providing all the theoretical tools we need and allowing to highlight the advantages of our method. Throughout the paper, we denote the base field by $\mathbb{F}$. Depending on context, this notion includes both $\mathbb{F}_p$ for a prime p and/or $\mathbb{F}_{2^n}$. We will also use the notions of polynomials and polynomial mappings interchangeably.

2.1 From an Attack to a System of Equations

We first present the constrained-input constrained-output (CICO) problem, that we will focus on solving. It was initially proposed by the designers of Keccak [35] as a crucial problem for estimating the security of permutations used in sponge constructions. It can also be seen as a variant of the limited birthday problem [32]. A natural instance in the context of algebraic cryptanalysis may be stated in the following form.

Problem 1 (CICO problem). Let $F : \mathbb{F}^t \longrightarrow \mathbb{F}^t$ be a permutation and $1 \leq \ell < t$ an integer. The goal is to find $x \in \{0\}^\ell \times \mathbb{F}^{t-\ell}$ such that $F(x) \in \{0\}^\ell \times \mathbb{F}^{t-\ell}$.

In this paper we will focus on the case $\ell = 1$. An attacker able to solve the CICO problem has control over both the input and output of the permutation, which is precisely what a good permutation is supposed to prevent. Furthermore, in our case, the value 0 in the output could be replaced *e.g.*, by a digest d to immediately obtain a preimage attack.

Encoding. To solve a CICO instance, we need to *model* or *encode* the problem into a system of polynomial equations. In symmetric cryptography, this is usually done iteratively by modelling one round after another, possibly adding new variables to keep the degree of the initial system low. The main challenge is

encoding the cipher's non-linear operations in a form that may be amenable for cryptanalysis. For STAP ciphers, the existence of low-degree models is usually possible by design: while such systems can be leveraged for cryptanalysis, they are also required for a fast verification in many ZK protocols.

Consider a non-linear function $S : \mathbb{F}^t \to \mathbb{F}^t$, and let $S_0, ..., S_{t-1}$ be its coordinate functions. A trivial model of such a function would consist of t equations of the form $y_i = S_i(x_0, ..., x_{t-1})$. In this case, a tuple $(x_0, ..., x_{t-1}, y_0, ..., y_{t-1})$ is a solution of the system if and only if we indeed have $y = S(x)$.

This situation corresponds to the simplest case, where the model is simply the evaluation of the function. However, more sophisticated models can exist, as first pointed out by the authors of Rescue [5,45]. Indeed, the non-linear layer of this permutation involves both $x \mapsto x^\alpha$ and $x \mapsto x^{1/\alpha}$, where α is a small integer. In this case, even though a non-linear function has a very high degree ($1/\alpha$ being a dense integer of $\mathbb{Z}/(p-1)\mathbb{Z}$), it is possible to design a low-degree model by using the equation $x = y^\alpha$ rather than $y = x^{1/\alpha}$.

More generally, all that is needed from a model is that it describes the graph of S, i.e., the set $\Gamma_S = \{(x, y) \mid (x, y) \in (\mathbb{F}^t)^2, y = S(x)\}$. In what follows, we focus on models corresponding to a system of multivariate polynomials $P = \{p_0, \ldots, p_{n-1}\} \subset \mathbb{F}[x_0, \ldots, x_{n-1}]$, which is associated with the following system of polynomial equations:

$$p_i(x_0, \ldots, x_{n-1}) = 0 \qquad 1 \leq i \leq n-1 \ . \tag{1}$$

The polynomial systems we consider have a finite number of solutions in the algebraic closure of $\mathbb{F}$, and are such that there exists a solution of P which directly leads to a solution of an associated CICO problem (or to a preimage of a given hash). In the remainder of this paper, we call `sysGen` the procedure used to generate a system of equation.

If $n = 1$, then P contains a unique equation of degree D in one variable, and we can use univariate techniques to solve the problem. In practice, such an attack needs a number of operations given by $\mathcal{O}\left((D(\log(D) + \log(p))\log(\log(D))\right)$ (see [10] for more details). We call the function returning the root(s) of a univariate polynomial `uniSol`.

2.2 Finding Structures in a System of Equations

Once our attack is represented by a system of equations, we need to solve it. To this end, we need to better understand the structures implied by a system of polynomial equations. Indeed, in order to solve the system, we need to somehow derive equations sharing the solutions of the original system, and whose solutions can be computed in practice. We thus need to formally describe the set of multivariate polynomials that have the roots we are interested in.

This set of polynomials is in fact an ideal $I = \langle P \rangle$, contained in the ring $R = \mathbb{F}[x_0, \ldots, x_{n-1}]$. We denote d_i the degree of equation p_i, and D_I the ideal degree of I, *i.e.* the number of solutions of P in the algebraic closure of $\mathbb{F}^n$, counted with multiplicity. In order to study this ideal, we need the notion of *monomial order*.

Definition 1. *A **monomial order** $\prec$ is a total order on the set of monomials of R such that i) for any monomial $m \in R$ we have $1 \prec m$; and ii) for any three monomials $m_1, m_2, t \in R$ we have*

$$m_1 \prec m_2 \implies t \cdot m_1 \prec t \cdot m_2 \ .$$

The typical monomial orders used in computations are the *lexicographical (lex)* order and the *graded reverse lexicographical (grevlex)* order. We now define a particular *weighted* order which we will use throughout the paper.

Definition 2. *Consider a weight vector $\boldsymbol{w} = (w_0, \ldots, w_{n-1}) \in \mathbb{R}^n$, where $w_0 \neq 0$. We say that $\boldsymbol{w}$ is **associated with the monomial order** $\prec$, defined by:*

$$\prod_{i=0}^{n-1} x_i^{\alpha_i} \prec \prod_{i=0}^{n-1} x_i^{\beta_i} \iff \begin{cases} \displaystyle\sum_{i=0}^{n-1} w_i\alpha_i < \sum_{i=0}^{n-1} w_i\beta_i \\ \qquad \text{or} \\ \exists k, \quad \displaystyle\sum_{i=0}^{n-1} w_i\alpha_i = \sum_{i=0}^{n-1} w_i\beta_i, \ \forall j > k, \alpha_j = \beta_j \text{ and } \alpha_k < \beta_k. \end{cases}$$

Technically speaking, this defines a weighted *graded lexicographical (deglex)* order with $x_0 \prec x_1 \prec \ldots \prec x_{n-1}$. The particularities of this choice will be needed in Section 5.

Definition 3. *The **leading monomial** of a nonzero polynomial $f \in R$, relative to a monomial order $\prec$, is the largest monomial contained in f according to $\prec$. It is denoted $\mathrm{LM}(f)$. The **leading coefficient** of f, $\mathrm{LC}(f)$, is the coefficient associated with $\mathrm{LM}(f)$. Finally, the **leading term** of f, $\mathrm{LT}(f)$, is the product of its leading monomial and coefficient.*

If $S = \{f_1, f_2, \ldots\} \subseteq R$, then we can extend the above definitions to the set S, *e.g.*, $LT(S) = \{LT(f_1), LT(f_2), \ldots\}$. We may now define the notion of a Gröbner basis of an ideal of R.

Definition 4 (Gröbner basis [15]). *Let I be an ideal of R. A finite set of polynomials $G \subset I$ is a **Gröbner basis** with respect to $\prec$ if the leading monomial of every polynomial in I is a multiple of the leading monomial of some polynomial in G. A Gröbner basis G is said to be **reduced** if for all $g \in G$, no monomial in g is divisible by an element of $\mathrm{LT}(G) \setminus \{\mathrm{LT}(g)\}$ and $\mathrm{LC}(G) = \{1\}$.*

An ideal I always contains a Gröbner basis. For a fixed $\prec$ there are usually many Gröbner bases, but only one reduced Gröbner bases. We will crucially rely on the following results throughout this paper.

Proposition 1 ([21, Chapter 2,§9, Prop 4 and Thm 3]). *Let G be a set of polynomials of R, $G = \{g_1, ..., g_m\}$. If the leading monomials of g_i and g_j are relatively prime for all $1 \leq i \neq j \leq m$, then G is a Gröbner basis for $\langle G \rangle$.*

Proposition 2 ([21, Chapter 2 §6 Prop 1]). *Let I be an ideal, $\prec$ a monomial order, G a Gröbner basis of I w.r.t. $\prec$, and $f \in R$. There exists a unique $r \in R$ such that:*

- $\mathrm{LT}(r)$ *is not divisible by any element of* $\mathrm{LT}(G)$.
- $\exists\ g \in I$, *such that* $f = g + r$.

The polynomial r is called the ***remainder*** *or* ***normal form*** *of f w.r.t. I and $\prec$.*

2.3 Exploiting a Gröbner Basis

The characteristics of a Gröbner basis can vary greatly depending on the underlying monomial order. For our goal, that is finding solutions of P, one typically wants to compute a Gröbner basis of $\langle P \rangle$ in the *lex* order. However, computing a Gröbner basis directly in this order tends to be computationally expensive. Instead, it is common to first compute a Gröbner basis in the *grevlex* order – which tends to be significantly faster – and then apply a dedicated order-changing algorithm, such as FGLM or its variants [28,26,27,43], to finally recover a basis in lex order. While we do not give a complete description of these algorithms, we nevertheless highlight some of the principles underpinning them, as they will be needed later for our own resolution algorithm.

The Quotient Ring. Thanks to Proposition 2, we can define the quotient ring R/I, where each class has a unique representative r such that $\mathrm{LT}(r)$ is not divisible by any element of a $\mathrm{LT}(I)$. The monomial order $\prec$ does not affect the quotient ring R/I, but determines the representative of each class. Macaulay's theorem [25, Theorem 15.3] states that the set of monomials in $R \setminus \mathrm{LT}(I)$ form a basis for R/I.

Definition 5. *Let I be an ideal of R. We say that I is* zero-dimensional *if* $\dim_{\mathbb{F}}(R/I)$ *is finite. In this case, its* ideal degree D_I *is* $\dim_{\mathbb{F}}(R/I)$.

The quotient ring R/I has a canonical basis with respect to $\prec$ denoted $\mathcal{B}_\prec(R/I)$, where the basis elements are given by all the monomials in R that are not in the ideal $\langle \mathrm{LM}(G) \rangle$, for G a Gröbner basis for I. If I is zero-dimensional, we have $|\mathcal{B}_\prec(R/I)| = D_I$. Each element r of R/I can then be written as a vector in the basis $\mathcal{B}_\prec(R/I)$, which we will call $\mathrm{NormalForm}(r)$. This allows us to define the linear matrix $T_j : R/I \to R/I$ corresponding to the multiplication by x_j.

Definition 6 (Multiplication matrix of x_j). *The* ***multiplication matrix*** *T_j of x_j relative to a zero-dimensional ideal I, a monomial order $\prec$, and the basis $\mathcal{B}_\prec(R/I) = (\epsilon_1, \ldots, \epsilon_{D_I})$ is defined as the square matrix which has each column defined as $C_i = \epsilon_i \times x_j$ represented in the basis $\mathcal{B}_\prec(R/I)$.*

Said differently, T_0 is the $D_I \times D_I$ matrix mapping the basis elements $\epsilon_i \mapsto \mathrm{NormalForm}(x_0 \epsilon_i)$. The following result is well-known in the literature, but we have not been able to find a reference that holds for finite fields (*e.g.*, it is derived as Corollary 4.6 in [22] over $\mathbb{C}$). For completeness, we provide a short proof that works over any field.

Proposition 3. *Let I be a zero-dimensional ideal of R, $\prec$ a monomial order, and T_0 the multiplication matrix of the variable x_0 with respect to $\prec$. We have $\det(x_0\boldsymbol{I} - T_0) \in I$, where $\boldsymbol{I}$ is the identity matrix.*

Proof. Let $C(x) = \det(x\boldsymbol{I} - T_0) = \sum_{i=0}^{D_I} c_i x^i$ be the characteristic polynomial of T_0. By the Cayley-Hamilton theorem we have $C(T_0) = \sum_{i=0}^{D_I} c_i T_0^i = \mathbf{0}$, where $\mathbf{0}$ is the $D_I \times D_I$ zero-matrix. Letting ϵ denote the column vector representing the constant polynomial 1 in R/I, we then have $C(T_0)\epsilon = 0$. As $T_0^i\epsilon$ is the representation of $\text{NormalForm}(x_0^i)$, this implies that $\text{NormalForm}(C(x_0)) = 0$, hence $C(x_0) \in I$. □

The next definition is a very standard and common hypothesis for an ideal when implementing Gröbner basis polynomial solving algorithms.

Definition 7 ([26], Definition 3.1). *An ideal I of R is in **shape position** if its reduced Gröbner basis in the lexicographical order has the following form*

$$G = \{f_0(x_0),\ x_1 - f_1(x_0),\ \ldots,\ x_{n-1} - f_{n-1}(x_0)\}.$$

In this case, it follows straightforwardly that $D_I = \deg(f_0)$, and the cost of finding solutions for I, given its lexicographic Gröbner basis, reduces to the problem of finding the roots of f_0.

3 The Algebraic FreeLunch

In this section, we present our own custom approach to algebraic attacks, which is applicable to solve CICO instances for some arithmetization-oriented permutations (see Section 4). As with the algebraic attacks we presented in the previous section, we first need to describe our problem using a system of polynomial equations. We start with a description of the general form of systems for which a Gröbner basis can be obtained for free (see Section 3.1). Then, we show how to deduce a univariate polynomial from this system in Section 3.2. Means to create these polynomial systems for various primitives are discussed in Sections 3.3 and 3.4, where we reuse and generalize an encoding technique introduced by the authors of `Griffin` in a way that can be applied to iterated functions. The entire solving strategy is summarized in Section 3.5.

3.1 FreeLunch Systems

We saw in Proposition 1 that there is a class of polynomial systems that admits a simple Gröbner basis. This is the motivation for the following definition.

Definition 8 (FreeLunch System). *Let R be the ring $\mathbb{F}[x_0, \ldots, x_{n-1}]$ and $P = \{p_0, \ldots, p_{n-1}\}$ be a sequence of polynomials of R. We say that P is a **FreeLunch system** if there exists a monomial order $\prec$ and integers $(\alpha_0, \ldots, \alpha_{n-1})$ such that for all $i \in \{0, \ldots, n-1\}$, $\text{LM}_\prec(p_i) = x_i^{\alpha_i}$. Any monomial order $\prec$ that verifies this property is said to be a FreeLunch order.*

Note that this is not the first time Proposition 1 has been used in cryptography. In [16], the authors describe a polynomial modeling for AES that can be said to be a FreeLunch system in a *graded lex* order. However, the ensuing change of order computation to a *lex* order is too costly to threaten the security of AES. The following properties are now easy to verify, and were also used in [16].

Proposition 4. *A FreeLunch system P is a Gröbner basis for the ideal $I = \langle P \rangle$ with respect to any of its FreeLunch orders. Moreover, I is zero-dimensional and of ideal degree $D_I = \prod_{i=0}^{n-1} \alpha_i$.*

Proof. The first statement follows directly from Proposition 1. For the latter statement, note that the canonical basis of R/I (w.r.t. a FreeLunch order $\prec$) is

$$\mathcal{B}_\prec(R/I) = \{x_0^{i_0} \cdots x_{n-1}^{i_{n-1}} \mid 0 \leq i_j < \alpha_j, \text{ for } 0 \leq j \leq n-1\}.$$

Counting all these basis elements yields D_I. □

We conceived a dedicated algorithm for the resolution of FreeLunch systems, which has a competitive time complexity. The following result will be proven in Section 3.2, where $2 \leq \omega \leq 3$ denotes the linear algebra exponent.

Theorem 1. *Given a FreeLunch system P, a FreeLunch order $\prec$, and the associated multiplication matrix T_0 of the variable x_0, there exists an algorithm to compute a solution for x_0 with time complexity*

$$\tilde{\mathcal{O}}\left(\alpha_0 \left(\prod_{i=1}^{n-1} \alpha_i\right)^\omega\right) .$$

3.2 Extracting a Univariate Equation from a FreeLunch System

We now turn to the problem of solving a FreeLunch system, with the aim of showing Theorem 1. As we will see in later sections, a FreeLunch system is typically only already a Gröbner basis under specially crafted monomial orders. To easily retrieve the solutions of a FreeLunch system, we look for a univariate polynomial belonging to the ideal spawned by the system. To do so, a common approach is to compute a Gröbner basis in the *lex* order. Given an initial Gröbner basis, computing a *lex* Gröbner basis can be performed using a *change of order* algorithm.

Existing change of order algorithms. The FGLM algorithm [28] provides an efficient method for changing the monomial orders of Gröbner bases of zero-dimensional ideals, with a running time of $\mathcal{O}(nD_I^3)$ operations and no conditions on the Gröbner bases, on the monomial order or on the ideal. Note that this cost includes computing the multiplication matrix T_0.

Later algorithms [26,27,43,11] significantly improve upon this running time, but require various assumptions on the input basis and underlying ideal. For

instance, [26,27,11] assume that the multiplication matrix T_0 is either given, or can be efficiently computed. Note that the latter is a consequence of the stability property (see [11, Definition 2.1]), which is assumed in some of these works. Unfortunately, FreeLunch systems do not generally satisfy this property. In fact, the authors of [11] state that when the base field is large enough and the ideal under consideration is radical, the stability property can be ensured through a generic linear change of coordinates. The issue is that doing so might transform the FreeLunch system into a different type of system that is not a Gröbner basis.

We briefly recall the effectiveness of the change of order algorithms assuming that T_0 is given and that the ideal is in shape position. In this case, the algorithm of [27] runs in $\mathcal{O}(D_I^\omega \log(D_I))$ and supposes that the input order is *grevlex* and the output order is *lex*. [43] runs in $\mathcal{O}(nD_I^\omega \log(D_I))$ with no additional hypothesis, and achieve $\mathcal{O}(D_I^\omega \log(D_I))$ when the ideal is in shape position. In our case, we are particularly interested in some algorithms that benefit from the sparsity of T_0, represented by its sparsity indicator t. The algorithm of [26, Theorem 3.2] for example runs in $\mathcal{O}(tD_I^2)$. The algorithm of [11] achieves an even better time complexity, of $\tilde{\mathcal{O}}(t^{\omega-1}D_I)$, if the input order is *grevlex* and the output order is *lex*. However, it is not clear to us if the ideas as presented in [11] can be directly generalized to our setting, *i.e.* with a weighted input monomial order, even if T_0 is given. Instead, we will develop a dedicated resolution algorithm from a basis in a FreeLunch order when T_0 is given, whose running time happens to coincide with that of [11] ($\tilde{\mathcal{O}}(t^{\omega-1}D_I)$).

A new approach. From the above discussion we see that while the original FGLM algorithm is applicable to the FreeLunch systems we are interested in, the improved variants are generally not. This motivated the design of a new dedicated algorithm for finding a univariate polynomial $f_0(x_0)$ belonging to the ideal, exploiting the sparsity of the multiplication matrix T_0.

Let I be a zero-dimensional ideal of $R = \mathbb{F}[x_0, \ldots, x_{n-1}]$, $\prec$ a monomial order giving a FreeLunch system, $\mathcal{B}_\prec = (\epsilon_1, \ldots, \epsilon_{D_I})$ the canonical basis of R/I, and T_0 the multiplication matrix corresponding to the variable x_0. Let H be the subspace of R/I containing the classes h of R/I where the unique representative of h with respect to $\prec$ does not contain the variable x_0. Let D_H be the dimension of H and $\mathcal{B}_\prec^H = [\phi_1, \ldots, \phi_{D_H}]$ be a canonical basis for the subspace H. It is clear that $\mathcal{B}_\prec^H$ exactly consists of the monomials m of $\mathcal{B}_\prec$ such that $x_0 \nmid m$. Thus, it holds that $D_H = \prod_{i=1}^{n-1} \alpha_i = D_I/\alpha_0$. We order the basis $\mathcal{B}_\prec$ specifically as

$$[\phi_1, \ldots, \phi_{D_H}, x_0\phi_1, \ldots, x_0\phi_{D_H}, x_0^2\phi_1, \ldots, x_0^2\phi_{D_H}, \ldots, x_0^{\alpha_0-1}\phi_1, \ldots, x_0^{\alpha_0-1}\phi_{D_H}],$$

and identify any polynomial $f \in R/I$ with its coefficient vector v_f of length D_I. The coefficient vector for the polynomial $x_0 f \in R/I$ can then be computed as a matrix/vector multiplication $T_0 v_f^\top$ for a fixed matrix T_0. The following lemma gives the structure of T_0.

Lemma 1. *Under the basis $\mathcal{B}_\prec$, the matrix T_0 is of the following form, represented as a block matrix with block sizes $D_H \times D_H$:*

$$T_0 = \begin{pmatrix} 0 & 0 & \dots & 0 & -M_0 \\ \boldsymbol{I} & 0 & \dots & 0 & -M_1 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & \boldsymbol{I} & -M_{\alpha_0-1} \end{pmatrix}.$$

The block matrices $M_0, \dots, M_{\alpha_0-1}$ are a representation of the reduction of $x_0^{\alpha_0}\mathcal{B}_\prec^H$ modulo I. The exact entries in the M_i matrices depend on the particular polynomials making up the Gröbner basis for the FreeLunch system. We call `matGen` the procedure which, given a basis $\mathcal{B}_\prec$, returns T_0.

From Proposition 3 it follows that $\det(x_0\boldsymbol{I}_{D_I} - T_0)$ is a univariate polynomial belonging to the ideal I. Computing this determinant and using a root-finding algorithm to solve $\det(x_0\boldsymbol{I}_{D_I} - T_0) = 0$ will finally give us a value for x_0 that solves the CICO problem. The following lemma shows that computing this determinant of this particularly structured matrix T_0 can be done with much lower complexity than for a generic matrix of dimension D_I.

Lemma 2. *Let $M_0, \dots, M_{\alpha_0-1}$ be the matrices defined in Lemma 1. We have*

$$\det(x_0\boldsymbol{I}_{D_I} - T_0) = \pm \det\left(x_0^{\alpha_0}\boldsymbol{I}_{D_H} + \sum_{i=0}^{\alpha_0-1} x_0^i M_i\right).$$

Proof.

$$\det(x_0\boldsymbol{I}_{D_I} - T_0) = \det\begin{pmatrix} x_0\boldsymbol{I} & 0 & \dots & 0 & M_0 \\ -\boldsymbol{I} & x_0\boldsymbol{I} & \dots & 0 & M_1 \\ \vdots & \ddots & \ddots & 0 & \vdots \\ 0 & 0 & -\boldsymbol{I} & x_0\boldsymbol{I} & M_{\alpha_0-2} \\ 0 & 0 & 0 & -\boldsymbol{I} & x_0\boldsymbol{I} + M_{\alpha_0-1} \end{pmatrix}.$$

The rows of this matrix can be split into a set of α_0 blocks of D_H rows each. Denote these blocks as $L_0, \dots, L_{\alpha_0-1}$ from top to bottom. We now do elementary row operations block-wise, from bottom to the top, with $L_i = L_i + x_0 L_{i+1}$, for $i = \alpha_0 - 2, \dots, 0$. This does not change the value of the determinant, and after these row operations, the resulting determinant to compute is:

$$\det\begin{pmatrix} 0 & 0 & \dots & 0 & x_0^{\alpha_0}\boldsymbol{I} + \sum_{i=0}^{\alpha_0-1} x_0^i M_i \\ -\boldsymbol{I} & 0 & \dots & 0 & x_0^{\alpha_0-1}\boldsymbol{I} + \sum_{i=0}^{\alpha_0-2} x_0^i M_{i+1} \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \dots & -\boldsymbol{I} & 0 & x_0^2\boldsymbol{I} + \sum_{i=0}^{1} x_0^i M_{i+\alpha_0-2} \\ 0 & \dots & 0 & -\boldsymbol{I} & x_0\boldsymbol{I} + M_{\alpha_0-1} \end{pmatrix}.$$

In this block matrix representation, the determinant of the full matrix is the determinant of the top right matrix, up to the sign $(-1)^{\alpha_0+1}$. □

Complexity Analysis. We call `polyDet` the procedure returning the polynomial $\det(x_0 \boldsymbol{I}_{D_I} - T_0)$ using Lemma 2. This step has a complexity $\tilde{\mathcal{O}}(D_I D_H^{\omega-1}) = \tilde{\mathcal{O}}(\alpha_0 D_H^{\omega})$ with the algorithm of [39]. Note that this is precisely the complexity that was obtained with the algorithm of [11] for systems satisfying the *stability* and *shape position* properties. In order to estimate the logarithmic factors in the complexity formula, we bound the complexity with [33, Theorem 4.4], using a polynomial matrix multiplication algorithm of complexity $\mathcal{O}(D_H^{\omega}\log(\alpha_0) + D_H^2 \log(\alpha_0)\log(\log(\alpha_0)))$ [19]. This way, we bound the number of operations of `polyDet` with (when D_H is large):

$$\mathcal{O}\left(\alpha_0 \log(\alpha_0)^2 D_H^{\omega} + \alpha_0 \log(\alpha_0)^2 \log(\log(\alpha_0)) D_H^2\right) \approx \mathcal{O}(\alpha_0 \log(\alpha_0)^2 D_H^{\omega}) \ . \quad (2)$$

The remaining task to show Theorem 1 is to recover the roots of a univariate polynomial of degree D_I, a step we refer to as `uniSol`. This costs $\tilde{\mathcal{O}}(D_I)$ operations and is thus negligible in comparison with the `polyDet` step.

We want to highlight that the complexity of the `matGen` step is hard to estimate precisely (recall that T_0 is assumed known in Theorem 1). This step can be upper bounded by $\mathcal{O}\left(nD_I^3\right)$ operations [28, Proposition 3.1] using the FGLM algorithm. However, this is likely to be a loose upper bound, as it does not take into account any of the underlying structure. Indeed, we have observed this in our experiments by naively taking some of this structure into account (see Appendix G). Still, as we will see later, the `matGen` step can sometimes be costlier than `polyDet`.

3.3 Ordering a FreeLunch

Having seen how to efficiently find solutions for FreeLunch systems, we will focus in the next two subsections on the problem of actually finding them. Recall that FreeLunch systems rely on the existence of specific monomial orders. How can we figure out if such an order exists (and thus, if a system is a FreeLunch)? In general, answering this question is not trivial. However, the systems we will be concerned with in Sections 4 and 5 naturally have a deeper structural property that allows for a procedural approach to this problem.

Definition 9 (Triangular System). *Let $P = (p_1, \ldots, p_{n-1}, g)$ be a polynomial system in $\mathbb{F}[x_0, x_1, \ldots, x_{n-1}]$. We say that P is a* ***triangular system*** *if there exists polynomials $q_0, q_1, \ldots, q_{n-1}$, integers $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$, and $c_0, \ldots, c_{n-1} \in \mathbb{F} \setminus \{0\}$ such that*

$$\begin{cases} p_i &= c_i x_i^{\alpha_i} + q_i(x_0, \ldots, x_{i-1}) \qquad \text{for } \ 1 \leq i \leq n-1 \ , \\ g &= c_0 x_0^{\alpha_0} + q_0(x_0, \ldots, x_{n-1}) \ . \end{cases}$$

A triangular system P can be assigned the following monomial order that is naturally motivated by the FreeLunch definition.

Construction 1 *For a triangular system P, we define its triangular order, $\prec_T$, as the monomial order from Def. 2 associated with the weight vector defined recursively by:*

$$\begin{cases} \mathrm{wt}(x_0) &= 1\ , \\ \mathrm{wt}(x_i) &= \mathrm{wt}\left(\mathrm{LM}_{\prec_T}\left(q_i(x_0, x_1, \ldots, x_{i-1})\right)\right)/\alpha_i \qquad \textit{for}\ \ 1 \leq i \leq n-1\ . \end{cases}$$

The recursion is well-defined since the leading monomial of q_i and its associated weight are only dependent on the weights of x_j for $j < i$. The definition ensures that $\mathrm{LM}_{\prec_T}(p_i) = x_i^{\alpha_i}$ for $1 \leq i \leq n-1$. Hence, a triangular system P is a FreeLunch system with respect to $\prec_T$ if the leading monomial of g is univariate in x_0, which gives the following Proposition:

Proposition 5 (Ordering a FreeLunch). *Let P be a triangular system, and $\prec_T$ be its triangular order. If $\alpha_0 > \mathrm{wt}(\mathrm{LM}_{\prec_T}(q_0(x_0, \ldots, x_{n-1})))$ then P is a FreeLunch system and $\prec_T$ is one of its FreeLunch orders.*

As we will see below, such systems naturally occur when investigating some cryptographic permutations.

3.4 FreeLunch Systems From Iterated Functions

The permutations we target share the same structure: a composition of a number of round functions. The input and output of every round is a state of t elements[8] from $\mathbb{F}$ and the round functions typically consist of a limited number of multiplications and α-th roots in $\mathbb{F}$. Writing them out directly as polynomial functions yields polynomials of high degree, owing to the α-th root operations. A natural modeling strategy introduces a new variable for each of them to keep the degree growth manageable, as $x = y^\alpha$ is of much lower degree than $y = x^{1/\alpha}$ when $\alpha \in \{3, 5, ..., 257\}$ and $|\mathbb{F}|$ is large. In this section, we take inspiration from an encoding suggested by the authors of `Griffin` [34] and show how to model this class of primitives as polynomials that form a low degree FreeLunch system.

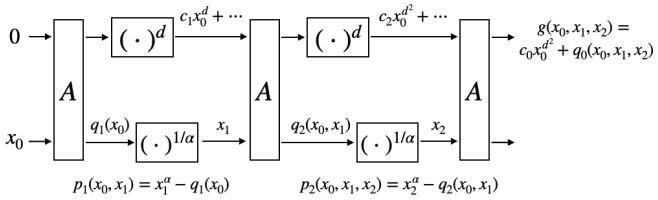


Fig. 1: Triangular system for a simple SPN with two branches and two rounds.

[8] We say that the permutation has t *branches*, or as we like to think of them, *brunches*.

Toy Example. Let us start with a toy SPN of two rounds, where the round function F is given by $F = S \circ A : \mathbb{F}^2 \to \mathbb{F}^2$, for an invertible affine layer A and a non-linear layer S. Moreover, we write $S = (S_1, S_2)$ where $S_1(y) = y^d$, for a small integer d, and $S_2(y) = y^{1/\alpha}$. This simple construction is shown in Figure 1, where we also label the branches with variables and polynomials at different points. We consider a CICO problem with input $(0, x_0)$.

As we assume $d \ll p$, we note that the round function F can only achieve a high degree as a polynomial function $\mathbb{F}^2 \to \mathbb{F}^2$ due to the map S_2. Thus, we introduce new variables x_1 and x_2 for the output of S_2 in the first and second rounds, respectively. The polynomials p_1, p_2 relate the symbolic input and output of the two S_2-functions, where q_1 is an affine polynomial in x_0 that is input to S_2 in the first round, and $q_2(x_0, x_1)$ is the input to S_2 in the second round and has degree d in the x_0-variable and degree 1 in the x_1-variable. Finally, we let $g(x_0, x_1, x_2)$ represent the first output of the construction that is required to be 0 by the CICO-problem. We can now write g as

$$g(x_0, x_1, x_2) = c_0 x_0^{d^2} + q_0(x_0, x_1, x_2),$$

for a suitable constant $c_0 \in \mathbb{F}$, and where $q_0(x_0, x_1, x_2)$ has degree $< d^2$ in x_0, degree d in x_1 and degree 1 in x_2. If $c_0 \neq 0$ we observe that $P = \{p_1, p_2, g\}$ forms a triangular system (Definition 9), whose solutions yield a solution to the specified CICO-problem. The weight vector of $\prec_T$ from Construction 1 is $(1, 1/\alpha, d/\alpha)$, and it is straightforward to verify that P satisfies the condition of Proposition 5. Hence, P is a FreeLunch system.

General Case. The above example shows the core idea for how a FreeLunch system can be made from a round function that relies on the functional inverse of low degree function to achieve a high degree. Let us generalize this insight. Let $F_i : \mathbb{F}^t \to \mathbb{F}^t$, $\boldsymbol{z}_{i-1} \mapsto \boldsymbol{z}_i$, denote the i-th round of a primitive, where $\boldsymbol{z}_{i-1} = (z_{i-1,0}, \ldots, z_{i-1,t-1})$ is the state after $i-1$ rounds. Recall that F_i may itself have a high degree (in $\boldsymbol{z}_{i-1}$), but suppose there exists a set of variables $\boldsymbol{x}_i = \{x_{i,0}, \ldots, x_{i,\ell_i-1}\}$ satisfying

$$x_{i,j}^{\alpha_{i,j}} = \mathcal{L}_{i,j}(\boldsymbol{z}_{i-1}), \text{ for } 0 \leq j < \ell_i, \tag{3}$$

where $\alpha_{i,j}$ is an integer and $\mathcal{L}_{i,j}$ an affine function. Moreover, suppose that there exists a polynomial function $G_i : \mathbb{F}^{t+\ell_i} \to \mathbb{F}^t$ of low degree d_i, satisfying

$$F_i(\boldsymbol{z}_{i-1}) = \{G_i(\boldsymbol{z}_{i-1}, \boldsymbol{x}_i) \mid \boldsymbol{x}_i \text{ satisfies (3)}\}. \tag{4}$$

In other words, while F_i and G_i are different as polynomial functions, they yield the same output when $\boldsymbol{x}_i$ is restricted by (3). For instance, in the toy example above, we used

$$G_1(\boldsymbol{z}_0, x_1) = \left((A_1(\boldsymbol{z}_0))^d, x_1\right), \qquad G_2(\boldsymbol{z}_1, x_2) = \left((A_1(\boldsymbol{z}_1))^d, x_2\right),$$

where A_1 denotes the first output of A.

Polynomial Modeling. We now have an iterated function of t branches where each round can be described using the functions $\mathcal{G} = \{G_1, \ldots, G_r\}$ satisfying (3) and (4), and where G_i is of degree d_i. We introduce the shorthand $d_{\leq i} = d_1 d_2 \cdots d_i$, and we require that the exponents d_i are small enough to ensure that their composition will not exceed the maximal degree determined by the finite field, *i.e.* $d_{\leq r} < |\mathbb{F}| - 1$.

With this in place, we give the following blueprint for constructing a polynomial system. Recall that we focus on the variant of the CICO-problem where a single input in $\mathbb{F}$ is unknown, which we will symbolically denote by x_0, and the output of the first branch should be 0. The initial state is written as $\boldsymbol{z}_0(x_0)$, which consists of t affine polynomials in x_0. The proceeding state is defined as $\boldsymbol{z}_1(x_0, \boldsymbol{x}_1) = G_1(\boldsymbol{z}_0, \boldsymbol{x}_1)$, where we note that $\boldsymbol{z}_1$ is now t polynomials of at most degree d_1 in the variables $x_0, \boldsymbol{x}_1$. Furthermore, we create functions $\boldsymbol{p}_1 = \{p_{1,0}, \ldots, p_{1,\ell_1 - 1}\}$ to encode the relations (3) that we encounter in this step. That is, for $\boldsymbol{x}_1 = \{x_{1,0}, \ldots, x_{1,\ell_1 - 1}\}$, we construct the polynomials

$$p_{1,j} = x_{1,j}^{\alpha_{1,j}} - \mathcal{L}_{1,j}(\boldsymbol{z}_0), \text{ for } 0 \leq j < \ell_1 \ .$$

This process of updating the state $\boldsymbol{z}_i$ and constructing polynomials[9] $\boldsymbol{p}_i$ is repeated for all rounds up to $r - 1$. In the last round, we generate polynomials $\boldsymbol{p}_r$ as before, but instead of updating the state, we compute the final polynomial

$$g(x_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_r) = [G_r(\boldsymbol{z}_{r-1}, \boldsymbol{x}_r)]_1,$$

where $[\cdot]_1$ means the first polynomial of $G_r(\boldsymbol{z}_{r-1}, \boldsymbol{x}_r)$. This construction yields the polynomial system $P_{\mathcal{G}} = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_r, g\}$ over the ring $\mathbb{F}[x_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_r]$.

***$P_{\mathcal{G}}$* as a FreeLunch system.** It is easy to verify that $P_{\mathcal{G}}$ is a triangular system if g contains a univariate monomial in x_0. In fact, this is a stronger case than the generic triangular systems considered in Section 3.3, since we are also able to bound the degrees of the polynomials in $P_{\mathcal{G}}$ by round degrees $d_1, \ldots, d_r$. This allows us to give an analogous variant of Proposition 5 for $P_{\mathcal{G}}$. Instead of a condition on the entire system that could be computationally expensive to verify, we reduce the assumption to the condition of a single monomial in g.

Proposition 6. *Let $P_{\mathcal{G}}$ be a polynomial system as constructed above, where all $\alpha_{i,j}$ from (3) are at least 2, and the functions $\mathcal{G} = \{G_1, \ldots, G_r\}$ are of degrees $d_1, \ldots, d_r \geq 2$. Then $P_{\mathcal{G}}$ is a FreeLunch system if g contains the monomial $x_0^{d_{\leq r}}$.*

Before proving the proposition, we start by defining $\prec_{\mathcal{G}}$, which is the monomial order from Definition 2 whose weight vector is given by

$$\begin{cases} \mathrm{wt}(x_0) &= 1 \ , \\ \mathrm{wt}(x_{i,j}) &= d_{\leq i-1}/\alpha_{i,j} \qquad \text{for } \ 1 \leq i \leq r \ \text{ and } \ 1 \leq j \leq \ell_i \ , \end{cases}$$

[9] If a single variable is introduced in a round, we will ease notation by writing $\boldsymbol{x}_i = x_i$, $\boldsymbol{p}_i = p_i$ and α_i.

where we define $d_{\leq 0} = 1$. Recall that $\boldsymbol{z}_i$ denotes the i-th state represented by t polynomials in $x_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_i$. We will write $\mathrm{wt}(\mathrm{LM}(\boldsymbol{z}_i))$ for the maximal weight among the monomials of these t polynomials.

Lemma 3. *Let $\boldsymbol{z}_i$ be the i-th state associated with a system $\mathcal{G}$ that satisfies the conditions of Proposition 6. Then the following inequality holds for $\prec_{\mathcal{G}}$:*

$$\mathrm{wt}\left(\mathrm{LM}(\boldsymbol{z}_i)\right) \leq d_{\leq i} \ .$$

Proof. We proceed by induction. The base case of $i = 0$ is immediate since $\boldsymbol{z}_0$ is affine in x_0, and $d_{\leq 0} = 1$ by definition. For the induction step, we recall that $\boldsymbol{z}_i = G_i(\boldsymbol{z}_{i-1}, \boldsymbol{x}_i)$, where G_i has degree d_i. Thus we have

$$\mathrm{wt}(\mathrm{LM}(\boldsymbol{z}_i)) \leq d_i \cdot \max\{\mathrm{wt}(\mathrm{LM}(\boldsymbol{z}_{i-1})), \mathrm{wt}(x_{i,1}), \ldots, \mathrm{wt}(x_{i,\ell_i})\} \ .$$

Now we have $\mathrm{wt}(x_{i,j}) < d_{\leq i-1}$, and $\mathrm{wt}(\mathrm{LM}(\boldsymbol{z}_{i-1})) \leq d_{\leq i-1}$ by the induction hypothesis. Hence

$$\mathrm{wt}(\mathrm{LM}(\boldsymbol{z}_i)) \leq d_i d_{\leq i-1} = d_{\leq i} \ . \qquad \square$$

The proof of this lemma also implies that $\prec_{\mathcal{G}}$ coincides with $\prec_T$ from Construction 1 if all functions $\mathcal{L}_{i,j}(\boldsymbol{z}_{i-1})$ achieve their maximal weight $d_{\leq i-1}$. We now have all we need to show Proposition 6.

Proof. (Proposition 6). From Lemma 3 we observe

$$\begin{aligned} \mathrm{wt}\left(x_{i,j}^{\alpha_{i,j}}\right) &= \alpha_{i,j}\mathrm{wt}\left(x_{i,j}\right) = d_{\leq i-1} \\ &\geq \mathrm{wt}(\mathrm{LM}(\boldsymbol{z}_{i-1})) \geq \mathrm{wt}(\mathrm{LM}(\mathcal{L}_{i,j}\left(\boldsymbol{z}_{i-1}\right))). \end{aligned}$$

Hence $\mathrm{LM}\left(f_{i,j}\right) = x_{i,j}^{\alpha_{i,j}}$. Moreover, Lemma 3 also guarantees that

$$\mathrm{wt}\left(\mathrm{LM}(g)\right) \leq \mathrm{wt}(\mathrm{LM}(\boldsymbol{z}_r)) \leq d_{\leq r}.$$

Due to the fact that $\alpha_{i,j} \geq 2$, the factor $1/\alpha_{i,j}$ that appears in the weight of all variables $\boldsymbol{x}_i$, $i \geq 1$, the above equality can only be achieved by the monomial $x_0^{d_{\leq r}}$. It then follows from the assumption that $\mathrm{LM}(g) = x_0^{d_{\leq r}}$, which makes $P_{\mathcal{G}}$ a FreeLunch system. $\square$

Computing a reduced Gröbner Basis for $\langle P_{\mathcal{G}} \rangle$ (`sysGen`). We have just seen that computing a Gröbner basis for a given FreeLunch system $P_{\mathcal{G}}$ is – as the name suggests – free. There are, however, two practical concerns worth addressing. Firstly, while $P_{\mathcal{G}}$ is itself a Gröbner basis, it is generally not the unique reduced Gröbner basis w.r.t. any of its FreeLunch orders. Secondly, generating the polynomials in $P_{\mathcal{G}}$ may itself be hard.

In practice we do not generate the polynomials in $P_{\mathcal{G}}$ in the direct manner outlined earlier. Rather, we will use the fact that $\boldsymbol{p}_i$ and g are constructed by composing certain round functions. This allows us to reduce by the polynomials $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{i-1}$ introduced earlier in the process in order to suppress the growth

of the number of monomials. This is detailed in Appendix A, where we show that when $P_{\mathcal{G}}$ satisfies the conditions of Proposition 6, the polynomial system generated in this manner is also a FreeLunch system that is the unique reduced Gröbner basis for $\langle P_{\mathcal{G}} \rangle$ w.r.t. $\prec_{\mathcal{G}}$. In Appendix A a complexity estimate for generating this latter polynomial system is also provided, under the assumption that reductions following every multiplication of multivariate polynomials can be done efficiently.

3.5 Summary of the FreeLunch Attack

The strategy of the attack presented in this section is summarized in Algorithm 1. The initial condition is that there exists a FreeLunch system associated

1. `sysGen`: Generate a FreeLunch system (Section 3.4).
2. `matGen`: Compute the multiplication matrix T_0 (Section 3.2).
3. `polyDet`: Compute $f(x_0) = \det\left(x_0^{\alpha_0} \boldsymbol{I}_{D_H} + \sum_{i=0}^{\alpha_0 - 1} x_0^i M_i\right)$ (Section 3.2).
4. `uniSol`: Solve $f(x_0) = 0$.

Algorithm 1: Overview of the FreeLunch Attack.

with the target primitives. Methods for constructing this FreeLunch system were presented in Section 3.3 and 3.4, and the complexities for `sysGen` using these methods are discussed in Appendix A. A different way of generating a FreeLunch system will also be shown in Section 5. We will estimate the complexity of `polyDet` by (2), but we do not have a clear estimate for `matGen`. The final step `uniSol` recovers the roots of a univariate polynomial of degree D_I. This costs $\tilde{\mathcal{O}}(D_I)$ operations and is thus negligible in comparison with the earlier steps. We expect the complexity of the attack as a whole to be dominated by either `matGen` or `polyDet` for the primitives we have investigated. This is in line with our experiments (see Section 6.1), where `matGen` seems to be the dominating step for larger instances.

The numbers for the complexity of the `polyDet` step in our attacks against several AOPs are shown in[10] Table 1. Details of how we obtained the complexities for the specific ciphers will be provided further in the paper. While we do not have a rigorous complexity estimate for the `matGen` step, recall that the complexity $\mathcal{O}(nD_I^3)$ of the FGLM algorithm serves as a loose upper bound. This upper bound is already sufficient to break a few instances of `Griffin` and α-`Arion`. For `Griffin` we have FGLM complexities of 2^{108} and 2^{122} for $t \geq 12$ and $\alpha = 3, 5$,

[10] The complexities correspond to the number of basic $\mathbb{F}_p$ operations; writing them as number of calls to the primitive would yield lower but hard to compute numbers.

respectively, and 2^{127} for $t = 8$ and $\alpha = 3$. For α-Arion with $\alpha = 121$ and $e = 3$ we get 2^{117} and 2^{127} for $t = 4, 5$ and for $e = 5$ we get complexities of 2^{114} and 2^{124} for $t = 3, 4$.

Name	α/e	Number of branches						
		2	3	4	5	6	8	≥ 12
Griffin	3	$\emptyset$	120 (16)	112 (15)	$\emptyset$	$\emptyset$	76 (11)	64 (10)
	5	$\emptyset$	141 (14)	110 (11)	$\emptyset$	$\emptyset$	81 (9)	74 (9)
Arion	3	$\emptyset$	128 (6)	134 (6)	114 (5)	119 (5)	98 (4)	$\emptyset$
	5	$\emptyset$	132 (6)	113 (5)	118 (5)	122 (5)	101 (4)	$\emptyset$
α-Arion	3	$\emptyset$	104 (5)	84 (4)	88 (4)	92 (4)	98 (4)	$\emptyset$
	5	$\emptyset$	83 (4)	87 (4)	91 (4)	94 (4)	101 (4)	$\emptyset$
Anemoi	3	118 (21)	$\emptyset$	-	$\emptyset$	-	-	-
	5	156 (21)	$\emptyset$	-	$\emptyset$	-	-	-
	7	174 (20)	$\emptyset$	-	$\emptyset$	-	-	-
	11	198 (19)	$\emptyset$	-	$\emptyset$	-	-	-

Table 1: Time complexity ($\log_2$) of polyDet in FreeLunch-based attacks against some full-round algorithms (aiming at 128-bit security). Number of rounds in parentheses, $\emptyset$ corresponds to undefined algorithms. The α/e column reports α for Griffin and Anemoi; and e for the Arion variants.

4 Using FreeLunch Systems Directly

Experimental Verification In this section and in Section 5, we support theoretical attacks with practical experiments on reduced-round versions. All experiments are performed on 1 core of AMD EPYC 7352 (2.3GHz) with 250 GB of memory, and on $\mathbb{F}_p$ with $p =$ 0x64ec6dd0392073. The sysGen step is performed with SageMath [47], Magma [13] or the NTL [48] and Flint [37] libraries, the matGen step is performed with Flint, and the polyDet step is perfomed with the Polynomial Matrix Library [46,38].

4.1 A Detailed Example: Griffin

Specification of Griffin. Griffin [34] is a family of sponge hash and compression functions proposed by Grassi *et al.* at Crypto 2023 designed to be used in Zero-Knowledge applications. As such, it makes use of the internal permutation Griffin-π, which is defined over the finite field $\mathbb{F}$.

Each round function of Griffin-π is composed of a non-linear layer, the addition of a round constant, and a linear layer defined by multiplication by an MDS matrix.The specific features of Griffin impose that the primitive is only suitable for $\mathbb{F}^t$ where $t = 3$ or t is a multiple of four.

Definition 10 (Non-linear layer of Griffin-π). *Let $\alpha \in \{3, 5, 7, 11\}$ be the smallest integer such that $\gcd(\alpha, p-1) = 1$, $p > 2^{63}$ and let t be the number of*

branches. For $0 \leq i \leq t-1$, let $(\delta_i, \mu_i) \in \mathbb{F}^2 \setminus \{(0,0)\}$ be pairwise distinct such that $\delta_i^2 - 4\mu_i$ is a quadratic nonresidue modulo p. Then, the non-linear layer of `Griffin-`π *is $S(x_0, \ldots, x_{t-1}) = (y_0, \ldots, y_{t-1})$, where each y_i is defined by the equations:*

$$y_i := \begin{cases} x_0^{1/\alpha} & \text{if } i = 0 \\ x_1^{\alpha} & \text{if } i = 1 \\ x_2 \cdot (L_2(y_0, y_1, 0)^2 + \delta_2 \cdot L_2(y_0, y_1, 0) + \mu_2) & \text{if } i = 2 \\ x_i \cdot (L_i(y_0, y_1, x_{i-1})^2 + \delta_i \cdot L_i(y_0, y_1, x_{i-1}) + \mu_i) & \text{otherwise,} \end{cases}$$

for $L_i(z_0, z_1, z_2) = (i-1) \cdot z_0 + z_1 + z_2$.

Definition 11 (`Griffin-`π). *Let r be the number of rounds, and for $1 \leq i \leq r-1$ let $\boldsymbol{c}^{(i)} \in \mathbb{F}^t$ be a constant vector (we assume $\boldsymbol{c}^{(r)} = 0$). Then* `Griffin-`π *$\mathcal{G}^\pi : \mathbb{F}^t \to \mathbb{F}^t$ is defined as*

$$\mathcal{G}^\pi(\cdot) := \mathcal{F}_r \circ \cdots \circ \mathcal{F}_2 \circ \mathcal{F}_1(M \times \cdot),$$

where for $1 \leq i \leq r$, the i-th round function $\mathcal{F}_i$ is defined as

$$\mathcal{F}_i(\cdot) = \boldsymbol{c}^{(i)} + M \times S(\cdot),$$

for $M \in \mathbb{F}^{t \times t}$ a matrix, and S the non-linear layer of `Griffin-`π.

The first round function of `Griffin-`π for $t = 4$ is depicted in Fig. 2 where, to simplify the construction, we denote by F_i the last two equations of Definition 10. The authors proposed various instances with a 128-bit security claim. The number of branches varies from 3 to 24 (though not all values are possible), and the number of rounds is computed for different degrees α based on the complexity of finding a Gröbner basis using the basic encoding as it was the most efficient attack they could find.

FreeLunch system for `Griffin`. We observe that the round function of `Griffin` readily lends itself to a naive construction of the system $P_{\mathcal{G}}$, as described in Section 3.4. Indeed, for each round i we can simply define $\boldsymbol{z}_i = G_i(\boldsymbol{z}_{i-1}, x_i)$ by $\boldsymbol{z}_i = \boldsymbol{c}^{(i)} + M \times \boldsymbol{z}'_i$, where $z'_{i,0} = x_i$ and $z'_{i,j}$ given as y_j, for $1 \leq j < t$, in Definition 10 of the i-th round. Note that G_i will be of degree at most $d_i = 2\alpha + 1$. Under the assumption that the polynomial g in $P_{\mathcal{G}}$ satisfies the monomial property of Proposition 6, we get an associated ideal degree of $(\alpha\,(2\alpha+1))^r$.

Remark 1. Note that the naive modeling $P_{\mathcal{G}}$ given above for `Griffin` is not new; in fact, it was proposed by the authors of this algorithm for their initial security analysis [34, Section 6.2]. However, the authors did not attempt to compute a Gröbner basis for $\langle P_{\mathcal{G}} \rangle$ in a FreeLunch order, but rather in the usual *grevlex* order. They estimate that computing a Gröbner basis in this latter monomial order well exceeds the security level for the suggested number of rounds.

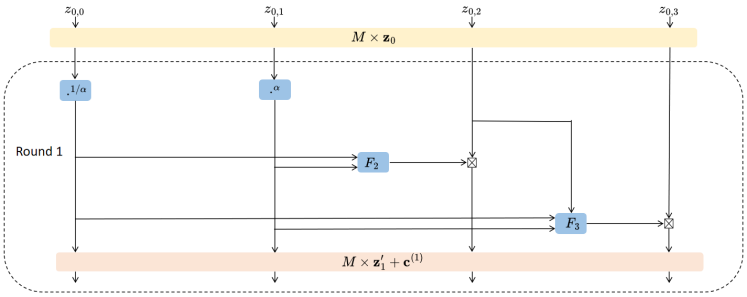


Fig. 2: First round function of `Griffin`-π with $t = 4$.

Bypassing Several Rounds. A further improvement is constructing an affine input in x_0 for the CICO problem that is tailored to bypass the inversion operation for a few initial rounds. This effectively means that fewer variables x_i are necessary, which in turn has a significant impact on the resulting ideal degree. Observe that bypassing inversions fits seamlessly with the machinery introduced in Section 3.4. The only difference is that we choose a different sequence of polynomial functions $\mathcal{G}^*$, where G_1^* effectively spans several rounds but only depends on $\boldsymbol{z}_0$. The ensuing functions G_i^*, $i \geq 2$, can still be constructed as described for the naive method above (though there will now be fewer of them). The exact number of initial rounds we can bypass will depend on t, where a larger t generally allows us to bypass more rounds[11]. All underlying details are given in Appendix B.

For $t = 3, 4$, we can bypass one round with linear functions in $\boldsymbol{z}_0$. For $t = 8$, we are able to bypass two rounds with cubic functions in $\boldsymbol{z}_0$, and three rounds can be bypassed with $\deg(\boldsymbol{z}_0) = 6\alpha + 3$ for $t \geq 12$. Assuming all systems satisfy Proposition 6, we get the following parameters:

$$D_{I,t} = \begin{cases} (\alpha\,(2\alpha+1))^{r-1}, & \text{for } t = 3, 4, \\ 3\,(\alpha\,(2\alpha+1))^{r-2}, & \text{for } t = 8, \\ (6\alpha+3)\,(\alpha\,(2\alpha+1))^{r-3}, & \text{for } t \geq 12\,. \end{cases} \tag{5}$$

$$D_{H,t} = \begin{cases} \alpha^{r-1}, & \text{for } t = 3, 4, \\ \alpha^{r-2}, & \text{for } t = 8, \\ \alpha^{r-3}, & \text{for } t \geq 12\,. \end{cases} \tag{6}$$

[11] A similar observation of bypassing rounds was already considered in [34, Section 6.2]. However, the authors only describe a method for bypassing a single round for $t = 3$ and do not consider the effect of having a larger t.

Table 2: Expected time complexity ($\log_2$) of `polyDet` for the different full-round instances of `Griffin`, where $\omega = 2.81$. Number of rounds in parentheses.

Branches	Complexity ($\log_2$)	
	$\alpha = 3$	$\alpha = 5$
3	120 (16)	141 (14)
4	112 (15)	110 (11)
8	76 (11)	81 (9)
12,16,20,24	64 (10)	74 (9)

Table 3: Experimental results on `Griffin` with $(t, \alpha) = (12, 3)$. `sysGen` uses Flint and NTL with the fast multivariate multiplication algorithm of Appendix A.

Number of rounds	Complexity of `polyDet`	Time (s)			Memory (MB)
		`sysGen`	`matGen`	`polyDet`	
5	26	0.17	0.02	0.53	14
6	34	4.0	6.67	50.78	471
7	41	2,558	3,361	5,727	27,600

Complexity Analysis and Experimental Results. We can now use the machinery described in Section 3.2 to solve the FreeLunch system for `Griffin`. As noted in Section 3.2, it is hard to theoretically estimate the complexity of `matGen` where one computes the multiplication matrix T_0. On the other hand, based on previous analysis, we estimate the complexity of `polyDet` by computing $D_{I,t}D_{H,t}^{\omega-1} = D_{I,t}(D_{I,t}/\alpha_0)^{\omega-1}$ for the different values of $D_{I,t}$. As a consequence, the running time for `polyDet` becomes

$$\tilde{\mathcal{O}}(D_{I,t}D_{H,t}^{\omega-1}) = \begin{cases} \tilde{\mathcal{O}}((\alpha^\omega\,(2\alpha+1))^{r-1}), & \text{for } t = 3, 4, \\ \tilde{\mathcal{O}}(3\,(\alpha^\omega\,(2\alpha+1))^{r-2}), & \text{for } t = 8, \\ \tilde{\mathcal{O}}((6\alpha+3)\,(\alpha^\omega\,(2\alpha+1))^{r-3}), & \text{for } t \geq 12\ . \end{cases} \tag{7}$$

The resulting estimated time complexities of running `polyDet` for the proposed instances of `Griffin` are listed in Table 2. Experimental results are presented in Table 3 and discussed in Section 6.1. One example of concrete input and output values solving the CICO problem for 7 rounds of `Griffin` can be found in Appendix C.

4.2 Applicability Beyond `Griffin`: the Example of `ArionHash`

Specification of `ArionHash`. `ArionHash` [44] is an arithmetization-oriented hash function proposed by Roy *et al.* that, much like `Griffin`, uses a permutation as its core primitive. Called `Arion`-π, this permutation utilizes in each round a polynomial of very high degree in one branch and low degree polynomials in the remaining branches to significantly decrease the number of necessary rounds to achieve the desired security.

Definition 12 (Non-linear layer of Arion-π). *Let $p \geq 5$ be a prime, t the number of branches, e the smallest positive integer be such that $\gcd(e, p-1) = 1$, and $121 \leq \alpha \leq 257$ an integer such that $\gcd(\alpha, p-1) = 1$.*

For $0 \leq i \leq t-2$, let $\delta_{i,1}, \delta_{i,2}, \mu_i \in \mathbb{F}_q$ be such that $g_i(x) = x^2 + \delta_{i,1} \cdot x + \delta_{i,2}$ is a quadratic function without zeroes in $\mathbb{F}_q$ and define $h_i(x) = x^2 + \mu_i \cdot x$. Then the non-linear layer of Arion-π *is $\mathcal{S} = \{f_0, \ldots, f_{t-1}\}$, where each f_i is defined "from-right-to-left" by the equations:*

$$f_{t-1}(y_0, \ldots, y_{t-1}) = y_{t-1}^{1/\alpha},$$
$$f_i(y_0, \ldots, y_{t-1}) = y_i^e \cdot g_i(\sigma_{i,t}) + h_i(\sigma_{i,t}), \qquad t-2 \geq i \geq 0,$$

where $\sigma_{i,t}$ represents the sum of all previously computed inputs and outputs

$$\sigma_{i,t} = \sum_{j=i+1}^{t-1} y_j + f_j(y_0, \ldots, y_{t-1}) \ .$$

Definition 13 (Arion-π). *Let r be the number of rounds, and for $1 \leq i \leq r$ let $\mathbf{c}_i \in \mathbb{F}^t$ be a constant vector. Then Arion-π is defined as the following composition over $\mathbb{F}^t$:*

$$\texttt{Arion-}\pi : (y_0, \ldots, y_{t-1}) \mapsto (L_{c_r} \circ \mathcal{S}_r) \circ \cdots \circ (L_{c_1} \circ \mathcal{S}_1) \circ L_0(y_0, \ldots, y_{t-1}),$$

where L_{c_i} is the affine map of [44, Definition 3] and $\mathcal{S}_i$ is the non-linear layer of Arion-π, for $1 \leq i \leq r$.

We illustrate the construction of the first round of Arion-π in Fig. 3 for $t = 4$ where, for the sake of clarity, we only represent the function f_i of the non-linear layer of Arionwithout the details of g_i and h_i.

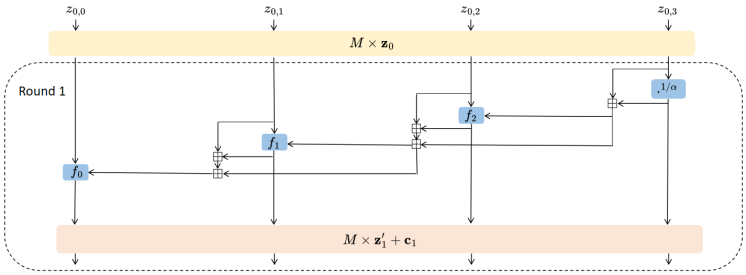


Fig. 3: First round function of Arion-π with $t = 4$.

We provide the parameters for Arion-π and ArionHash as well as for their additionally proposed aggressive versions α-Arion and α-ArionHash with $e = 3, 5$ and $\alpha = 121$ in Table 4 (number of rounds are in parenthesis). The authors claim 128-bit security for each parameter set.

	Arion-π & ArionHash		α-Arion& α-ArionHash	
Branches	Complexity ($\log_2$)		Complexity ($\log_2$)	
	$e = 3$	$e = 5$	$e = 3$	$e = 5$
3	128 (6)	132 (6)	104 (5)	83 (4)
4	134 (6)	113 (5)	84 (4)	87 (4)
5	114 (5)	118 (5)	88 (4)	91 (4)
6	119 (5)	122 (5)	92 (4)	94 (4)
8	98 (4)	101 (4)	98 (4)	101 (4)

Table 4: Expected time complexity ($\log_2$) of polyDet for the different full-round instances of ArionHash, where $\alpha = 121$ and $\omega = 2.81$. Number of rounds in parentheses.

FreeLunch system for ArionHash. Due to the similarities in construction between Arion-π and Griffin-π, it comes as no surprise that the round function of Arion-π also fits the naive construction of the system $P_{\mathcal{G}}$ described in Section 3.4. In this case, we start with a constrained input $\boldsymbol{z}_0$ depending linearly on a variable x_0, and for each round i we define $\boldsymbol{z}_i = G_i(\boldsymbol{z}_{i-1}, x_i)$ by $\boldsymbol{z}_i = L_{c_i}(\boldsymbol{z}'_i)$, where $z'_{i,t-1} = x_i$ and $z'_{i,j} = f_j(z_{i-1,0}, \ldots, z_{i-1,t-2}, x_i^{\alpha})$ for $t-2 \geq j \geq 0$. Note that each component of $\boldsymbol{z}'_i$ (and thus of $\boldsymbol{z}_i$) will have degree at most $d_i = (2^{t-1}(e+1)-e)^i$ in x_0. Assuming that the polynomial g in $P_{\mathcal{G}}$ satisfies the monomial property of Proposition 6, we get an associated ideal degree of $\left(\alpha\left(2^{t-1}(e+1)-e\right)\right)^r$.

In addition, one can further improve this technique by generating a set of input states constructed so that the inversion operation for the first round is bypassed, reducing the number of necessary variables and, consequently, the associated ideal degree. This is done analogously to Griffin, and all underlying details can be found in Appendix D. For Arion we are only able to bypass a single round with $\deg(\boldsymbol{z}_0) = 3e$, independent of t. Assuming all systems satisfy Proposition 6, we get the following parameters:

$$D_I = 3e\left(\alpha\left(2^{t-1}(e+1)-e\right)\right)^{r-1},$$
$$D_H = \alpha^{r-1}.$$

Complexity Analysis and Experimental Results. We can now apply the new methods introduced in Section 3.2 to solve the FreeLunch system for ArionHash. Based on the general complexity analysis of the attack, we list the estimated time complexities of polyDet for the different proposed ArionHash parameters in Table 4. Note that here $D_H = D_I/\alpha_0 = \alpha^{r-1}$, so that the running time for polyDet becomes

$$\tilde{\mathcal{O}}(D_I D_H^{\omega-1}) = \tilde{\mathcal{O}}\left(3e\left(\alpha^{\omega}\left(2^{t-1}(e+1)-e\right)\right)^{r-1}\right).$$

Experimental results are presented in Table 5 and discussed in Section 6.1.

Number of branches	Complexity of `polyDet`	Time (s) `sysGen`	`matGen`	`polyDet`	Memory (MB)
3	32	1.31	< 0.01	6.8	3,387
4	33	1.46	0.07	18.7	7,551
5	35	9.54	0.08	64.5	15,903
6	36	247	0.31	215	32,626
8	39	24,872	4.86	2,545	134,165

Table 5: Experimental results on 2-round `Arion`, with $(e, \alpha) = (3, 121)$. `sysGen` is performed using `SageMath`. `polyDet` uses an evaluation/interpolation algorithm of pml [46] since the algorithm of [39] implemented in pml does not work for the non-generic polynomial matrix in input of `polyDet`.

4.3 Last Example: `XHash8`

`XHash8` is a permutation proposed by Ashur, Kindi and Mahzoun in [6]. Along with `XHash12`, it is a follow-up of RPO [7], itself a follow-up of Rescue-Prime [45]. `XHash8` features a layer with inversion operations in eight out of twelve branches. Thus we need to introduce polynomials $\boldsymbol{p}_i = (p_{i,0}, \ldots, p_{i,7})$ and variables $\boldsymbol{x}_i = (x_{i,0}, \ldots, x_{i,7})$ in these layers. We can, by adjusting for this minor difference, directly define a FreeLunch system as we have seen in the previous two subsections. Since this is very similar to what we saw for `Griffin` and `Arion`, we will give the details for this analysis in Appendix F, and only limit ourselves to a short discussion of the highlights in the immediate following. We note that the related constructions XHash12, RPO and Rescue-Prime all contain a layer of inversion operations in all branches, and hence we cannot directly obtain a FreeLunch from them.

Complexity and Impact on Security Analysis. In contrast to the flexible constructions of `Griffin`-π and `Arion`-π, `XHash8` is only defined for a fixed prime $p \approx 2^{64}$ and a fixed sponge setting with state size $t = 12$, where the rate is 8 and capacity 4. Although this does not directly lend itself to a CICO problem with a single zero in input and output, we applied the FreeLunch approach to this setting. We generate a FreeLunch system with $\alpha_0 = 7^6$, $D_H = 7^{24}$, and $D_I = 7^{30}$, whose time complexity of `polyDet` is approximately 2^{214} for $\omega = 2.81$. As this is significantly higher than brute force for the chosen p, we conclude that `XHash8` seems very secure against the techniques presented in this paper.

That said, we note that the current security estimates for `XHash8` are (conservatively) extrapolated from scaled-down experiments with $t = 3$ using a single unknown input [6, App. B]. While the FreeLunch framework cannot currently be extrapolated in a similar manner for the full construction, we still hope it could provide a basis for future insights into the security of `XHash8`.

5 Forcing the Presence of a FreeLunch for Anemoi

We have just seen three examples where the FreeLunch machinery of Section 3.4 could be readily applied. Anemoi is another class of permutations that rely on the inverse of low degree monomials in a finite field to achieve a high degree and so it would, a-priori, seem like another candidate where we can apply the FreeLunch techniques. However, we will see that this is not as straightforward as it may appear because a direct application of the technique creates a polynomial system $P_{\mathcal{G}}$ where g does not satisfy the assumption of Proposition 6. Instead, we will show how to compute a modified polynomial system $P_{\mathcal{G}^*}$ that retains the valid solution to the CICO problem, which will turn out to be a FreeLunch system. This comes at the cost of a somewhat larger, yet still comparable, ideal degree than what was given in Conjecture 2 of [14]. We start by describing Anemoi.

Description of Anemoi. The Anemoi permutations [14] operate on $\mathbb{F}_q^{2\ell}$ for $\ell \geq 1$, and either $q = 2^n$ with n odd, or $q = p$ for any prime $p \geq 3$. There are differences between the operations for the odd and even characteristic cases that will impact our later modeling. Thus, we focus on the setting of $\ell = 1$ and p prime, leaving the even case as future work. In odd characteristic, Anemoi takes a parameter α such that $x \mapsto x^\alpha$ is a permutation of $\mathbb{F}_p$, usually $\alpha = 3$, 5, 7 or 11. The original paper gives two specific hash function instances based on Anemoi with $\ell = 1$: AnemoiSponge-BN-254, with a 254-bit prime p, AnemoiSponge-BLS12-381, with a 381-bit prime p. 127 bits of security are claimed for both of these.

Definition 14 (Odd Anemoi with $\ell = 1$). *For a given p, α and number of rounds r,* Anemoi *is a permutation of $\mathbb{F}_p^2$ defined as*

$$\texttt{Anemoi}_{p,\alpha,r}(x, y) = \mathcal{M} \circ \mathsf{R}_r \circ \cdots \circ \mathsf{R}_1(x, y).$$

For $1 \leq i \leq r$, the i-th round function R_i is defined as

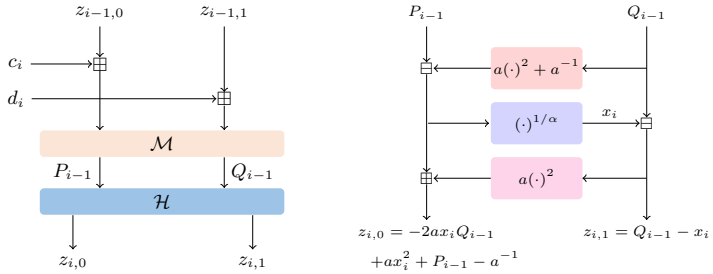
$$\mathsf{R}_i(x, y) = \mathcal{H} \circ \mathcal{M}(x + c_i, y + d_i) \qquad \text{and} \qquad \mathcal{M}(x, y) = (2x + y, x + y),$$

for constants $c_i, d_i \in \mathbb{F}_p$. $\mathcal{H}$ is the nonlinear operation over $\mathbb{F}_p^2$ that is described in Figure 4b for a non-zero constant $a \in \mathbb{F}$.

Failure of the Direct FreeLunch Approach. As a starting point, we consider the following slight modification[12] of the polynomial system $P_{\mathcal{G}}$ for Anemoi, for $1 \leq i \leq r$.

$$p_i(x_0, \ldots, x_i) = x_i^\alpha + aQ_{i-1}(x_0, \ldots, x_{i-1})^2 - P_{i-1}(x_0, \ldots, x_{i-1}) + a^{-1}, \quad (8)$$
$$g(x_0, \ldots, x_r) = P_r(x_0, \ldots, x_r)\ . \quad (9)$$

[12] The only difference from the description in Section 3.4 is that we allow $\mathcal{L}_{i,j}$ from (3) to be quadratic, due to the term Q_{i-1}^2.



(a) `Anemoi` round function. (b) $\mathcal{H}$ in odd characteristic.

Fig. 4: Description of `Anemoi` over prime fields with $\ell = 1$.

A first observation is that $z_{i,0}$ must have a larger leading term than $z_{i,1}$ under any monomial order. Since this leading term gets distributed to both branches under $\mathcal{M}$ (without the possibility of cancelling the leading term), we have $\mathrm{LM}(Q_i) = \mathrm{LM}(P_i)$. Now note from the output shown in Figure 4b that in the computation of $z_{i,0}$, the terms aQ_{i-1}^2 and $-aQ_{i-1}^2$ will both occur and cancel each other. Hence, the leading monomial of g must be either $x_r\mathrm{LM}(Q_{r-1})$ or x_r^2, so there is no possible choice of monomial order where g will have a leading monomial in only x_0.

In order to circumvent this issue, we will multiply g by suitable monomials in $x_1, \ldots, x_r$ that leads to a reduction by the polynomials $p_1, \ldots, p_r$. This process will ultimately lead to a new polynomial g^*, whose leading monomial will be univariate in x_0. To briefly illustrate the idea, we consider the first step of this procedure. Writing out g in terms of P_{r-1}, Q_{r-1} and x_r, we have

$$\begin{aligned} g(P_{r-1}, Q_{r-1}, x_r) &= (1 - 4ax_r)Q_{r-1} + (2ax_r - 1)x_r + 2(P_{r-1} - a^{-1}) \\ &= (-4aQ_{r-1} + 2ax_r - 1)x_r + Q_{r-1} + 2P_{r-1} - 2a^{-1} \ , \end{aligned}$$

taking into account the final $\mathcal{M}$-transformation. In order to cancel out the product x_rQ_{r-1} using p_r, we construct the following polynomial:

$$\begin{aligned} g' &= x_r^{\alpha-1}g + (4aQ_{r-1} - 2ax_r + 1)p_r \\ &= x_r^{\alpha}(-4aQ_{r-1} + 2ax_r - 1) + x_r^{\alpha-1}(Q_{r-1} + 2P_{r-1} - 2a^{-1}) \\ &\quad + (4aQ_{r-1} - 2ax_r + 1)(x_r^{\alpha} + aQ_{r-1}^2 - P_{r-1} + a^{-1}) \\ &= x_r^{\alpha-1}(Q_{r-1} + 2P_{r-1} - 2a^{-1}) + 4a^2Q_{r-1}^3 + aQ_{r-1}^2 \\ &\quad + 4Q_{r-1}(1 - aP_{r-1}) - P_{r-1} + a^{-1} - 2x_r(a^2Q_{r-1}^2 - aP_{r-1} + 1) \ . \end{aligned}$$

Hence, we have successfully eliminated x_r from the leading monomial of g' under any monomial order that satisfies

$$\begin{aligned}
\mathrm{wt}\left(\mathrm{LM}\left(Q_{r-1}^3\right)\right) &> \mathrm{wt}\left(\mathrm{LM}\left(x_r^{\alpha-1}Q_{r-1}\right)\right) \ , \\
\mathrm{wt}\left(\mathrm{LM}\left(Q_{r-1}^3\right)\right) &> \mathrm{wt}\left(\mathrm{LM}\left(x_r^{\alpha-1}P_{r-1}\right)\right) \ , \\
\mathrm{wt}\left(\mathrm{LM}\left(Q_{r-1}^3\right)\right) &> \mathrm{wt}\left(\mathrm{LM}\left(x_rQ_{r-1}^2\right)\right) \ , \\
\mathrm{wt}\left(\mathrm{LM}\left(Q_{r-1}^3\right)\right) &> \mathrm{wt}\left(\mathrm{LM}\left(x_rP_{r-1}\right)\right) \ ,
\end{aligned}$$

which, since $\alpha \geq 3$ and $\mathrm{LM}(P_{r-1}) = \mathrm{LM}(Q_{r-1})$, can be simplified further to:

$$\mathrm{wt}\left(\mathrm{LM}\left(Q_{r-1}^2\right)\right) > \mathrm{wt}\left(x_r^{\alpha-1}\right) = (\alpha-1)\mathrm{wt}(x_r) \ .$$

Constructing FreeLunch Systems From `Anemoi`. We now turn our attention to the general construction of g^* that will allow us to apply the FreeLunch machinery for solving the CICO problem for `Anemoi`. Here, we will not only be interested in the leading monomials of the intermediate states and p_i, but also in the second and third monomials. To this end, we define $\prec_A$ to be the monomial order associated with the weight vector defined recursively by

$$\begin{cases}
\mathrm{wt}(x_0) = 1, \\
\mathrm{wt}(x_i) = \dfrac{2}{\alpha}\mathrm{wt}(x_0 \cdots x_{i-1}), \text{ for } 1 \leq i \leq r \ .
\end{cases}$$

Indeed, this choice of monomial order allows us to prove the following two lemmas. For a polynomial h, we let $\mathrm{Mon}_j(h)$ denote the j-th monomial of h according to $\prec_A$. As usual, we write $\mathrm{LM}(h) = \mathrm{Mon}_1(h)$. To avoid pathological cases, we always consider an affine input in x_0 for the CICO-problem such that x_0 is not eliminated after the initial linear operation $\mathcal{M}$. Finally, remember that two monomials may have equal weight, and only get sorted by their lexicographic order.

Lemma 4. *Let $Q_i(x_0, \ldots, x_i)$ be as defined in Figure 4a and ordered according to $\prec_A$. Then the following holds for $\alpha \geq 3$.*

$$\mathrm{LM}(Q_i) = x_0 \cdots x_i, \qquad \textit{and} \qquad \mathrm{wt}(\mathrm{LM}(Q_i)) > \mathrm{wt}(\textit{Mon}(Q_i)) \ .$$

Proof. We proceed by induction. The statements are clearly true for $i = 0$, as Q_0 is an affine polynomial in x_0 by our CICO setting. Now assume it holds for $i-1$. As mentioned above, leading terms cannot be canceled under $\mathcal{M}$, and the leading terms come from the first output of $\mathcal{M}$. Thus, we can restrict ourselves to the two largest monomials in the first output from $\mathcal{M}$, that is $2ax_iQ_{i-1} + ax_i^2 + P_{i-1} - a^{-1}$. From the induction hypothesis we have $\mathrm{LM}(x_iQ_{i-1}) = x_0 \cdots x_i$, and it follows from the definition of $\prec_A$ that this has a strictly higher weight than x_i^2 when $\alpha \geq 3$. □

Lemma 5. *Let $p_i(x_0, \ldots, x_i)$ be as defined in (8) and ordered according to $\prec_A$, and let $\alpha \geq 3$. Then for all $1 \leq i \leq r$ the following holds.*

1. $\mathrm{LM}(p_i) = x_i^\alpha$.
2. $Mon_2(p_i) = (x_0 \cdots x_{i-1})^2$.
3. $\mathrm{wt}(\mathrm{LM}(p_i)) = \mathrm{wt}(Mon_2(p_i)) > \mathrm{wt}(Mon_3(p_i))$.

Proof. We see from the definition of p_i that $\mathrm{LM}(p_i)$ must be either x_i^α or $\mathrm{LM}(Q_i^2)$. From Lemma 4, we have $\mathrm{wt}(\mathrm{LM}(Q_i^2)) = 2\mathrm{wt}(x_0 \cdots x_{i-1})$, so these two monomials have the same weight by definition of $\prec_A$. $\mathrm{LM}(p_i) = x_i^\alpha$ then follows from Definition 2. Finally, $\mathrm{Mon}_3(p_i) = \mathrm{Mon}_2(Q_i^2)$ and thus has a strictly smaller weight than the initial two monomials (Lemma 4). □

Before we can define g^*, we also need a way to predict the powers of x_i we will use in the multiplication of g prior to the reductions by $p_1, \ldots, p_r$. This is handled by the following integer sequences.

Definition 15. *We define two integer sequences $\{u_i\}_{0 \leq i \leq r}$ and $\{k_j\}_{1 \leq j \leq r}$, where $u_r = 1$, and the remaining sequences are recursively defined as follows:*

- *k_i is the unique integer $0 \leq k_i < \alpha$ such that $k_i \equiv -u_i \mod \alpha$;*
- *$u_{i-1} = u_i + 2(u_i + k_i)/\alpha$.*

In the following, we will denote $u = u_0$.

Note that in the above definition, $u_i + k_i$ is always a multiple of α; hence u_{i-1} is indeed an integer.

For a polynomial h and sequence of polynomials H, we write $\mathrm{Red}(h, H)$ to denote the reduction of h by H w.r.t. $\prec_A$. More specifically, $\mathrm{Red}(h, H)$ is the remainder after performing multivariate division of h by H (see [21, Ch. 2, §3]). We are now in a position to define g^*. Let $g'_r = g$, and recursively define

$$g'_{i-1} = \mathrm{Red}(x_i^{k_i} g'_i, \{p_i, p_{i+1}, \ldots, p_r\}), \text{ for } i = r, r-1, \ldots, 1 \ .$$

We set $g^* = g'_0$, and $I_A = \langle P_{\mathcal{G}^*} \rangle$, where $P_{\mathcal{G}^*} = \{p_1, \ldots, p_r, g^*\}$. It is clear from the construction that I_A is a subideal of $\langle P_{\mathcal{G}} \rangle$. The following result guarantees that $P_{\mathcal{G}^*}$ is a FreeLunch system generating this subideal.

Proposition 7. *The polynomial system $P_{\mathcal{G}^*}$ is a FreeLunch system, where $\mathrm{LM}_{\prec_A}(g^*) = x_0^u$. Moreover, the variety of the associated ideal I_A contains all valid solutions of the underlying instance of* `Anemoi`.

Proof. By Lemma 5 we have $\mathrm{LM}(p_i) = x_i^\alpha$, so we need only show that $\mathrm{LM}(g^*)$ is a univariate monomial in x_0 to guarantee that $P_{\mathcal{G}^*}$ is a FreeLunch system. To this end, we will show by induction on descending i that $\mathrm{LM}(g'_i) = (x_0 \cdots x_i)^{u_i}$, and $\mathrm{wt}(\mathrm{LM}(g'_i)) > \mathrm{LM}(\mathrm{Mon}_2(g'_i))$.

For $i = r$, we have $g'_r = g = Q_r$, and the statement holds by Lemma 4. Suppose the hypothesis holds for a given i and consider $i - 1$. If we denote $s_i = (u_i + k_i)/\alpha$, we have $\mathrm{LM}(x_i^{k_i} g'_i) = (x_0 \cdots x_{i-1})^{u_i} x_i^{\alpha s_i}$. We now reduce this

monomial by p_i. Write c for the leading coefficient of $x_i^{k_i} g_i'$. From Lemma 5, we have that the first two monomials in p_i have the same weight, while all other monomials have smaller weights. Moreover, the induction hypothesis ensures that $\mathrm{Mon}_j(x_i^{k_i} g_i')$ will a have smaller weight than $\mathrm{LM}(x_i^{k_i} g_i')$ for $j \geq 2$. Hence,

$$\mathrm{LM}\left(x_i^{k_i} g_i' - c(x_0 \cdots x_{i-1})^{u_i} x_i^{\alpha(s_i-1)} p_i\right) = (x_0 \cdots x_{i-1})^{u_i+2} x_i^{\alpha(s_i-1)},$$

following from the fact that $\mathrm{wt}((x_0 \cdots x_{i-1})^{u_i+2} x_i^{\alpha(s_i-1)}) = \mathrm{wt}(\mathrm{LM}(x_i^{k_i} g_i')) = \mathrm{wt}\left(\mathrm{LM}\left((x_0 \cdots x_{i-1})^{u_i} p_i\right)\right)$, and the weight of all other monomials are guaranteed to be strictly smaller. Repeating this process $s_i - 1$ times, we get $\mathrm{LM}(g_{i-1}') = (x_0 \cdots x_{i-1})^{u_i+2s_i}$, which proves the induction statement. Since this implies that $\mathrm{LM}(g^*) = \mathrm{LM}(g_0') = x_0^u$, it also concludes the proof of the first part of the proposition. The second part holds since I_A is a subideal of $\langle P_{\mathcal{G}} \rangle$, where the variety of the latter ideal will contain all solutions of `Anemoi`. □

Ideal Degree. Based on experiments, the authors of `Anemoi` conjectured a tight upper bound on the ideal degree of one modeling of the CICO-problem to be $(\alpha+2)^r$ [14, Conjecture 2]. As I_A is a FreeLunch system, we have $D_{I_A} = \alpha^r u$, where we recall that u is an integer depending on r and α. As I_A is a subideal of $\langle P_{\mathcal{G}} \rangle$, we generally expect D_{I_A} to be strictly larger than $(\alpha+2)^r$. The following result proved in Appendix E guarantees that D_I can at most differ by a factor close to α, which in practical instances will be a small constant.

Proposition 8. *Let u be as defined in Definition 15 for integers $r, \alpha \geq 1$. Then*

$$\left(\frac{\alpha+2}{\alpha}\right)^r \leq u \leq (\alpha+1)\left(\frac{\alpha+2}{\alpha}\right)^r - \alpha \ .$$

From experiments for $\alpha = 3$ and large r, we find $u \approx 2.1\,(5/3)^r$.

Summary. For `Anemoi`, we get a polynomial system with r equations of respective leading terms $x_1^\alpha, \ldots x_r^\alpha$ and one equation of leading term x_0^u. This gives the following parameters:

$$D_I = \alpha^r u,$$
$$D_H = \alpha^r.$$

Complexity Analysis and Experimental Results. The FreeLunch system $P_{\mathcal{G}^*}$ consists of r polynomials of degree α and one polynomial of degree u. The algorithm of Section 3.2 has a complexity of $\tilde{\mathcal{O}}(\alpha^{r\omega} u)$. We plugged in the numbers for odd `Anemoi` ($\ell = 1$), see Table 6. We also ran experiments for `Anemoi` with $(\ell, \alpha) = (1, 3)$ and different number of rounds to verify the theory presented above. The results are presented in Table 7.

Security claim	$\alpha = 3$	$\alpha = 5$	$\alpha = 7$	$\alpha = 11$
128	118 (21)	156 (21)	174 (20)	198 (19)
256	203 (37)	270 (37)	307 (36)	358 (35)

Table 6: Expected time complexity ($\log_2$) of `polyDet` against different full-round instances of `Anemoi` over $\mathbb{F}_p$, where $\ell = 1$ and $\omega = 2.81$. Number of rounds in parentheses.

Number of rounds	Complexity of `polyDet`	Time (s) `sysGen`	Time (s) `matGen`	Time (s) `polyDet`	Memory (MB)
3	20	< 0.01	< 0.01	0.02	< 400
4	26	< 0.01	0.34	0.24	< 400
5	32	0.07	23.3	7.6	< 400
6	37	2.52	2,127	292	2,863
7	43	128	156,348	10,725	42,337

Table 7: Experimental results on `Anemoi` with $(\ell, \alpha) = (1, 3)$. `sysGen` is performed with MAGMA and refers to the generation of the polynomial system $P_{\mathcal{G}^*}$ from scratch, including the computation of $P_{\mathcal{G}}$.

6 Conclusions

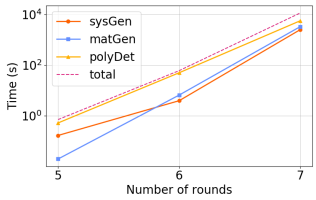
We have presented the FreeLunch approach, an algebraic attack particularly efficient against arithmetization-oriented permutations. We conclude this paper with some comments regarding our experiments as well the consequences of our results, in particular regarding the areas we believe are worth investigating further.

6.1 Discussion on Experimental Results

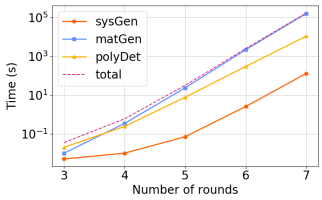
Figure 5 depicts the runtimes of each step of our attack that we obtained experimentally when targeting `Griffin` and `Anemoi`. A first observation is that the running time of a full FreeLunch-based attack is hard to predict: there are three steps (`sysGen`, `matGen`, and `polyDet`), and we experimentally found situations where each of them was the slowest. The case of `sysGen` is a bit peculiar: using `SageMath`, MAGMA or Flint/NTL yields very different results and a deeper understanding seems out of our grasp. We nevertheless would argue (see Appendix A) that, should their implementations use similar tools, `sysGen` will always be of lower complexity than that of the rest of the attack.

Assuming that the dominating step is either `matGen` or `polyDet`, it then seems easy to extrapolate: as we can see in Figure 5, their logarithm increases linearly with the number of rounds. Even better: for `Griffin`, Equation (7) predicts that adding a round multiplies the complexity of `polyDet` by $\alpha^\omega(2\alpha + 1) \approx 109.5$, which closely matches our observations as $5727/50.79 \approx 112.7$. For `matGen`, we see that adding a round multiplies the time complexity by about 500. Extrapolating from this, an attack against full-round `Griffin` should take

about $4.2 \cdot 10^{11}$s on a single CPU (around 13,000 years), or around 2^{70} clock cycles at 2.3 GHz. Similarly, for `Anemoi` with $\mathbb{F}_p$ and $(l, \alpha) = (1, 3)$, adding a round multiplies the time complexity of `matGen` by about 75. Extrapolating gives respectively 2^{104} seconds (or 2^{135} clock cycles) and 2^{204} seconds (2^{235} clock cycles) for full-round `Anemoi` with 128 and 256 bits of security.



(a) `Griffin` complexity (from table 3).



(b) `Anemoi` complexity (from table 7).

Fig. 5: Experimental time complexity of our attacks on `Griffin` and `Anemoi`.

6.2 Preventing the FreeLunch Attack

Our attack breaks full-round instances of symmetric primitives built using state-of-the-art security arguments, which consequently must be revisited: one must learn how to prevent the relevant applicability of the FreeLunch approach.

At the Primitive Level. An obvious but perhaps costly countermeasure consists of simply adding more rounds. This is particularly tempting as we are able to tightly estimate the complexity of `polyDet`, a step which we have found to often be the most expensive in practice. Choosing a number of rounds high enough to prevent it would be a simple yet convincing argument. Primitive designers must also be mindful of "classical" tricks, i.e., symmetric cryptanalysis techniques (a priori) unrelated to root finding that can be used to enhance its efficiency. In the case of 12-branch `Griffin`, the fact that we can bypass 3 out of 10 rounds using some kind of subspace trail is a problem we deem worth studying.

At the Mode of Operation Level. The FreeLunch systems are multivariate, but a single variable (x_0) plays an inherently different role. This makes them particularly well suited to CICO instances whereby a single output word has to be set to 0, but they will not work if more 0's are needed in the output. Thus, a simple countermeasure against the FreeLunch approach (and univariate ones) consists of forcing the capacity of the sponge to have at least two words set to 0, even if one word would a priori be enough. Still, while easy to implement, the efficiency of this countermeasure in the long term is uncertain. Indeed, as argued below, inventing a variant of the FreeLunch that can handle multiple words is an interesting open problem.

6.3 Open Problems for Future Work

Time Taken by Polynomial Reductions. A roadblock in our complexity estimates is the number of operations needed to perform certain reductions of a polynomial modulo an ideal. This is crucial for understanding the complexity of the `matGen` step and, to a lesser degree, `sysGen` (see Appendix A). A tighter estimate for these computations would greatly benefit our analysis: we would be able to figure out which step of our attack is the actual bottleneck without the need for experiments or assumptions, and designers could then be able to use fewer rounds to achieve a given security level against FreeLunch-based attacks. For instance, estimating the complexity of a reduction by a FreeLunch triangular system (Definition 9) would be a big step forward.

Other Custom Approaches. The FreeLunch approach is, to the best of our knowledge, the first "custom" root finding method designed specifically for use in symmetric cryptanalysis. There is, of course, no reason to believe that it is the only one possible, and we consider it a direction worth pursuing. As a first step, a multivariate variant of FreeLunch where several variables play the role of x_0, and where several words need to be set to 0, would be an interesting target.

Acknowledgements

This work has been facilitated through the COSINUS associate team between Inria and Simula. The authors would like to thank Gaëtan Leurent for the helpful insights on the new dedicated Gröbner basis solving algorithm, and Pierre Briaud and Carlos Cid for the good discussions in the early stages of this work. The authors would also like to thank Vincent Neiger for the proof-reading and the discussions on the algorithmic aspects of the Gröbner basis theory. The research in this paper was supported in part by the French DGA, and by the French grant 22-PECY-0010 (project CRYPTANALYSE). The work of Aurélien Bœuf, Axel Lemoine, and Léo Perrin was supported by the European Research Council (ERC, grant agreement no. 101041545 "ReSCALE"). Morten Øygarden has been supported by the Norwegian Research Council through the project qsIo2.

References

1. Advanced Encryption Standard (AES). National Institute of Standards and Technology, NIST FIPS PUB 197, U.S. Department of Commerce (Nov 2001)
2. Albrecht, M.R., Cid, C., Grassi, L., Khovratovich, D., Lüftenegger, R., Rechberger, C., Schofnegger, M.: Algebraic Cryptanalysis of STARK-Friendly Designs: Application to MARVELlous and MiMC. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III. Lecture Notes in Computer Science, vol. 11923, pp. 371–397. Springer (2019). https://doi.org/10.1007/978-3-030-34618-8_13

3. Albrecht, M.R., Grassi, L., Rechberger, C., Roy, A., Tiessen, T.: MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity. In: Cheon, J.H., Takagi, T. (eds.) Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10031, pp. 191–219 (2016). https://doi.org/10.1007/978-3-662-53887-6_7
4. Albrecht, M.R., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for MPC and FHE. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9056, pp. 430–454. Springer (2015). https://doi.org/10.1007/978-3-662-46800-5_17
5. Aly, A., Ashur, T., Ben-Sasson, E., Dhooghe, S., Szepieniec, A.: Design of Symmetric-Key Primitives for Advanced Cryptographic Protocols. IACR Transactions on Symmetric Cryptology **2020**(3), 1–45 (Sep 2020), https://tosc.iacr.org/index.php/ToSC/article/view/8695
6. Ashur, T., Kindi, A., Mahzoun, M.: XHash8 and XHash12: Efficient STARK-friendly Hash Functions. Cryptology ePrint Archive, Paper 2023/1045 (2023), https://eprint.iacr.org/2023/1045
7. Ashur, T., Kindi, A., Meier, W., Szepieniec, A., Threadbare, B.: Rescue-Prime Optimized. Cryptology ePrint Archive, Paper 2022/1577 (2022), https://eprint.iacr.org/2022/1577
8. Ashur, T., Mahzoun, M., Toprakhisar, D.: Chaghri - An FHE-friendly Block Cipher. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022. pp. 139–150. ACM (2022). https://doi.org/10.1145/3548606.3559364
9. Bariant, A.: Algebraic Cryptanalysis of Full Ciminion. Cryptology ePrint Archive, Paper 2023/1283 (2023), https://eprint.iacr.org/2023/1283
10. Bariant, A., Bouvier, C., Leurent, G., Perrin, L.: Algebraic Attacks against Some Arithmetization-Oriented Primitives. IACR Transactions on Symmetric Cryptology **2022**(3), 73–101 (Sep 2022), https://tosc.iacr.org/index.php/ToSC/article/view/9850
11. Berthomieu, J., Neiger, V., Safey El Din, M.: Faster change of order algorithm for Gröbner bases under shape and stability assumptions. In: Proceedings of the 2022 International Symposium on Symbolic and Algebraic Computation. pp. 409–418 (2022)
12. Bertoni, G., Daemen, J., Peters, M., Assche, G.V.: Cryptographic Sponge Functions (2011), https://keccak.team/files/CSF-0.1.pdf (accessed 23.05.2024)
13. Bosma, W., Cannon, J., Playoust, C.: The Magma algebra system, I. The User Language. J. Symbolic Comput. **24**(3-4), 235–265 (1997), http://dx.doi.org/10.1006/jsco.1996.0125, computational algebra and number theory (London, 1993)
14. Bouvier, C., Briaud, P., Chaidos, P., Perrin, L., Salen, R., Velichkov, V., Willems, D.: New Design Techniques for Efficient Arithmetization-Oriented Hash Functions: Anemoi Permutations and Jive Compression Mode. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology – CRYPTO 2023. Lecture Notes in Computer Science, vol. 14083, pp. 507–539. Springer Nature Switzerland, Cham (2023)
15. Buchberger, B.: A theoretical basis for the reduction of polynomials to canonical forms. ACM SIGSAM Bulletin **10**(3), 19–29 (1976)

16. Buchmann, J., Pyshkin, A., Weinmann, R.P.: A Zero-Dimensional Gröbner Basis for AES-128. In: Robshaw, M. (ed.) Fast Software Encryption. Lecture Notes in Computer Science, vol. 4047, pp. 78–88. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
17. Canteaut, A., Carpov, S., Fontaine, C., Lepoint, T., Naya-Plasencia, M., Paillier, P., Sirdey, R.: Stream Ciphers: A Practical Solution for Efficient Homomorphic-Ciphertext Compression. In: Peyrin, T. (ed.) Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9783, pp. 313–333. Springer (2016). https://doi.org/10.1007/978-3-662-52993-5_16
18. Canteaut, A., Carpov, S., Fontaine, C., Lepoint, T., Naya-Plasencia, M., Paillier, P., Sirdey, R.: Stream Ciphers: A Practical Solution for Efficient Homomorphic-Ciphertext Compression. J. Cryptol. **31**(3), 885–916 (2018). https://doi.org/10.1007/S00145-017-9273-9
19. Cantor, D.G., Kaltofen, E.: On Fast Multiplication of Polynomials over Arbitrary Algebras. Acta Informatica **28**(7), 693–701 (1991)
20. Cosseron, O., Hoffmann, C., Méaux, P., Standaert, F.: Towards Case-Optimized Hybrid Homomorphic Encryption - Featuring the Elisabeth Stream Cipher. In: Agrawal, S., Lin, D. (eds.) Advances in Cryptology - ASIACRYPT 2022 - 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5-9, 2022, Proceedings, Part III. Lecture Notes in Computer Science, vol. 13793, pp. 32–67. Springer (2022). https://doi.org/10.1007/978-3-031-22969-5_2
21. Cox, D., Little, J., OShea, D.: Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra. Springer Science & Business Media (2013)
22. Cox, D.A., Little, J.B., O'Shea, D.: Using Algebraic Geometry, Graduate Texts in Mathematics, vol. 185. Springer (1998). https://doi.org/10.1007/978-1-4757-6911-1
23. Dobraunig, C., Eichlseder, M., Grassi, L., Lallemand, V., Leander, G., List, E., Mendel, F., Rechberger, C.: Rasta: A Cipher with Low ANDdepth and Few ANDs per Bit. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10991, pp. 662–692. Springer (2018). https://doi.org/10.1007/978-3-319-96884-1_22
24. Duval, S., Lallemand, V., Rotella, Y.: Cryptanalysis of the FLIP Family of Stream Ciphers. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9814, pp. 457–475. Springer (2016). https://doi.org/10.1007/978-3-662-53018-4_17
25. Eisenbud, D.: Commutative Algebra: With a View Toward Algebraic Geometry, vol. 150. Springer Science & Business Media (2013)
26. Faugère, J.C., Mou, C.: Sparse FGLM algorithms. Journal of Symbolic Computation **80**, 538–569 (2017)
27. Faugère, Jean-Charles and Gaudry, Pierrick and Huot, Louise and Renault, Guénaël: Sub-cubic change of ordering for Gröbner basis: a probabilistic approach. In: Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation. pp. 170–177 (2014)

28. Faugère, Jean-Charles and Gianni, Patrizia and Lazard, Daniel and Mora, Teo: Efficient computation of zero-dimensional Gröbner bases by change of ordering. Journal of Symbolic Computation **16**(4), 329–344 (1993)
29. Faugère, J.C.: A new efficient algorithm for computing Gröbner bases (F_4). Journal of pure and applied algebra **139**(1-3), 61–88 (1999)
30. Faugère, J.C.: A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5). In: Proceedings of the 2002 international symposium on Symbolic and algebraic computation. pp. 75–83 (2002)
31. Gilbert, H., Boissier, R.H., Jean, J., Reinhard, J.: Cryptanalysis of Elisabeth-4. In: Guo, J., Steinfeld, R. (eds.) Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part III. Lecture Notes in Computer Science, vol. 14440, pp. 256–284. Springer (2023). https://doi.org/10.1007/978-981-99-8727-6_9
32. Gilbert, H., Peyrin, T.: Super-Sbox Cryptanalysis: Improved Attacks for AES-Like Permutations. In: Hong, S., Iwata, T. (eds.) Fast Software Encryption, 17th International Workshop, FSE 2010, Seoul, Korea, February 7-10, 2010, Revised Selected Papers. Lecture Notes in Computer Science, vol. 6147, pp. 365–383. Springer (2010). https://doi.org/10.1007/978-3-642-13858-4_21
33. Giorgi, P., Jeannerod, C.P., Villard, G.: On the Complexity of Polynomial Matrix Computations. In: Proceedings of the 2003 international symposium on Symbolic and algebraic computation. pp. 135–142 (2003)
34. Grassi, L., Hao, Y., Rechberger, C., Schofnegger, M., Walch, R., Wang, Q.: Horst Meets Fluid-SPN: Griffin for Zero-Knowledge Applications. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part III. Lecture Notes in Computer Science, vol. 14083, pp. 573–606. Springer (2023). https://doi.org/10.1007/978-3-031-38548-3_19
35. Guido, B., Joan, D., Michaël, P., Gilles, V.: Cryptographic sponge functions (2011), https://keccak.team/files/CSF-0.1.pdf
36. Ha, J., Kim, S., Lee, B., Lee, J., Son, M.: Rubato: Noisy Ciphers for Approximate Homomorphic Encryption. In: Dunkelman, O., Dziembowski, S. (eds.) Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part I. Lecture Notes in Computer Science, vol. 13275, pp. 581–610. Springer (2022). https://doi.org/10.1007/978-3-031-06944-4_20
37. Hart, W.B.: Flint: Fast Library for Number Theory. Computeralgebra-Rundbrief: Vol. 49 (2011)
38. Hyun, S.G., Neiger, V., Schost, É.: Implementations of Efficient Univariate Polynomial Matrix Algorithms and Application to Bivariate Resultants. In: Proceedings ISSAC 2019. pp. 235–242. ACM (2019). https://doi.org/10.1145/3326229.3326272, https://github.com/vneiger/pml
39. Labahn, G., Neiger, V., Zhou, W.: Fast, deterministic computation of the Hermite normal form and determinant of a polynomial matrix. Journal of Complexity **42**, 44–71 (2017)
40. Masure, L., Méaux, P., Moos, T., Standaert, F.: Effective and Efficient Masking with Low Noise Using Small-Mersenne-Prime Ciphers. In: Hazay, C., Stam, M. (eds.) 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. Lecture Notes in Computer Science, vol. 14007, pp. 596–627. Springer (2023). https://doi.org/10.1007/978-3-031-30634-1_20
41. Méaux, P., Journault, A., Standaert, F., Carlet, C.: Towards Stream Ciphers for Efficient FHE with Low-Noise Ciphertexts. In: Fischlin, M., Coron, J. (eds.) Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9665, pp. 311–343. Springer (2016). https://doi.org/10.1007/978-3-662-49890-3_13
42. Moenck, R.T.: Practical fast polynomial multiplication. In: Proceedings of the third ACM symposium on Symbolic and algebraic computation. pp. 136–148 (1976)
43. Neiger, V., Schost, É.: Computing syzygies in finite dimension using fast linear algebra. J. Complex. **60**, 101502 (2020). https://doi.org/10.1016/J.JCO.2020.101502, https://doi.org/10.1016/j.jco.2020.101502
44. Roy, A., Steiner, M.J., Trevisani, S.: Arion: Arithmetization-Oriented Permutation and Hashing from Generalized Triangular Dynamical Systems (2023), https://arxiv.org/abs/2303.04639
45. Szepieniec, A., Ashur, T., Dhooghe, S.: Rescue-Prime: a Standard Specification (SoK). Cryptology ePrint Archive, Paper 2020/1143 (2020), https://eprint.iacr.org/2020/1143
46. The PML team: PML: Polynomial Matrix Library (2023), version 0.3, https://github.com/vneiger/pml
47. The Sage Developers: SageMath, the Sage Mathematics Software System (2022), https://www.sagemath.org
48. V. Shoup, et al.: NTL: A Library for Doing Number Theory. https://libntl.org/
49. Von Zur Gathen, J., Gerhard, J.: Modern Computer Algebra. Cambridge university press (2013)

A Computing a Reduced Gröbner Basis for $\langle P_{\mathcal{G}} \rangle$

As noted at the end of Section 3.4, we do not generate the polynomial system $P_{\mathcal{G}}$ directly in practice. Rather, we construct a related polynomial system iteratively while reducing as many monomials as possible along the way. More formally, for a polynomial h and an ordered sequence of polynomials H, we let $\mathrm{Red}(h, H)$ denote the operation of reducing h by H (according to a specified monomial order). That is, $\mathrm{Red}(h, H)$ is the remainder after performing multivariate division of h by H (see [21, Ch. 2, §3]). For a tuple of polynomials $\boldsymbol{h} = (h_1, \ldots, h_t)$, we write $\mathrm{Red}(\boldsymbol{h}, H) = (\mathrm{Red}(h_1, H), \ldots, \mathrm{Red}(h_t, H))$. Now fix a monomial order, and define $\boldsymbol{z}'_0 = \boldsymbol{z}_0$. We generate $\boldsymbol{p}'_i = (p'_{i,1}, \ldots, p'_{i,l_i})$ and the reduced states $\boldsymbol{z}'_i$ recursively as follows for $1 \le i \le r$ and $1 \le j \le l_i$.

$$p'_{i,j} = \mathrm{Red}\left(x_{i,j}^{\alpha_{i,j}} - \mathcal{L}_{i,j}(\boldsymbol{z}'_{i-1}), \{\boldsymbol{p}'_1 \ldots, \boldsymbol{p}'_{i-1}\}\right),$$
$$\boldsymbol{z}'_i = \mathrm{Red}\left(G_i(\boldsymbol{z}'_{i-1}, \boldsymbol{x}_i), \{\boldsymbol{p}'_1, \ldots, \boldsymbol{p}'_i\}\right),$$

where $\mathcal{L}_{i,j}$ is the polynomial from (3). Finally, we define

$$g' = \mathrm{Red}\left([G_r(\boldsymbol{z}'_{r-1}, \boldsymbol{x}_r)]_1, \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_r\}\right),$$

and write $P'_{\mathcal{G}} = \{\boldsymbol{p}'_1, \ldots, \boldsymbol{p}'_r, g'\}$. Since the construction of $P'_{\mathcal{G}}$ only differs from that of $P_{\mathcal{G}}$ by reductions with generators in the ideal $I_{\mathcal{G}} = \langle P_{\mathcal{G}} \rangle$ their ideals should, intuitively speaking, be identical. This intuition is confirmed by the following lemma.

Lemma 6. *For any fixed monomial order we have*

$$I_{\mathcal{G}} = \langle P_{\mathcal{G}} \rangle = \langle P'_{\mathcal{G}} \rangle \ .$$

Proof. For any polynomial h and polynomial sequence H, we can write the reduction operation as $\mathrm{Red}(h, H) = h + W$, for some polynomial $W \in \langle H \rangle$. Since the G_i's used in the construction of $\boldsymbol{p}'_i$ and $\boldsymbol{z}'_i$ are polynomial functions, one can show by induction that

$$p'_{i,j} \in p_{i,j} + \langle \{\boldsymbol{p}_1 \ldots, \boldsymbol{p}_{i-1}\} \rangle, \qquad z'_{i,j} \in z_{i,j} + \langle \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_i\} \rangle \tag{10}$$

holds for all $1 \leq i \leq r$ and $1 \leq j \leq l_i$. In particular, we have $g' \in g + \langle \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_r\} \rangle$. Thus it is clear that $P_{\mathcal{G}}$ and $P'_{\mathcal{G}}$ generate the same polynomial ideal. □

The following result relates $P'_{\mathcal{G}}$ and $P_{\mathcal{G}}$ when Proposition 6 holds. Recall that we write $d_{\leq i} = d_1 \cdots d_i$, where $d_i = \deg(G_i)$.

Proposition 9. *Let $P_{\mathcal{G}}$ satisfy the condition of Proposition 6. Then constructing $P'_{\mathcal{G}}$ w.r.t. $\prec_{\mathcal{G}}$ is also a FreeLunch system. Moreover, replacing g' in $P'_{\mathcal{G}}$ with $g'/LC(g')$ yields the unique reduced Gröbner basis for $I_{\mathcal{G}}$ w.r.t. $\prec_{\mathcal{G}}$.*

Proof. By definition of polynomial division, we have $\mathrm{wt}(\mathrm{LM}(\mathrm{Red}(h, H))) \leq \mathrm{wt}(\mathrm{LM}(h))$. Since $\mathrm{LM}(p_{i,j})'$ cannot be reduced by $\{\boldsymbol{p}'_1 \ldots, \boldsymbol{p}'_{i-1}\}$ under $\prec_{\mathcal{G}}$, it follows from (10) and the prior discussion that $\mathrm{LM}(p'_{i,j}) = \mathrm{LM}(p_{i,j})$. For the similar statement on $\mathrm{LM}(g')$, we note that the condition $\mathrm{LM}(g) = x_0^{d_{\leq r}}$ can only hold if for every i there exist a j_i such that $\mathrm{LM}(z_{i,j_i}) = x_0^{d_{\leq i}}$. By construction of $\prec_{\mathcal{G}}$, this monomial will not be reduced by $\{\boldsymbol{p}'_1 \ldots, \boldsymbol{p}'_{i-1}\}$. Again, it follows from (10) that $\mathrm{LM}(z'_{i,j_i}) = x_0^{d_{\leq i}}$. In particular, $\mathrm{LM}(g') = \mathrm{LM}(g)$, hence $P'_{\mathcal{G}}$ is also a FreeLunch system.

For the last assertion, one observes from the way $p_{i,j}$ only depends on the variables $x_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{i-1}, x_{i,j}$ that

$$\mathrm{Red}(p'_{i,j}, P'_{\mathcal{G}} \setminus \{p'_{i,j}\}) = \mathrm{Red}(p'_{i,j}, \{\boldsymbol{p}_1 \ldots, \boldsymbol{p}_{i-1}\}) \ ,$$

holds for $\prec_{\mathcal{G}}$. Hence $P'_{\mathcal{G}}$ is already fully reduced, and replacing g' with $g'/\mathrm{LC}(g')$ makes all polynomials monic. □

Remark 2. Recall from Proposition 2 that if H is a Gröbner basis for $\langle H \rangle$, then $\mathrm{Red}(h, H)$ does not depend on the order of the sequence H. It follows from Proposition 9 that if $P_{\mathcal{G}}$ satisfies the condition of Proposition 6, then the reductions in the construction of $p'_{i,j}$, $\boldsymbol{z}'_i$ and g' are independent of the order of the sequence $\{\boldsymbol{p}_1, \ldots \boldsymbol{p}_i\}$, w.r.t. $\prec_{\mathcal{G}}$.

Complexity of computing $P'_{\mathcal{G}}$. We are left with bounding the complexity of computing $P'_{\mathcal{G}}$, which will yield our estimate for the `sysGen` step. In the setting we will be interested in, this is expected to be dominated by the cost of applying the last round function G_r to compute g', and its reduction by $\{\boldsymbol{p}'_1 \dots, \boldsymbol{p}'_r\}$. Our insight is that the reductions involved in the `sysGen` process are cheaper than the reductions required in `matGen`, since the reductions are performed on a smaller Gröbner basis; but we do not have a proof for such a statement. However, it is possible to bound the cost of the multiplications performed on the state $\boldsymbol{z}'_{r-1}$ when applying G_r. Let m denote the number of these multiplication, where we recall that m is typically small by design. We reduce by $\{\boldsymbol{p}'_1 \dots, \boldsymbol{p}'_r\}$ after each multiplication, and will assume that this reduction is negligible compared to the cost of the multiplications themselves. Thus we have m multiplications of multivariate polynomials of maximal degree $d_{\leq r}$ in x_0 and $\alpha_{i,j} - 1$ in $x_{i,j}$, for $1 \leq i \leq r$, $1 \leq j \leq l_i$. We can then use the Kronecker trick presented by Moenck [42, Section 3.4] to perform these multiplications in an efficient manner. In short, the Kronecker trick starts by transforming the multivariate polynomials to univariate polynomials. This allows us to perform the multiplication using an efficient univariate multiplication algorithm, before converting the result back to a multivariate polynomial. Moenck describes the algorithm and proves its correctness for any bound on the degree of each variable in both polynomials in the input of multiplication, but only gives a complexity estimate when all bounds are equal. It is, however, easy to verify that the complexity formula for the multivariate multiplication algorithm in our setting will be:

$$\tilde{\mathcal{O}}(d_{\leq r} \prod_{\substack{1 \leq i \leq r \\ 1 \leq j \leq l_i}} 2\alpha_{i,j}) \ ,$$

when applying either the Fast Fourier Transform, or Schönhage & Strassen's algorithm to perform the univariate multiplication [49, Chapter 8]. Repeating this m times yields our estimate for cost of multiplications in the `sysGen` step:

$$\tilde{\mathcal{O}}(m d_{\leq r} \prod_{\substack{1 \leq i \leq r \\ 1 \leq j \leq l_i}} 2\alpha_{i,j}) \ .$$

In comparison, recall that the `polyDet` step of our analysis is expected by Theorem 1 to require

$$\tilde{O}(d_{\leq r}(\prod_{\substack{1 \leq i \leq r \\ 1 \leq j \leq l_i}} \alpha_{i,j})^{\omega})$$

operations in $\mathbb{F}$. Thus, when m remains small, we do not expect the multiplications in `sysGen` to be the bottleneck of the overall attack.

Use in Experiments. We implemented the Kronecker trick for the experiments we ran with the Flint library [37], using the NTL library [48] for the univariate multiplication; the mapping between flint and NTL polynomial representations was performed by hand. The multivariate multiplications performed for experiments with Magma and `SageMath` used their own built-in functionalities.

B Bypassing the first rounds of `Griffin`

The number of rounds that can be bypassed before we need to introduce x_1 depends on the number of branches. For $t \geq 12$ branches we can find an easily computable set of input states that allows to bypass the first three rounds of `Griffin`, so x_1 only appears in the fourth round. We explain in detail how this can be done for $t = 12$. After that it will become clear that three rounds can also be bypassed for $t \in \{16, 20, 24\}$, and how to determine how many rounds can be bypassed for $t < 12$.

Denote the input state to `Griffin` as

$$(a_0x_0 + b_0, a_1x_0 + b_1, a_2x_0 + b_2, \ldots, a_{10}x_0 + b_{10}, 0).$$

The a_i and b_j are constants in $\mathbb{F}$ that we now proceed to determine. Once the a_i and b_j are fixed the variable x_0 can be varied freely over $\mathbb{F}$, generating a set of input states for the CICO problem that all have constant input to the $x^{1/\alpha}$ function in the three first rounds. Figure 6 illustrates the evolution of one of the chosen input states up to the start of round 3.

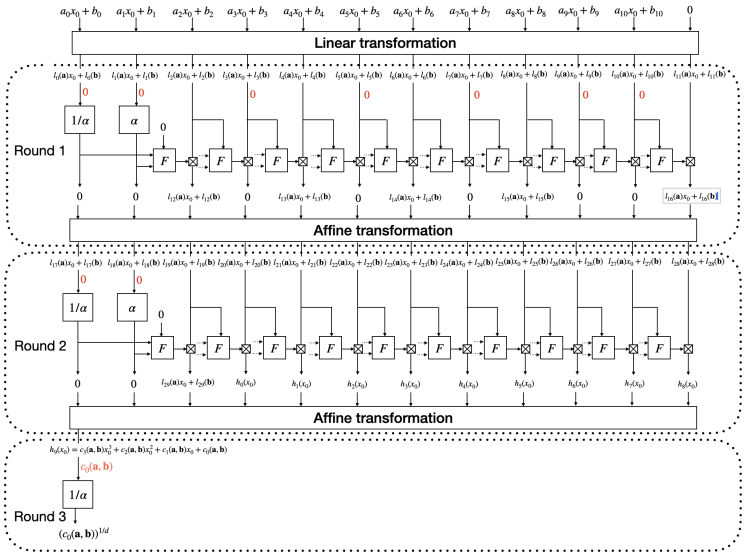


Fig. 6: Evolution of chosen set of input states to `Griffin` with 12 branches. Red values give conditions on the a_i and b_j such that the input of $x^{1/\alpha}$ in the third round becomes a known constant independent of x_0.

The values of a_i and b_j in Figure 6 can be determined as follows. After the initial linear transformation before the first round, all branches can be expressed as $l_i(\boldsymbol{a})x_0 + l_i(\boldsymbol{b})$ for $0 \leq i \leq 11$, where $l_i(\cdot)$ is a known linear combination. To get 0 on the branches indicated in Figure 6, the a_i's and b_j's need to satisfy the following linear equations

$$\begin{aligned} l_0(\boldsymbol{a}) &= 0 & l_0(\boldsymbol{b}) &= 0 \\ l_1(\boldsymbol{a}) &= 0 & l_1(\boldsymbol{b}) &= 0 \\ l_3(\boldsymbol{a}) &= 0 & l_3(\boldsymbol{b}) &= 0 \\ l_5(\boldsymbol{a}) &= 0 & l_5(\boldsymbol{b}) &= 0 \\ l_7(\boldsymbol{a}) &= 0 & l_7(\boldsymbol{b}) &= 0 \\ l_9(\boldsymbol{a}) &= 0 & l_9(\boldsymbol{b}) &= 0 \\ l_{10}(\boldsymbol{a}) &= 0 & l_{10}(\boldsymbol{b}) &= 0. \end{aligned}$$

With 0 on any two adjacent branches, the input to F will either be all 0, with $F(0,0,0)$ being equal to a constant, or the output of F will be multiplied with 0, making sure the value on the branch remains 0. This ensures that the algebraic expressions on the branches stay linear in $x_0, \boldsymbol{a}$ and $\boldsymbol{b}$ after the affine transformation at the start of the second round. The need to have input 0 to $x^{1/\alpha}$ and x^{α} in the second round gives four more linear constraints

$$\begin{aligned} l_{17}(\boldsymbol{a}) &= 0 & l_{17}(\boldsymbol{b}) &= 0 \\ l_{18}(\boldsymbol{a}) &= 0 & l_{18}(\boldsymbol{b}) &= 0, \end{aligned}$$

where the γ_i are known constants.

Before the affine transformation in the second round, most branches will have cubic polynomials in x_0 as their values (the $h_i(x_0)$ in Figure 6). These are again linearly mixed in the affine transformation at the end of round two, producing the cubic polynomial

$$h_9(x_0) = c_3(\boldsymbol{a}, \boldsymbol{b})x_0^3 + c_2(\boldsymbol{a}, \boldsymbol{b})x_0^2 + c_1(\boldsymbol{a}, \boldsymbol{b})x_0 + c_0(\boldsymbol{a}, \boldsymbol{b})$$

on the first branch. We want to enforce that $c_3(\boldsymbol{a}, \boldsymbol{b}) = c_2(\boldsymbol{a}, \boldsymbol{b}) = c_1(\boldsymbol{a}, \boldsymbol{b}) = 0$ such that the input to the $x^{1/\alpha}$ function in round three becomes a known constant independent from x_0. The expressions for the coefficients are cubic in the a_i and b_j, but note that all the polynomials $h_i(x_0)$ for $0 \leq i \leq 8$ are made as products of linear factors as

$$(l_i(\boldsymbol{a})x_0 + l_i(\boldsymbol{b}))(l_j(\boldsymbol{a})x_0 + l_j(\boldsymbol{b}))(l_k(\boldsymbol{a})x_0 + l_k(\boldsymbol{b})),$$

and that $h_9(x_0)$ is a sum of these. By calculating the coefficients for the x_0^3, x_0^2, and x_0 terms, we see that $c_3(\boldsymbol{a}, \boldsymbol{b})$ is cubic in $\boldsymbol{a}$, but does not contain $\boldsymbol{b}$ at all. Similarly, $c_2(\boldsymbol{a}, \boldsymbol{b})$ is quadratic in $\boldsymbol{a}$ and linear in $\boldsymbol{b}$ and $c_1(\boldsymbol{a}, \boldsymbol{b})$ is linear in $\boldsymbol{a}$ and quadratic in $\boldsymbol{b}$.

We can now use the 9 linear equations in $\boldsymbol{a}$ introduced above to eliminate $a_2, \ldots, a_{10}$ from $c_3(\boldsymbol{a})$. This leaves c_3 as $c_3(a_0, a_1)$, a cubic expression in a_0 and a_1. Next we fix a_1 to an arbitrary non-zero value (to avoid the trivial solution $a_0 = \ldots = a_{10} = 0$) and solve for $c_3(a_0) = 0$ using a root-finding algorithm for univariate polynomials. With a_0 and a_1 fixed, all the other a_i gets fixed as well from the linear constraints from rounds 1 and 2.

Once all a_i have been found, $c_2(\boldsymbol{a}, \boldsymbol{b}) = 0$ just becomes a linear equation in $\boldsymbol{b}$. Using this linear equation together with the 9 from above, we can eliminate $b_1, \ldots, b_{10}$ from the last coefficient $c_1(\boldsymbol{a}, \boldsymbol{b})$. With all the a_i fixed, c_1 then just becomes $c_1(b_0)$, a quadratic expression in b_0 and we easily solve $c_1(b_0) = 0$. This determines all the values for the b_i.

With the a_i and b_j now fixed, we know that the input state from our chosen set will generate polynomials in x_0 of degree $6\alpha + 3$ on the branches at the start of round 4. We can then start the basic attack from there, adapting the weighted order of the variables accordingly. When the number of x_i-variables is reduced by 3 and with the degree of x_0 bounded to $6\alpha + 3$ until the fourth round, the dimension of the Gröbner basis ideal becomes much smaller, which again reduces the overall attack complexity significantly.

When there are more than 12 branches we can do the exact same trick as explained above. The only difference is that there will be more values of a_i and b_j that can be chosen arbitrarily when solving for $c_3(\boldsymbol{a}, \boldsymbol{b}) = c_2(\boldsymbol{a}, \boldsymbol{b}) = c_1(\boldsymbol{a}, \boldsymbol{b}) = 0$. When there are less than 12 branches, there is not enough degrees of freedom to make it through the third round. For $t = 8$ we can bypass the two first rounds, so x_1 only needs to be introduced in round 3, and for $t = 3, 4$ it is possible to bypass the first round and introduce x_1 in round 2.

C Solutions to the CICO problem with respect to `Griffin`

In this section we give explicit solutions to the CICO problem related to a `Griffin` permutation whose characteristics are specified below. Everything discussed here can be checked by the reviewers using the supplementary material provided in verify.zip.

Parameters of `Griffin` instances

- Prime number: $p = 28407454060060787$ (55 bits);
- Exponent: $\alpha = 3$;
- Number of rounds: $r \in \{5, 6, 7\}$;
- Number of branches: $t = 12$ (corresponds to 10 rounds in the real version);
- Parameters of the quadratic functions: $\delta_i = 4(i + 1),\ \mu_i = 7(i + 1)^2,\ i = 0, \ldots, r - 1$;
- Round constants[13] (up to 7 rounds):

[13] Our `Griffin` with $r \leq 7$ rounds will use constants $(\boldsymbol{c}_0, \ldots, \boldsymbol{c}_{r-2}, \boldsymbol{c}_6)$.

$$
\begin{aligned}
c_0 = (&24948861045225956, 21203603017242449, 5137804740880040,\\
&17203989140901077, 15884693750499599, 202426034695061,\\
&21925627314327927, 14915791625715646, 2270637844838412,\\
&19287066862534400, 23053628619630528, 20205234482325465)\\
c_1 = (&1953994697959081, 13694436956212586, 2244645965787647,\\
&20803493439220167, 13296675195272853, 18296898451764242,\\
&20376008308269607, 10239947264048958, 1116873941458788,\\
&19425600591729552, 20854422412323996, 10085561279368253)\\
c_2 = (&16905640598099854, 25230133406843513, 8957962046730991,\\
&14294289436907403, 10949906559535418, 28179662462119909,\\
&20848690834284278, 5962920227944130, 15129107418293752,\\
&6002695762195648, 6114627516150292, 22521669951514122)\\
c_3 = (&13008050022403386, 2809135024568407 9, 23189230572909585,\\
&8101795236077784, 3593606052472638, 11330866710107896,\\
&9840541134611106, 13915746912957553, 19822110644988410,\\
&24750875289653592, 25496607366081073, 2269647499797729)\\
c_4 = (&15770582149454036, 4472996328290429, 8197094411507273,\\
&14151116175893923, 19977244056516294, 22071066831282832,\\
&10912395968228633, 28293903648852908, 15600461636809584,\\
&16248565278833955, 23850742575902912, 12384888390231181)\\
c_5 = (&24731271392756981, 4234794164011219, 5709721189329773,\\
&23115678305163655, 11185048660199721, 21406367947811616,\\
&14929808901464726, 14209993563715217, 19373914823111461,\\
&12307896526346864, 16319890415782340, 19440350754040851)\\
c_6 = (&0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)
\end{aligned}
$$

The round constants listed above were generated randomly using the following code in `SageMath`:

```
p = 28407454060060787
Fp = FiniteField(p)
R = 6
b = 12
set_random_seed(int.from_bytes(b"Griffin","little"))
RC = [random_vector(Fp, b) for _ in range(R)]
```

Solutions to the CICO problem

Solutions of the CICO problem for $r \in \{5, 6, 7\}$ are presented here. $\texttt{Griffin}_r$ denotes r-round `Griffin`.

$\texttt{Griffin}_5(0, 3490692521816093, 23601145558450866,$
$12269607430774150, 9967688977125539, 6082447726448232,$
$5654276540748670, 3202643372242143, 1409612926527540,$
$1829705606091884 1, 3219981769736554, 1403954519004962)$
$= (0, 5621630451433068, 19970363721022544,$
$26741918912648639, 19340983417234439, 703450676999922,$
$28208520610521445, 25436703150515389, 27364572020087999,$
$24554342355067488, 12423823785589327, 9583319713824981)$
$\texttt{Griffin}_5(0, 20508905520120247, 23936168820965785,$
$23455122418677455, 17553236564762600, 3639182016432478,$
$17868020430519088, 3713757452078792, 5123533774106823,$
$15978370877576178, 18526890154199809, 7146019784219766)$
$= (0, 398957666284762, 14158444455164092,$
$23271855932022231, 21234778660794826, 254005063154066,$
$20580021469733731, 20943402370356289, 5582293336098692,$
$4611071270038675, 18302255515509522, 26476200309584297)$
$\texttt{Griffin}_6(0, 15940424764849354, 15734551202904841,$
$2252716448242183, 24464738475171520, 22965208929786740,$
$9160692692879124, 10856186368261439, 8227155735266026,$
$27520010287112407, 2695459349798501, 26535488466949249)$
$= (0, 28084819548111304, 27440009447766981,$
$18719426192325148, 15867588640423583, 21846560125606713,$
$6096085299560336, 3230042720230543, 8933226700213982,$
$16946922076875842, 4787108825372316, 24658818061833687)$
$\texttt{Griffin}_6(0, 13453267118668680, 16821847582014700,$
$16088365946216368, 5399506335051336, 27388226484505332,$
$23347730487451583, 17984517745797377, 8860239602618916,$
$21421104166559501, 2200106791585633, 4277034158946419)$
$= (0, 15056970863769267, 16790578228924700,$
$8672008243997393, 19887571512638539, 26391726423500753,$
$20727079056917053, 20813391870109600, 20843251064205404,$
$18441611384455618, 7490737291933204, 20355381094708976)$

$$\begin{aligned}
&\texttt{Griffin}_7(0, 471901030567494, 19368389705049758,\\
&\quad 6207658171606065, 13611402711597287, 12429039749894833,\\
&\quad 22884233714242756, 18540641300363820, 25230426657954964,\\
&\quad 17547513396056919, 15871260254068404, 5181585218256522)\\
&= (0, 24311659110177349, 24245222836079936,\\
&\quad 1360999973748497, 25535756342488541, 10834064085568113,\\
&\quad 2487598456547217, 22567275155120838, 2042666706826108,\\
&\quad 694695024982032, 10782435475712749, 20264250160050251)
\end{aligned}$$

D Bypassing the first round of Arion-π

We can use a trick similar to that used for Griffin to bypass the first round of Arion-π such that the variable x_1 is only first introduced in the second round. Unlike Griffin, this method can be applied to any number of branches in Arion-π. However, we can only bypass a single round.

Denote the input state to Arion as

$$(a_0x_0 + b_0, a_1x_0 + b_1, a_2x_0 + b_2, \ldots, a_{t-2}x_0 + b_{t-2}, 0).$$

The a_i and b_j are constants in $\mathbb{F}$ that will be determined. Once the a_i and b_j are fixed, the variable x_0 can be varied freely over $\mathbb{F}$, generating a set of input states for the CICO problem that all have input 0 to the $x^{1/\alpha}$ function in the first round. Figure 7 illustrates the evolution of one of the chosen input states up to the start of round 2.

The values of a_i and b_j can, in general, be determined as follows. After the initial matrix multiplication, all branches can be expressed as $l_i(\boldsymbol{a})x_0 + l_i(\boldsymbol{b})$ for $0 \leq i \leq t-1$, where $l_i(\boldsymbol{a})$ and $l_i(\boldsymbol{b})$ are known linear combinations. To get 0 on the last $t-2$ branches, the a_i's and b_j's need to satisfy the following linear equations

$$\begin{aligned}
l_2(\boldsymbol{a}) &= 0 & l_2(\boldsymbol{b}) &= 0\\
&\vdots & &\vdots\\
l_{t-1}(\boldsymbol{a}) &= 0 & l_{t-1}(\boldsymbol{b}) &= 0.
\end{aligned}$$

With $2t-4$ equations on $2t-2$ variables, these constraints leave two degrees of freedom for the variables in $\boldsymbol{a}$ and $\boldsymbol{b}$. Naively, one could think of additionally imposing the constraints $l_1(\boldsymbol{a}) = 0$ and $l_1(\boldsymbol{b}) = 0$ such that only the first branch is nonzero and the degree on x_0 is further reduced. However, the unique solution to this system is the trivial solution $(\boldsymbol{a}, \boldsymbol{b}) = (0, 0)$, which is not of interest. Thus, we avoid this by instead imposing arbitrary conditions for two variables a_k and b_l (as long as a_k is set to be a non-zero value to avoid the trivial solution). With a_k and b_l fixed, all the other variables get fixed, too, from the previous linear constraints. For simplicity, one could fix the values $a_0 = 1$ and $b_0 = 0$, leading to an input state of the form $(x_0, a_1x_0, a_2x_0, \ldots, a_{t-2}x_0, 0)$, where all a_i's are fixed.

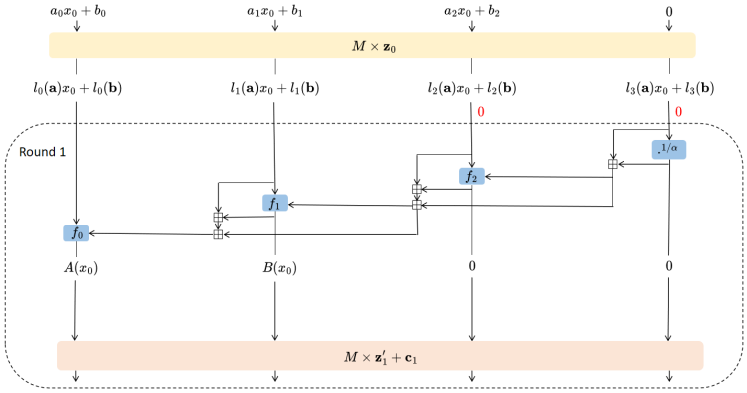


Fig. 7: Evolution of chosen set of input states to `Arion-`π with 4 branches. Red values give conditions on the a_i and b_j.

With 0 on the last $t-2$ input branches, the output of the non-linear layer of `Arion-`π will be of the form $(A(x_0), B(x_0), 0, \ldots, 0)$, where A and B are polynomials in x_0 of degree $3e$ and e, respectively. Thus, the input state from our chosen set will generate polynomials in x_0 of degree $3e$ on the branches after the affine transformation in round 1. We can then start the basic attack from there, adapting the weighted order of the variables accordingly. When the number of x_i-variables is reduced by 1 and with the degree of x_0 bounded to $3e$ until the second round, the dimension of the Gröbner basis ideal becomes smaller, which again reduces the overall attack complexity.

E Proof of Proposition 8

Proof. Recall from Definition 15 that u is defined as $u = u_0$ through the sequence $\{u_i\}_{0 \le i \le r}$. To simplify the exposition, we will work with the sequence $\{v_i\}_{0 \le i \le r}$ defined by $v_0 = 1$, and

$$v_{i+1} = v_i + 2\left\lceil \frac{v_i}{\alpha} \right\rceil, \text{ for } 0 \le i < r.$$

Note that $v_i = u_{r-i}$ and, in particular, $v_r = u$. Define two more integer sequences $\{a_i\}_{0 \le i \le r}$ and $\{b_i\}_{0 \le i \le r}$ defined by $a_0 = b_0 = 1$ and for $0 \le i < r$

$$a_{i+1} = \frac{\alpha + 2}{\alpha} a_i, \qquad b_{i+1} = \frac{\alpha + 2}{\alpha} b_i + 2.$$

As a first step, we will prove $a_i \le v_i \le b_i$. This is clearly true for $i = 0$. Supposing it holds up to some i, then using the identity $x \le \lceil x \rceil < x + 1$, we have

$$\frac{\alpha + 2}{\alpha} v_i \le v_i + 2\left\lceil \frac{v_i}{\alpha} \right\rceil < \frac{\alpha + 2}{\alpha} v_i + 2,$$

Thus, using the induction hypothesis and the definitions of $\{a_i\}$ and $\{b_i\}$,

$$a_{i+1} \leq v_{i+1} \leq b_{i+1}.$$

Observe that $a_i = \left(\frac{\alpha+2}{\alpha}\right)^i$, which proves the left-hand side of the inequality in the proposition. For the right-hand side we note that $\{b_i\}$ can be written as $b_i = (\alpha+1)\left(\frac{\alpha+2}{\alpha}\right)^i - \alpha$, when $i \geq 1$. Indeed, this can be verified for $i = 1$. Supposing it holds up to some i, then

$$\begin{aligned} b_{i+1} &= \frac{\alpha+2}{\alpha} b_i + 2 \\ &= (\alpha+1)\left(\frac{\alpha+2}{\alpha}\right)^{i+1} - \frac{\alpha(\alpha+2)}{\alpha} + 2 \\ &= (\alpha+1)\left(\frac{\alpha+2}{\alpha}\right)^{i+1} - \alpha, \end{aligned}$$

and the bounds on u stated in Proposition 8 follows. □

F FreeLunch Systems for XHash8

Description of XHash8. XHash8 is an SPN with nonlinear S-boxes, multiplication by a fixed MDS matrix M, and addition by round constants C_i. Its state contains $t = 12$ elements in $\mathbb{F}_p$ where $p = 2^{64} - 2^{32} + 1$. The rate is fixed to 8 and capacity 4. There are 3 rounds in total, and each round consists of 3 steps, for a total of 9 steps (plus the initial affine layer (I)). With the cipher state denoted as $\boldsymbol{z} = (z_0, \ldots, z_{11})$, one round of XHash8 is constructed from the following functions (excluding $(P3)^{(k)}$ which is specified below):

$$\begin{aligned} (I) &: \boldsymbol{z} \mapsto M \times (C_0 + \boldsymbol{z}), \\ (F)^{(k)} &: \boldsymbol{z} \mapsto C_{3k} + M \times (z_0^7, \ldots, z_{11}^7), \\ (B')^{(k)} &: \boldsymbol{z} \mapsto C_{3k+1} + (z_0^{\frac{1}{7}}, z_1, z_2^{\frac{1}{7}}, z_3^{\frac{1}{7}}, z_4, z_5^{\frac{1}{7}}, z_6^{\frac{1}{7}}, z_7, z_8^{\frac{1}{7}}, z_9^{\frac{1}{7}}, z_{10}, z_{11}^{\frac{1}{7}}). \end{aligned}$$

The last step of a round, $(P3)^{(k)}$, consists of naturally mapping $\boldsymbol{z}$ to a state of four elements in a cubic expansion $\mathbb{F}_{p^3}$, denoted $(S_{0,1,2}, S_{3,4,5}, S_{6,7,8}, S_{9,10,11})$, and then computing $S_{i,i+1,i+2}^7$ and mapping the result back to $\mathbb{F}_p$. After that, like with $(F)^{(k)}$, an MDS layer is applied, and the round constant C_{3k+2} is added. Effectively, $(P3)^{(k)}$ is equivalent to mapping each z_{3q+r} to a multivariate polynomial of degree 7 in $z_{3q}, z_{3q+1}, z_{3q+2}$ (see also the detailed description in [6, Appendix A]), which is the way we modelize it.

The steps are applied in the following order, from left to right:

$$(I)\ (F)^{(1)}(B')^{(1)}(P3)^{(1)}(F)^{(2)}(B')^{(2)}(P3)^{(2)}(F)^{(3)}(B')^{(3)}(P3)^{(3)}.$$

One round preceded by (I) is shown in figure 8, taken from [6].

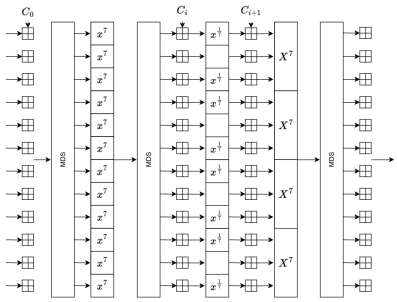


Fig. 8: Round i of `XHash8` preceded by an (I) step: $(I)(F)^{(i)}(B')^{(i)}(P3)^{(i)}$.

FreeLunch System for `XHash8` Our resolution allows us to solve the CICO problem on one branch. However, since the size of one branch is roughly 64 bits, this CICO problem could simply be solved by making 2^{64} queries to the permutation, for which our solving algorithm does not give us an advantage. On top of that, the real capacity of `XHash8` is $c = 4$ for a security claim of 128 bits. Rather than claiming a full attack on `XHash8`, we show a special case where a FreeLunch system can be easily extracted. However, the later solving steps, in particular the `polyDet` step, will still have a very high complexity.

Following the construction of FreeLunch systems from Section 3 we define the initial state as $\boldsymbol{z}_0 = (x_0, 0, \ldots, 0)$ and add a new variable $x_{i,j}$ for $0 \leq i \leq 2$ and $j \in \{0, 2, 3, 5, 6, 8, 9, 11\}$ after every $(\cdot)^{1/7}$. All other nonlinear operations can be represented as polynomials of degree 7, fixing the weights of the introduced variables to

$$\mathrm{wt}(x_0) = 1, \qquad \mathrm{wt}(x_{i,j}) = 7^{2i} + 1 \ .$$

We end up with 25 polynomials in 25 variables; 24 of these polynomials have $x_{i,j}^7$ as leading monomials and the last polynomial has $x_0^{7^6}$ as a leading monomial. The coefficient of the $x_0^{7^6}$-term in the last polynomial will be non-zero with a very high probability, ensuring we get a FreeLunch system, with $D_H = 7^{24}$ and $\alpha_0 = 7^6$.

Complexity of solving the system. We can solve the system using the algorithm described in Section 3. The complexity of `matGen` is hard to estimate precisely. The complexity of the `polyDet` step is:

$$\mathcal{O}(D_1^\omega \alpha_0 \log(\alpha_0)^2) \approx 2^{240}$$

when $\omega = 2.81$. Note that this is significantly higher than 2^{64}, the brute force complexity for solving this CICO problem.

G Implementation of the `matGen` Step

To the best of our knowledge, the `matGen` step has not been studied in the literature beyond [28]. Let ϕ_i, $i \in \{1, \dots D_1\}$, denote the elements in the standard basis of R/I that is not divisible by x_0. Then, naively, `matGen` requires the reductions of $x_0^{\alpha_0}\phi_i$ by the FreeLunch system (which is a Gröbner basis). The bound on the number of steps in a reduction by a Gröbner basis in a weighted monomial order is not clear, but experiments suggest that the complexity grows with the degree of the polynomial to reduce. In order to give an estimation of the complexity of this step, we implemented `matGen` along with the FreeLunch attack in section 4 and section 5, and benchmarked it. We implemented a variant of the naive approach: we observed that the computation of NormalForm$(x_0^{\alpha_0}\phi_i)$ can be sped up if ϕ_i is not a single variable x_i. If $\phi_i = a \times b$ (a and b being non-trivial monomials), compute NormalForm$(x_0^{\alpha_0}a)$, and then NormalForm(NormalForm$(x_0^{\alpha_0}a)b)$. The intermediary normal form corresponds to another columns of T_0 and can be considered free if the columns of T_0 are computed in the right order. In our implementations, we chose $b = x_i$ with α_i as low as possible; this seemed to be the fastest approach.

Article III

5.3 Zeroed Out: Cryptanalysis of Weak PRFs in Alternating Moduli

Irati Manterola Ayala and Håvard Raddum

Published in *IACR Transactions on Symmetric Cryptology, 2025*, vol 2, pages 1-15. To be presented at *FSE* in March 2026.
https://doi.org/10.46586/tosc.v2025.i2.1-15

Version reproduced here in Cryptology ePrint Archive, Report 2024/2055, 2024.
https://eprint.iacr.org/2024/2055

Zeroed Out: Cryptanalysis of Weak PRFs in Alternating Moduli

Irati Manterola Ayala[1] and Håvard Raddum[2]

[1] Simula UiB, Bergen, Norway, irati@simula.no
[2] Simula UiB, Bergen, Norway, haavardr@simula.no

Abstract. The growing adoption of secure multi-party computation (MPC) has driven the development of efficient symmetric key primitives tailored for MPC. Recent advances, such as the alternating moduli paradigm, have shown promise but leave room for cryptographic and practical improvements. In this paper, we analyze a family of weak pseudorandom functions (wPRF) proposed at Crypto 2024, focusing on their One-to-One parameter sets. We demonstrate that these configurations fail to achieve their intended one-to-one mappings and exploit this observation to develop an efficient key recovery attack.
Our analysis reveals critical vulnerabilities, reducing the complexity of key recovery to $\mathcal{O}(2^{\lambda/2}\log_2\lambda)$ for the Standard One-to-One wPRF and $\mathcal{O}(2^{0.84\lambda})$ for the Reversed Moduli variant – both substantially below their claimed λ-bit security. We validate our findings through experimental evaluation, confirming alignment between predicted and observed attack complexities.

Keywords: Multi-Party Computation · Weak pseudorandom functions · Alternating moduli paradigm · Symmetric cryptanalysis · Key recovery attack

1 Introduction

The rise of interest in secure multi-party computation (MPC) and the growing threat of quantum computers have created an urgent need for efficient and quantum-resistant symmetric key primitives specifically designed for use in MPC settings. While classic symmetric key primitives hold promise due to their simplicity and performance potential, existing constructions were developed for different (and usually incompatible) settings. This creates a pressing need for new designs that avoid such vulnerabilities while remaining suitable for MPC applications.

Important cryptographic tasks, such as ring signatures, oblivious pseudorandom functions (OPRFs), verifiable random functions (VRFs), and blind signatures, require efficient solutions tailored to these evolving challenges [RST01, FIPR05, NR97, MRV99, Cha82]. Ideally, these primitives should be evaluable in a single round of communication using linear secret-sharing techniques. While there has been progress in adapting existing symmetric key primitives for MPC [AGP+19, ARS+16, DEG+18, DGH+21, GØSW22, GRR+16], many constructions still require too many communication rounds or involve high overheads [BIP+18]. This inefficiency stems in part from the difficulty of balancing low-depth functions, which are essential for efficiency in MPC settings, with security requirements.

To address these issues, Boneh et al. [BIP+18] introduced the alternating moduli paradigm, separating the requirements for MPC efficiency from those for cryptographic security. By alternating linear operations over different moduli, they built a depth-2 weak pseudorandom function (wPRF) that can be securely evaluated in a single communication

round after preprocessing. Dinur et al. [DGH+21] extended this work by introducing new one-way functions (OWFs), pseudorandom generators (PRGs), and wPRFs within the same framework. They showed that their OWF could be used to build a post-quantum signature scheme with good efficiency. Despite these advances, the protocols built around these constructions often fell short of state-of-the-art performance. Moreover, the two-party computation (2PC) protocols for these constructions require significant preprocessing time to generate correlated randomness, with communication overheads remaining higher than optimal.

Building on this line of work, Alamati et al. [APRR24] revisited the alternating moduli paradigm to propose a new wPRF that improves on previous constructions in terms of efficiency and practicality. According to the authors, their design significantly reduces communication and computational costs, particularly in the main evaluation phase, and minimizes the need for oblivious transfers. In terms of cryptanalysis, they argue that the security of their wPRF depends on the hardness of solving sparse multivariate polynomial systems over $\mathbb{F}_3$ or, in the dual form, on sparse multilinear interpolation. This argument is used by the authors to justify their focus on subset-sum attacks as the primary cryptanalytic threat. However, our analysis shows that this focus may be too narrow, as other potential attack vectors remain relevant and deserve further attention.

Our Contributions. In this paper, we present cryptanalysis of the One-to-One parameter sets proposed by Alamati et al. for their alternating moduli wPRF. We show that these do not provide the approximately one-to-one mappings they were designed to achieve. Leveraging this observation, we present a novel key recovery attack against the Standard One-to-One parameter set of the wPRF. Our attack achieves a complexity of $\mathcal{O}(2^{\lambda/2}\log_2\lambda)$, significantly lower than the claimed λ-bit security level.

Next, we adapt the key recovery attack to the Reversed Moduli One-to-One parameter set. While this variant introduces additional challenges, our modified attack successfully recovers the key with a complexity of $\mathcal{O}(2^{0.84\lambda})$, once again breaking the claimed 2^λ security level. We have also considered the Many-to-One parameter sets but did not find any successful attacks against these variants.

We provide both theoretical complexity analyses and experimental verification of our attacks. Our experiments confirm that the observed attack complexities closely align with the theoretical predictions. Beyond identifying these vulnerabilities, we propose potential countermeasures to mitigate these attacks, aiming to enhance the security of future designs.

Outline of this Paper. This paper is structured as follows. In Section 2, we provide the necessary preliminaries, namely the definition and security notions of weak pseudorandom functions, as well as the classical and generalized birthday paradox. Section 3 presents the wPRF construction by Alamati et al., detailing its specification, variants, and recommended parameter sets. In Section 4, we describe our primary contributions, offering a comprehensive cryptanalysis of two of the proposed wPRF parameter sets and analyzing the theoretical complexity of our key recovery attack. Section 5 validates our theoretical analysis with experimental results, showcasing the feasibility and accuracy of our approach. Finally, in Section 6, we conclude by summarizing our findings, discussing potential countermeasures, and outlining open problems for future research.

2 Preliminaries

In this section, we present the foundational concepts necessary for understanding the results and analysis in this paper. These include the definition and security notions of weak pseudorandom functions, along with the classical and generalized forms of the birthday paradox.

2.1 Weak Pseudorandom Functions

The definition of a weak pseudorandom function below follows Definition 2.1 from [BIP+18].

Definition 1. A **weak pseudorandom function (wPRF)** is a keyed function $f : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ that, when queried on random inputs $x \in \mathcal{X}$, is computationally indistinguishable from a truly random function. More formally, for a randomly selected key $k \in \mathcal{K}$, the output $f(k, x)$ for x sampled uniformly at random from $\mathcal{X}$ is indistinguishable from the output $g(x)$ of a random function $g : \mathcal{X} \to \mathcal{Y}$ to any adversary running in time $t(\lambda)$ with access to an oracle for f.

The distinction between a *weak PRF* and a *strong PRF* lies in the adversarial query model: wPRFs restrict adversaries to query only random inputs, whereas strong PRFs permit the adversary to query adaptive, chosen inputs.

Security Notion of a wPRF. The security of a wPRF f is quantified by the advantage an adversary $\mathcal{A}$ running in time $t(\lambda)$ has in distinguishing $f(k, x)$ from a random function. We say that f is *secure* if

$$\mathrm{Adv}^{\mathrm{wPRF}}_{f,\mathcal{A}} = \left|\Pr[\mathcal{A}^{f(k,\cdot)} = 1] - \Pr[\mathcal{A}^{g(\cdot)} = 1]\right| \leq \epsilon(\lambda),$$

where $\epsilon(\lambda)$ is negligible in λ. If a wPRF claims to provide λ-*bit security*, it means that the above security notion holds when $t(\lambda) = 2^\lambda$. That is, the above advantage remains negligible even when $\mathcal{A}$ is allowed up to 2^λ queries and runs in time 2^λ.

2.2 The Birthday Paradox

The *birthday paradox* is a probabilistic phenomenon that explains the counterintuitive likelihood of repeated outcomes when drawing samples from a finite set. It is particularly relevant in cryptographic contexts, where it is used to estimate the probability of repeated outputs in hash functions and similar structures.

Given a function that maps inputs to $|\mathcal{Y}|$ equally likely outputs, the birthday paradox quantifies the number of samples required to observe the same output at least twice.

Lemma 1. *[CLR90, Sec. 5.4.1]* ***Classical Birthday Paradox****. For a uniform random distribution over $|\mathcal{Y}|$ possible outputs, the expected number of samples S required to observe the first repeated outcome is:*

$$S \approx \sqrt{2|\mathcal{Y}|}.$$

The analysis given in [CLR90] naturally extends to estimating the number of random samples needed to find multiple pairs of repeated outcomes. In this paper, we refer to this as the *generalized birthday paradox*, noting that it differs from other common generalizations [Wag02, Das05].

Lemma 2. ***Generalized Birthday Paradox****. For a uniform random distribution over $|\mathcal{Y}|$ possible outputs, the expected number of samples S required to observe c pairs of repeated outcomes is:*

$$S \approx \sqrt{2|\mathcal{Y}|c}.$$

The generalized form reveals that the sample complexity scales proportionally to $\sqrt{c}$, meaning that detecting more collisions requires only a sublinear increase in samples.

3 A New Weak PRF

At Crypto 2024, Alamati et al. [APRR24] introduced a novel wPRF tailored for efficient multiparty computation (MPC) applications. This construction builds upon and generalizes the alternating-moduli paradigm initially proposed by Boneh et al. [BIP+18]. By alternating computations over two distinct moduli, typically $\mathbb{F}_2$ followed by $\mathbb{F}_3$, this approach has demonstrated significant potential in achieving both simplicity and efficiency in advanced cryptographic protocols.

We explore the details of Alamati et al.'s new wPRF constructions, and discuss their recommended parameter sets for achieving λ-bit security under various constraints.

3.1 Specification.

In their work [BIP+18], Boneh et al. considered the function

$$f(\mathbf{K}, x) := g(\mathbf{K} \cdot_2 x), \quad \text{where } g(w) = \sum_i w_i \mod 3.$$

Here, the operation $\cdot_p$ denotes multiplication modulo p, the matrix $\mathbf{K} \in \mathbb{F}_2^{m \times n}$ is the secret key and the term $\mathbf{K} \cdot_2 x \in \mathbb{F}_2^m$ is embedded into $\mathbb{F}_3^m$ component-wise in the natural way. Extensions to this idea defined the wPRF

$$f(\mathbf{K}, x) := \mathbf{B} \cdot_3 (\mathbf{K} \cdot_2 x),$$

where $\mathbf{K}$ is a square matrix and $\mathbf{B}$ is a compressing matrix.

To improve upon Boneh et al.'s construction, Alamati et al. propose a new wPRF that optimizes the end-to-end cost of MPC protocols while enhancing performance during the main computation phase, leading to significant gains in both communication complexity and computational efficiency. Their construction relies on three core components:

1. Non-linear combination of the input and key modulo two.
2. Matrix multiplication modulo two.
3. Natural modulus conversion followed by a public compressing linear map $\mathbf{B}$.

3.2 Definition of the Standard wPRF.

The proposed standard $(\mathbb{F}_2, \mathbb{F}_3)$-wPRF is formulated as follows:

$$F(k, x) := \mathbf{B} \cdot_3 (\mathbf{A} \cdot_2 [k \odot_2 x]),$$

where:

- $x, k \in \mathbb{F}_2^n$ are random vectors representing the input and key,
- $\mathbf{A} \in \mathbb{F}_2^{m \times n}$ is a random matrix,
- $\mathbf{B} \in \mathbb{F}_3^{t \times m}$ is a random compressing matrix (i.e., $t < m$).

Here, the operation $\odot_p$ denotes component-wise multiplication modulo p.
A visual representation of the standard wPRF construction is given in Fig. 1.

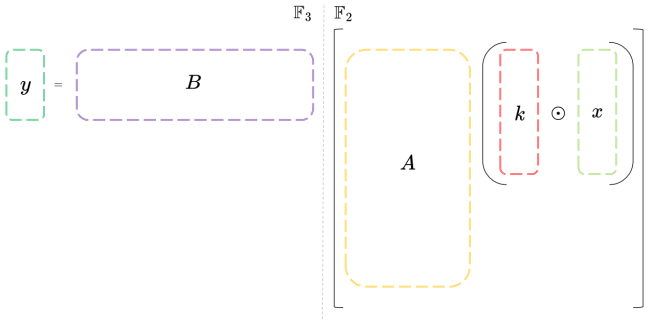


Figure 1: Construction of the standard $(\mathbb{F}_2, \mathbb{F}_3)$-wPRF.

Variants of the Standard wPRF. The generalized $(\mathbb{F}_p, \mathbb{F}_q)$-wPRF extends this concept to arbitrary primes p and q in a straightforward manner as

$$F(k, x) := \mathbf{B} \cdot_q (\mathbf{A} \cdot_p [k \odot_p x]),$$

where $x, k \in \mathbb{F}_p^n$, $\mathbf{A} \in \mathbb{F}_p^{m \times n}$, and $\mathbf{B} \in \mathbb{F}_q^{t \times m}$.
For scenarios requiring binary secret-sharing outputs, Alamati et al. propose the Reversed Moduli $(\mathbb{F}_3, \mathbb{F}_2)$-wPRF:

$$F(k, x) := \mathbf{B} \cdot_2 (\mathbf{A} \cdot_3 [k \odot_3 x]),$$

where the roles of the moduli are reversed.

3.3 Parameters.

Table 2 summarizes the recommended parameter sets from [APRR24] across the different wPRF constructions. Alamati et al. divide the parameter sets into two groups, namely One-to-One and Many-to-One parameters. The authors assert that each parameter set achieves λ-bit security.

One-to-One Parameters. The One-to-One parameter set is designed to provide a (roughly) one-to-one mapping between inputs and outputs. Specifically, the input and output spaces are of the same size, and for any given input x, the authors claim we can expect a unique corresponding output y. This setup represents their most conservative alternative.

Many-to-One Parameters. As the name suggests, the Many-to-One parameter set has a larger input space than output space. This means that for any given output y, there should be multiple input values x mapping to y, leading to a many-to-one mapping between input and output values.

4 Our Attack

In this section, we present a key recovery attack against the two One-to-One parameter sets proposed by the authors. Our method exploits weaknesses in these parameter sets, and efficiently identifies key bits using collisions. The attack is able to recover the key in $\mathcal{O}(2^{\lambda/2} \log_2 \lambda)$ calls to the wPRF in the standard version, and in $\mathcal{O}(2^{0.84\lambda})$ calls in the reversed moduli variant, demonstrating a significant reduction in complexity compared to

Table 1: Recommended parameter sets for the wPRF for λ-bit security.

Variant	*One-to-One*			*Many-to-One*		
	n	m	t	n	m	t
$(\mathbb{F}_2, \mathbb{F}_3)$-wPRF	2λ	7.06λ	$\frac{2\lambda}{\log_2(3)}$	4λ	2λ	$\frac{\lambda}{\log_2(3)}$
$(\mathbb{F}_3, \mathbb{F}_2)$-wPRF	$\frac{2\lambda}{\log_2(3)}$	$\frac{7.06\lambda}{\log_2(3)}$	2λ	$\frac{4\lambda}{\log_2(3)}$	2λ	λ

the claimed 2^λ calls. We begin with an analysis of the Standard One-to-One wPRF to establish the basic methodology. Following this, we demonstrate how the attack can be modified for the Reversed Moduli variant, overcoming its additional complexities.

In the following, let X denote the input space of the wPRF, let M denote the output space of the multiplication with the matrix $\mathbf{A}$, and let Y denote the output space of the wPRF.

One-to-One? We target the proposed parameter sets where the input space size is $|X| = 2^{2\lambda}$ $(= 3^{(2\lambda/\log_2 3)})$, the intermediate space has size $|M| = 2^{7.06\lambda}$ $(= 3^{(7.06\lambda/\log_2 3)})$, and the output space size is $|Y| = 3^{2\lambda/\log_2 3} = 2^{2\lambda}$. The authors argue that these configurations result in a (roughly) one-to-one mapping between inputs and outputs of the wPRF. However, this assumption does not hold once the wPRF is instantiated with a fixed key k. We exploit this observation to construct a key recovery attack.

- *Standard One-to-One.* Define h_1 as the Hamming weight of k, and let $h_0 = 2\lambda - h_1$ denote the number of zeros in k. For a key k chosen uniformly at random, we expect $h_1 \approx h_0 \approx \lambda$, following a binomial distribution. In positions where $k_i = 0$, the value of x_i is irrelevant, as $k_i \odot x_i$ will always equal zero. This leads to 2^{h_0} distinct values of x producing the same input to the multiplication with $\mathbf{A}$, creating a 2^{h_0}-to-1 sub-mapping in the wPRF. Consequently, once the key is fixed, the wPRF becomes a 2^{h_0}-to-1 mapping. The image of the wPRF F, denoted as $im(F)$, thus has size $2^{h_1} \approx 2^\lambda$ instead of $2^{2\lambda}$.

- *Reversed Moduli One-to-One.* Extending notation, let h_i^* be the number of elements in k that take the value i (for $i = 0, 1, 2$). For a uniformly random key k, we expect $h_0^* \approx h_1^* \approx h_2^* \approx \frac{2\lambda}{3\log_2(3)}$. Similarly to the standard case, the operation $k \odot x$ induces a $3^{h_0^*}$-to-1 sub-mapping, which can be expressed as $2^{2\lambda - \log_2(3)(h_1^* + h_2^*)}$-to-1. As a result, $im(F)$ has size $2^{\log_2(3)(h_1^* + h_2^*)}$, with an expected value of $2^{4\lambda/3}$, instead of the intended $2^{2\lambda}$.

4.1 Key Recovery Attack on Standard One-to-One wPRF

Our attack aims to recover the key k by finding pairs x, x' such that $F(k, x) = F(k, x')$. Whenever this occurs we say we have a *collision*. The attack is described in Algorithm 1 and explained in the following.

We initialize a key K as $K = [1, 1, 1, \ldots, 1]$ and iteratively refine it towards the correct key k by identifying positions in k that must be 0. The idea is to query the wPRF on random inputs, building up a table of input and output values (x, y). By the birthday paradox (see Lemma 1), collisions are expected to appear after approximately $\sqrt{2|im(F)|} = 2^{(h_1+1)/2}$ samples.

Let x and x' be two inputs producing the same output y. If $x_i \neq x'_i$, then it must hold that $k_i = 0$. To understand this, note that $|M| = 2^{7.06\lambda}$ is much larger than $|X| = 2^{2\lambda}$, and only 2^{h_1} different values go into multiplication with $\mathbf{B}$, which is much smaller than $|Y| = 2^{2\lambda}$. Thus, the probability of creating collisions *after* multiplying $k \odot x$ with $\mathbf{A}$ becomes negligible. Therefore, with overwhelming probability, the only source of collisions is the 2^{h_0}-to-1 mapping of $k \odot x$. So, if $x_i \neq x'_i$, then k_i must be zero, as the differing input bits would otherwise result in different values in M (and therefore a different output y).

To further support this and address potential false positives, we observe that both the input and output spaces have cardinalities $|X| = |Y| = 2^{2\lambda}$, so the probability that a difference in input results in the same output is analogous to finding a collision in a random mapping from a space of size $2^{2\lambda}$ to itself. By the birthday bound, the probability of any collision after I random inputs is approximately $I^2/2^{2\lambda+1}$. In our setting, we will later show that the attack requires approximately $I \approx 2^{\lambda/2}$ such queries, yielding a collision probability of $1/2^{\lambda+1}$, which is exponentially small in λ. This makes false positives—cases where $x_i \neq x'_i$ but $k_i = 1$—highly improbable, so each collision reveals with high confidence that the differing positions in x and x' correspond to zero entries in k.

To further analyze the key recovery, let

$$J_0 = \{i | k_i = 0\} \text{ and } J_1 = \{i | k_i = 1\}.$$

For two colliding inputs x, x', let $X_= = X_=(x, x') = \{i | x_i = x'_i\}$ and $X_\neq = X_\neq(x, x') = \{i | x_i \neq x'_i\}$.

As collisions accumulate, we progressively update K by changing 1-bits in K to 0 for all indices in $X_\neq$. For each collision, we know that $J_1 \subseteq X_=$ and $X_\neq \subseteq J_0$. For positions $i \in J_0$, we have either $x_i = x'_i$ or $x_i \neq x'_i$ with equal probability since both x and x' are drawn uniformly at random. Consequently, we expect that only half of the set J_0 will be revealed from any one collision. Thus, each new independent collision is expected to reveal half of the previously unrevealed positions where $k_i = 0$. This suggests that the Hamming distance between K and k is halved with each new collision. Specifically, the expected Hamming distance after c collisions can be expressed as

$$d_c = h_0/2^c. \tag{1}$$

4.1.1 Collision Saturation Point.

As the attack progresses and additional collisions are found, the rate of discovering new J_0 positions decreases, as many zeros in k have already been determined. At a certain point, it becomes more efficient to perform an exhaustive search among keys within a small Hamming distance of the current guess K. The transition occurs when the expected cost of generating the $(c+1)$-th collision surpasses the cost of exhaustive search among vectors with hamming distance at most d_c from K.

Cost of New Collision. By the generalized birthday paradox (see Lemma 2), the expected number of samples required to find c collisions is $\sqrt{2^{h_1+1}c}$. To find the $(c+1)$-th collision after already obtaining c collisions, the number of new queries needed is

$$\sqrt{2^{h_1+1}(c+1)} - \sqrt{2^{h_1+1}c} = 2^{(h_1+1)/2}(\sqrt{c+1} - \sqrt{c}). \tag{2}$$

The total cost of generating the $(c+1)$-th collision is dominated by this term, since verifying collisions can be done in constant time by storing (x, y)-pairs in a hash table.

Cost of Exhaustive Search. For exhaustive search, we consider all keys within a Hamming distance of at most d_c from K. Notably, we only flip 1-bits in K to 0, never changing

0-bits to 1. This restricted search space is referred to as the *one-sided Hamming distance* from K. Thus, the number of candidate keys to search is $\sum_{j=1}^{\lceil d_c \rceil} \binom{H_1}{j}$, where H_1 is the current Hamming weight of K. To verify each key candidate, we compute an output using the current key guess and a previously queried input. Since the number of candidate keys remains below $2^{(\lambda+1)/2} \cdot (\sqrt{c+1} - \sqrt{c}) < 2^{(\lambda+1)/2}$ (see Inequality 3 below), the probability of an incorrect key producing the expected output is negligibly small. This should guarantee that only the correct key passes with very high probability. In the worst case, this results in a total query cost of

$$\sum_{j=1}^{\lceil d_c \rceil} \binom{H_1}{j}.$$

To determine the optimal transition point to exhaustive search, we substitute the expected values $h_0 = h_1 = \lambda$ into Equations (2) and (1). Minimizing the total attack cost requires switching strategies once c collisions have been found and the following inequality holds:

$$\sum_{j=1}^{\lceil \lambda/2^c \rceil} \binom{H_1}{j} < 2^{(\lambda+1)/2} \cdot (\sqrt{c+1} - \sqrt{c}). \tag{3}$$

This condition does not guarantee that the correct key lies within d_c Hamming distance of K. If exhaustive search fails to find the correct key at this stage, we simply identify one more collision and retry.

4.1.2 Complexity Analysis.

We measure the complexity of the attack in terms of the needed number of queries to the wPRF. The attack follows a two-phase approach: first, collisions are accumulated until reaching the transition point; then, exhaustive search on the key is applied. Let C denote the number of collisions at the transition point. We know that $C \leq \log_2 \lambda$, since for $C = \log_2 \lambda$ and $H_1 \leq n = 2\lambda$ Inequality (3) always holds for $\lambda \geq 17$.

As discussed above, the total number of samples required to recover the key is approximately

$$\sqrt{2^{\lambda+1} C} + \sum_{j=1}^{\lceil \lambda/2^C \rceil} \binom{H_1}{j},$$

where H_1 represents the Hamming weight of the guessed key after C collisions. By construction of K, we estimate H_1 as $h_1 + \frac{h_0}{2^C} \approx \lambda + \frac{\lambda}{2^C}$.

For $C = \log_2 \lambda$, the sum in the expression above stops at $j = 1$, leading to an attack complexity of the order

$$\mathcal{O}\left(2^{\lambda/2} \log_2 \lambda\right).$$

The total cost of the attack is thus significantly lower than 2^λ, demonstrating a clear compromise of the claimed security level. We explicitly note that this complexity analysis assumes $h_1 = \lambda$, which corresponds to the expected Hamming weight of a uniformly random key. While this assumption provides a realistic estimate of the computational cost for a typical key, we acknowledge that the actual complexity may vary slightly for specific instances where the key deviates significantly from the expected Hamming weight.

Algorithm 1 Key Recovery Attack

Require: Input-output oracle $\mathcal{O}$, security parameter λ
Ensure: Recovered key k

$K \leftarrow [1, 1, \ldots, 1]$
$\mathcal{P} \leftarrow \emptyset$
$c \leftarrow 0$
$H_1 \leftarrow n$
while Correct key not found **do**
  repeat
    Collect a new input-output pair (x, y) using $\mathcal{O}$
    if $(x', y) \in \mathcal{P}$ for some $x' \neq x$ **then**
      for $i \in X_{\neq}$ **do**
        if $K_i = 1$ **then**
          $K_i \leftarrow 0$
          $H_1 \leftarrow H_1 - 1$
        end if
      end for
      $c \leftarrow c + 1$
    end if
    Add (x, y) to $\mathcal{P}$
  until Collision is found
  if $\sum_{j=1}^{\lceil \lambda / 2^c \rceil} \binom{H_1}{j} \leq 2^{(\lambda+1)/2} \cdot (\sqrt{c+1} - \sqrt{c})$ **then**
    for each k' with one-sided Hamming distance at most d_c from K **do**
      if k' matches an input-output pair from $\mathcal{P}$ **then**
        return k'
      end if
    end for
  end if
end while

4.2 Attack on Reversed Moduli One-to-One wPRF

We adapt the collision-based key recovery attack methodology used in the standard parameter set to the reversed moduli One-to-One wPRF. The key difference in this variant is that non-zero key positions can take two distinct values, requiring modifications to our approach. The modified attack still remains feasible and reveals vulnerabilities in the construction. Below, we detail the process and analyze its computational complexity.

4.2.1 Collisions: Identifying Zero Key Positions.

The first step of the attack is to identify the positions in the key k where $k_i = 0$. To achieve this, we employ the same collision-finding method used previously in the standard case. By the birthday paradox (see Lemma 1), we expect collisions to appear after collecting approximately

$$\sqrt{2|im(F)|} = 2^{(\log_2(3)(h_1^*+h_2^*)+1)/2} \approx 2^{(4\lambda+3)/6}$$

samples.

In this setting, the size of the domain M is again significantly larger than the size of the input space X, ensuring that collisions arise solely from the $3^{h_0^*}$-to-1 mapping induced by $k \odot x$ with overwhelming probability. Therefore, each collision reveals information about positions in k where $k_i = 0$.

Let $J_0 = \{i \mid k_i = 0\}$ and x and x' two colliding inputs as before. We again have

$$X_{\neq} = X_{\neq}(x, x') = \{i | x_i \neq x'_i\} \subseteq J_0.$$

For positions $i \in J_0$, we have either $x_i = x'_i$ or $x_i \neq x'_i$, but these events do not occur with equal probability in the reversed moduli case. Since x takes values in $\mathbb{F}_3$, we have $x_i \neq x'_i$ with probability $2/3$. Thus, we expect to recover approximately $2/3$ of J_0 from any given collision. This higher recovery rate, compared to the standard wPRF, reduces the number of collisions required to fully determine J_0.

We continue generating collisions until all zero positions in the key are likely identified. To estimate the number of collisions required, we analyze the probability of revealing additional zeroes as we accumulate collisions. As discussed, the first collision is expected to reveal approximately $2/3 \cdot h_0^*$ zeroes. The second collision builds upon this, revealing another $2/3^2 \cdot h_0^*$ zeroes. More generally, after c collisions, the total number of recovered zeroes is

$$\sum_{i=1}^{c} \frac{2}{3^i} \cdot h_0^*.$$

Thus, the number of remaining zero positions in k yet to be identified after c collisions is given by

$$h_0^* - \sum_{i=1}^{c} \frac{2}{3^i} \cdot h_0^* = \left(1 - \sum_{i=1}^{c} \frac{2}{3^i}\right) h_0^*.$$

To ensure all zero positions are likely identified, the number of remaining positions must be less than 1, i.e.,

$$\left(1 - \sum_{i=1}^{c} \frac{2}{3^i}\right) h_0^* \approx \left(1 - \sum_{i=1}^{c} \frac{2}{3^i}\right) \frac{2\lambda}{3\log_2(3)} < 1.$$

Solving this expression for c gives the minimum number of expected collisions required to recover all zero positions in the key. To ensure high-probability recovery, we introduce a small safety margin by multiplying the derived value for c by three. In any case, the complexity of determining all zero positions remains of the order $\mathcal{O}(\log_3(\lambda))$ collisions.

4.2.2 Exhaustive Search over Non-Zero Key Positions.

Once the positions in J_0 are determined, the values of the remaining positions $J_1 \cup J_2 = \{i \mid k_i \in \{1, 2\}\}$ remain unknown. These positions are expected to constitute $2/3$ of the key. However, for these positions, each k_i can only take one of two possible values, 1 or 2, since all zeroes have already been detected. For a key of length $n = \dfrac{2\lambda}{\log_2 3}$, the total number of candidates for the remaining key components is therefore

$$2^{(2/3)\cdot(2\lambda/\log_2 3)} \approx 2^{0.84\lambda}.$$

The correctness of any candidate key can be verified by querying the wPRF on an input-output pair as before. Thus, the exhaustive search over all possible keys in $J_1 \cup J_2$ requires at most $2^{0.84\lambda}$ queries.

4.2.3 Complexity Analysis.

The overall complexity of the attack consists of two main components: identifying zero positions via collisions and performing an exhaustive search over non-zero key positions.

Collision Complexity. By the generalized birthday paradox (see Lemma 2), the expected cost of finding enough collisions to identify all zero positions of the key is

$$\sqrt{2^{(4\lambda+3)/3} C},$$

where $C = \mathcal{O}(\log_3(\lambda))$ denotes the number of collisions required to fully determine J_0. The total complexity of this step thus becomes $\mathcal{O}(2^{2\lambda/3} \log_3(\lambda))$.

Exhaustive Search Complexity. Once the zero positions are known, the exhaustive search requires testing $2^{0.84\lambda}$ key candidates, each verified with a query. This results in a total cost of $2^{0.84\lambda}$.

Total Complexity. The overall complexity of the attack is the sum of the costs of the collision and exhaustive search steps. Notably, the complexity is dominated by the exhaustive search step, and so the attack has a total cost of $\mathcal{O}(2^{0.84\lambda})$. This is well below the claimed security level of 2^λ, demonstrating that the Reversed Moduli One-to-One parameter set also fails to provide the intended security guarantees.

4.3 Applicability beyond the One-to-One Parameter Sets

Many-to-One Parameter Sets. The attack described above does not apply to the Many-to-One variants of the wPRF. In these cases, the input space size $|X| = 2^{4\lambda}$ significantly exceeds the output space size $|Y| = 2^\lambda$, making collisions unavoidable. Since the intermediate output space of the pointwise multiplication followed by multiplication with $\mathbf{A}$ has size $|M| = 2^{2\lambda}$, most collisions occur independently of the term $k \odot x$. More specifically, distinct points in M produce a collision in Y at a rate of once every $2^{\lambda/2}$ queries, while collisions due to $k \odot x$ being a $2^{2\lambda}$-to-1 mapping only appear at a rate of once every 2^λ queries. As a result, generating even a single collision where $k \odot x = k \odot x'$ will take $\mathcal{O}(2^\lambda)$ time, making our approach ineffective for these parameter sets.

Boneh et al.'s wPRF. Recall that Alamati et al.'s construction builds upon the wPRF derived by Boneh et al.'s alternating-moduli function [BIP+18]. This wPRF is defined as $f(\mathbf{K}, x) := \mathbf{B} \cdot_3 (\mathbf{K} \cdot_2 x)$, where $\mathbf{K}$ is a square matrix acting as the secret key, and x is an input vector. Unlike Alamati et al.'s wPRF, which employs component-wise

multiplication $k \odot x$ before a linear transformation, Boneh et al.'s construction instead performs a full matrix-vector multiplication $\mathbf{K} \cdot_2 x$ directly. This difference has a crucial impact on the applicability of our attack. In Alamati et al.'s wPRF, zero entries in k eliminate contributions from corresponding positions in x, effectively reducing the entropy of the input to subsequent computations and enabling the attack. However, in Boneh et al.'s construction, each x_i in the input x is multiplied with n different bits of the key $\mathbf{K}$, ensuring that no input component is entirely zeroed out due to a zero entry in $\mathbf{K}$. Consequently, the structure that enables our attack in Alamati et al.'s scheme is absent here, making our approach ineffective against Boneh et al.'s wPRF.

5 Experimental Verification

To validate our proposed approach, we have conducted a series of low-scale experiments[1] in the Standard One-to-One parameter set, using $\lambda = 28$ and $\lambda = 34$ as test cases. For each scenario, we have performed 1000 independent experiments to ensure statistical significance, recording the average results obtained. Table 2 summarizes our experimental findings, which corroborate the theoretical estimations presented in Section 4 and demonstrate the feasibility of a successful key recovery attack.

We analyze the average complexity of the two principal components outlined in Section 4: finding collisions and exhaustive search.

- **Collision Finding ($C_{\mathbf{col}}$):** We measure the average number of samples required to generate a sufficient number of collisions necessary for key recovery.
- **Exhaustive Search ($C_{\mathbf{exs}}$):** Once the sufficient number of collisions is identified, we perform an exhaustive search over the key candidates with a small Hamming distance from the current key guess. We record the average number of calls to F for this step.

By combining these components, we compute the total complexity $C_{\text{tot}} = C_{\text{col}} + C_{\text{exs}}$ of the attack. Our results demonstrate that we achieve key recovery with complexity closely aligned with the theoretical expectation of $\mathcal{O}(2^{\lambda/2} \log_2(\lambda))$:

- For $\lambda = 28$, the observed average total complexity is $C_{\text{tot}} = 2^{16.6}$, which is consistent with the estimated complexity of $2^{\lambda/2} \log_2(\lambda) = 2^{16.27}$.
- For $\lambda = 34$, the observed average total complexity is $C_{\text{tot}} = 2^{19.82}$, closely matching the estimated complexity of $2^{\lambda/2} \log_2(\lambda) = 2^{19.35}$.

Note that all 1000 experiments recovered the correct key, and that the numbers used to calculate C_{col} and C_{exs} represent the total number of calls to the wPRF oracle, including instances where the attack had to go back and find an additional collision before trying exhaustive search again.

Additionally, we evaluate the accuracy of the transition step discussed in Section 4, which estimates the optimal transition point between collision finding and exhaustive search. Specifically, we measure the success rate of the computed transition point C from Inequality 3 by verifying whether, after finding C collisions, the key is successfully recovered on the first attempt at exhaustive search. The measured success rate is 76.6% for $\lambda = 28$ and 88.8% for $\lambda = 34$, indicating that the theoretical model becomes increasingly accurate for larger values of λ.

Furthermore, we report the average number of collisions required to recover the full key. Our results show that the attack requires approximately 4.39 collisions for $\lambda = 28$

[1]The implementation details and source code are available at `https://github.com/Simula-UiB/wPRF-Collision-Attack`.

and 4.19 collisions for $\lambda = 34$ to achieve full key recovery. We would expect the number of necessary collisions to increase for higher values of λ, as a higher λ corresponds to a higher Hamming weight of the key on average, requiring more bits to be flipped to 0 to reach the final correct guess of the key. However, our experiments indicate that this is not necessarily the case. This discrepancy may be attributed to the significant differences in the accuracy of the computed transition point C. For $\lambda = 28$, fewer collisions should, in theory, have been required before successfully switching to exhaustive search. Nonetheless, due to inaccuracies in approximating the transition point correctly in nearly 25% of the cases, more collisions were needed than expected. As the accuracy of C improves with higher values of λ, we observe fewer such deviations. We therefore hypothesize that this theoretical trend persists for higher values of λ, where the discrepancy in the transition point accuracy is likely to diminish further, thereby reducing unexpected variations in the number of collisions needed to recover the key through exhaustive search.

Table 2: Summary of experimental results for $\lambda = 28$ and $\lambda = 34$. The columns represent the average complexity of collision finding (C_{col}), exhaustive search (C_{exs}), and total complexity (C_{tot}). Additionally, the table reports the average number of collisions required to achieve full key recovery and the success rate of the transition point C estimated using Inequality 3. All values are averaged over 1000 independent experiments.

λ	C_{col}	C_{exs}	C_{tot}	# Collisions	Accuracy of C (%)
28	$2^{16.6}$	$2^{7.64}$	$2^{16.6}$	4.39	76.6
34	$2^{19.82}$	$2^{10.88}$	$2^{19.82}$	4.19	88.8

5.1 Hamming Distance Analysis

In addition to the previously described experiments, we also verify the assumption that the Hamming distance between the actual key k and the guessed key K is approximately halved with each new collision.

We have performed 100 independent experiments for various values of λ and recorded the average results. While the findings are consistent across different values of λ, we present the case of $\lambda = 34$ as a representative example. Figure 2 illustrates the average decrease in Hamming distance between the current guessed key and the actual key after each identified collision. The graph shows that the Hamming distance roughly halves with each collision, as expected.

6 Conclusions

In this paper, we conducted a detailed cryptanalysis of the One-to-One parameter sets in the alternating moduli wPRFs proposed by Alamati et al. Our analysis reveals critical vulnerabilities in these constructions, allowing for efficient key recovery attacks that compromise the claimed λ-bit security levels. Specifically, we presented an attack with complexity $\mathcal{O}(2^{\lambda/2} \log_2 \lambda)$ against the Standard One-to-One wPRF and $\mathcal{O}(2^{0.84\lambda})$ against the Reversed Moduli variant. Both attacks exploit the reduction in output space caused by the 0-values in the random but fixed key, which induces sub-mappings that deviate significantly from the intended one-to-one mappings. The effectiveness of the attacks was further validated through experimental implementations.

To address these vulnerabilities, we propose potential countermeasures. One strategy is to restrict the selection of keys to elements in $\mathbb{F}_p^*$, thereby excluding zero values as

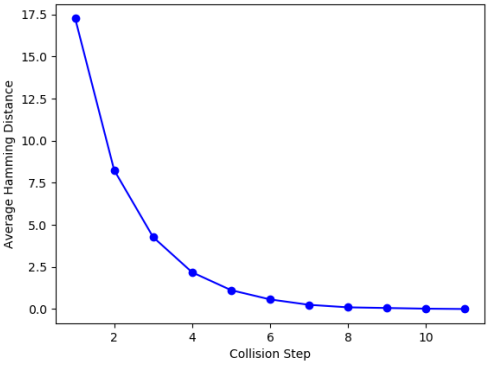


Figure 2: Number of found collisions vs. the average Hamming distance between the guessed key and the actual key for $\lambda = 34$.

coefficients in the key and ensuring that no part of the input is zeroed out in the first operation of the wPRF. The drawback of this mitigation is that p must be greater than 2 for this countermeasure to be applicable, and so one can not have $\mathbb{F}_2^n$ as the space for inputs and keys. Another approach is to replace the pointwise multiplication operation with addition or another operation that does not make any part of the input irrelevant.

We also identify open problems for future research. A deeper analysis of the Many-to-One parameter sets, which were not susceptible to our current attack, could shed light on the resilience of alternating moduli constructions in different configurations. Additionally, studying the trade-offs between mitigation techniques and their impact on performance in secure MPC environments requires further investigation. Finally, exploring alternative low-depth cryptographic designs that balance efficiency and security remains an important direction.

References

[AGP+19] Martin R. Albrecht, Lorenzo Grassi, Léo Perrin, Sebastian Ramacher, Christian Rechberger, Dragos Rotaru, Arnab Roy, and Markus Schofnegger. Feistel structures for MPC, and more. In Kazue Sako, Steve Schneider, and Peter Y. A. Ryan, editors, *ESORICS 2019, Part II*, volume 11736 of *LNCS*, pages 151–171. Springer, Cham, September 2019.

[APRR24] Navid Alamati, Guru-Vamsi Policharla, Srinivasan Raghuraman, and Peter Rindal. Improved alternating-moduli PRFs and post-quantum signatures. Cryptology ePrint Archive, Report 2024/582, 2024.

[ARS+16] Martin Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. Cryptology ePrint Archive, Report 2016/687, 2016.

[BIP+18] Dan Boneh, Yuval Ishai, Alain Passelègue, Amit Sahai, and David J. Wu. Exploring crypto dark matter: New simple PRF candidates and their applications. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 699–729. Springer, Cham, November 2018.

[Cha82] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO'82*, pages 199–203. Plenum Press, New York, USA, 1982.

[CLR90] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. McGraw-Hill, 1990.

[Das05] Anirban DasGupta. The matching, birthday and the strong birthday problem: a contemporary review. *Journal of Statistical Planning and Inference*, 130(1):377–389, 2005. Herman Chernoff: Eightieth Birthday Felicitation Volume.

[DEG+18] Christoph Dobraunig, Maria Eichlseder, Lorenzo Grassi, Virginie Lallemand, Gregor Leander, Eik List, Florian Mendel, and Christian Rechberger. Rasta: A cipher with low ANDdepth and few ANDs per bit. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 662–692. Springer, Cham, August 2018.

[DGH+21] Itai Dinur, Steven Goldfeder, Tzipora Halevi, Yuval Ishai, Mahimna Kelkar, Vivek Sharma, and Greg Zaverucha. MPC-friendly symmetric cryptography from alternating moduli: Candidates, protocols, and applications. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 517–547, Virtual Event, August 2021. Springer, Cham.

[FIPR05] Michael J. Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 303–324. Springer, Berlin, Heidelberg, February 2005.

[GØSW22] Lorenzo Grassi, Morten Øygarden, Markus Schofnegger, and Roman Walch. From farfalle to megafono via ciminion: The PRF hydra for MPC applications. Cryptology ePrint Archive, Report 2022/342, 2022.

[GRR+16] Lorenzo Grassi, Christian Rechberger, Dragos Rotaru, Peter Scholl, and Nigel P. Smart. MPC-friendly symmetric key primitives. Cryptology ePrint Archive, Report 2016/542, 2016.

[MRV99] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *40th FOCS*, pages 120–130. IEEE Computer Society Press, October 1999.

[NR97] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th FOCS*, pages 458–467. IEEE Computer Society Press, October 1997.

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 552–565. Springer, Berlin, Heidelberg, December 2001.

[Wag02] David Wagner. A generalized birthday problem. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 288–303. Springer, Berlin, Heidelberg, August 2002.

Graphic design: Communication Division, UiB / Print: Skipnes AS

uib.no

ISBN: 9788230895979 (print)
9788230879092 (PDF)