# A Survey on Transciphering and Symmetric Ciphers for Homomorphic Encryption

Indranil Thakur<sup>1</sup>, Angshuman Karmakar<sup>1</sup>, Chaoyun Li<sup>2,3</sup>, and Bart Preneel<sup>3</sup>

<sup>1</sup>Indian Institute of Technology Kanpur, India {indra, angshuman}@cse.iitk.ac.in <sup>2</sup>University of Surrey, England c.li@surrey.ac.uk <sup>3</sup>Katholieke Universiteit Leuven, Belgium bart.preneel@esat.kuleuven.be

#### Abstract

Data privacy concerns are sharply rising in the current digital era, hyperdriven by cloud computing, big data analytics, and the Internet of Things. Homomorphic Encryption (HE) has emerged as an ideal technique for computing on encrypted data, but current schemes suffer from slow encryption speed and large ciphertext expansion. Practical implementation is hindered, especially when the client has limited bandwidth, memory, and computing power. In 2011, Naehrig et al. proposed transciphering, reducing computational and communication overload on the client side. This involves symmetric ciphers with minimized multiplicative complexity, referred to as HE-Friendly Ciphers (HEFCs).

In this work, we present a detailed study of transciphering for HE by systematizing existing knowledge and crystallizing research challenges. Particularly we conduct a comprehensive study on state-of-the-art HEFC constructions. Our work highlights gaps, open problems, and directions for future research.

Keywords— Homomorphic Encryption, Symmetric Cryptography, Data Privacy, Cloud Computing, Transciphering, Hybrid Homomorphic Encryption, Block Cipher, Stream Cipher

**Version History:** Date of the current version: December 18, 2024. A preliminary version of the paper was submitted to PETS 2025(1) on June 1, 2024. After rejection, a slightly revised version was submitted to a journal on December 18, 2024.

# Contents

1	Introduction31.1HE in the real world41.2Our Contribution51.3Organization of Article5
2	An Overview of HE Schemes52.1Lattice Based Cryptography52.2FHE: From Gentry to CKKS62.3Performance Bottlenecks72.4Applications to Privacy-Preserving Outsourced Computation8
3	Transciphering93.1Protocols: Theory and Practice93.2HE-Friendly Cipher Design113.3Authenticated Transciphering113.4Verifiable Transciphering123.4.1Verifiable Computation123.4.2Implications on Transciphering12
4	An Overview of HE-friendly Ciphers       12         4.1       BGV/BFV- Friendly Ciphers       13         4.1.1       LowMC       14         4.1.2       Rasta (and variants)       14         4.1.3       Chaghri       15         4.2       TFHE- Friendly Ciphers       15         4.2.1       Trivium and Kreyvium       15         4.2.2       FLIP, FiLIP and Elisabeth       15         4.3       CKKS- Friendly Ciphers       16         4.3.1       Hera       16         4.3.2       Rubato       16
5	Benchmarks and Comparisons of HEFCs         17           5.1         BGV/BFV         17           5.2         TFHE         18           5.3         CKKS         18
6	Real-world Employment of HEFCs 18
Α	A Brief Overview of Symmetric Key Cryptography29A.1 Stream Ciphers29A.2 Block Ciphers30A.3 Authenticated Encryption30
В	Details of Ciphers       30         B.1       Trivium and Kreyvium
С	Attacks on HEFC       34         C.1 Statistical Attacks       34         C.2 Algebraic and Structural Attacks       34

# 1 Introduction

The phrase "Data is the new oil" [1] depicts how important data has become in today's digital world. Just like oil, when data is collected, processed, and used safely and efficiently, it can be supremely valuable in driving innovation, economic growth, and societal evolution. However, also like oil, if data is mishandled, stored, or processed incorrectly, it can potentially harm the interests of mankind by unauthorized access, identity theft, and privacy breaches [2, 3, 4, 5, 6, 7].

In the last few decades, the rapid progress and ongoing innovations in web and communication technologies have led to a manifold increase in data generation [8]. Particularly, machine learning and artificial intelligence based methods have been used to hyperdrive the innovations in the field of information processing [9, 10]. Often these technologies rely on massive computations performed on big data. However, *Internet of Things* (IoT) and resource-constrained devices lack sufficient resources for such tasks. As a result, outsourcing data for computations to the cloud has become a trend across several organizations, industries, governments, and others [11, 12]. However, cloud clients have to blindly trust third-party cloud service providers, posing data privacy concerns. To safeguard individuals' privacy and regulate ethical handling of data, several countries and regions have enacted data protection laws, such as the European Union's *General Data Protection Regulation* (GDPR) [13], United States' *California Consumer Privacy Act* (CCPA) [14], India's *Digital Personal Data Protection* (DPDP) [15] etc. However, these laws are confined by their respective jurisdictions and exert restricted influence beyond specific geographic areas. Consequently, relying solely on these laws is inadequate.

Cryptology has become the only solution to ensure data privacy with *provable security*. As of now, it has played a vital role in safeguarding data both during storage (data-at-rest) and during transmission (data-in-transit), by using Advanced Encryption Standard (AES) [16] and by using TLS 1.3 [17] respectively. However, as data is frequently processed in untrusted environments in today's digital world, there is a growing demand to ensure its protection while it is in process (data-in-use). A simple solution could be Trusted Execution Environments (TEEs), such as Intel's Software Guard Extensions [18], Technology Lifecycle Solutions [19] technology and ARM's TrustZone technology [20], where users can complete all sensitive data manipulation in an isolated secure system. Specifically, data is encrypted outside the microprocessor, decrypted upon entering it for computations, and then re-encrypted as it leaves the processor for storage. However, modern processors with efficiency-driven features like deep pipelines, out-of-order execution, speculative execution, etc., inherently create signals that external observers can exploit to discern internal processor activities. This challenges TEEs as successful attacks on technologies such as SGX have been demonstrated [21, 22, 23]. This is where, cryptographic techniques such as Homomorphic Encryption (HE), Functional Encryption, Multi-party Computation, often denoted by the umbrella term *Computing On Encrypted Data* (COED), and privacy techniques such as *Zero-Knowledge Proofs* have emerged with greater potential. Each of these techniques has distinct properties and can be employed in different scenarios to facilitate ethical data processing while preserving its privacy. In this paper, we focus solely on HE and related technologies.

HE [24] is a very powerful notion that enables one to compute on encrypted data without first having to decrypt it. Specifically, the results of these computations remain encrypted and, when decrypted, produce the same output as if the operations had been performed on the unencrypted data. HE is categorized by the type of functions that can be performed on the encrypted data (i.e. type of circuits that can be evaluated), as follows:

- Fully HE (FHE). These schemes allow the evaluation of arbitrary circuits of unbounded depth and is often referred to as the "holy grail" in cryptography. The notion of (F)HE was first introduced by Rivest *et al.* in 1978 [25], and first constructed by Gentry in 2009 [26, 27]. To this day, the fundamental concept behind creating an FHE remains unchanged from Gentry's approach, namely to construct a *simpler* HE (capable of evaluating only limited circuits) and then transform it into an FHE through a costly process named *bootstrapping*. Examples include Torus-FHE (TFHE) [28] and Cheon-Kim-Kim-Song (CKKS) [29]. While FHE is the strongest notion of HE, its high computational cost makes *simpler* HE schemes more suitable and preferable for certain applications.
- Leveled Fully HE (LHE). These schemes allow the evaluation of arbitrary circuits of bounded (pre-determined) depth and are very useful in scenarios where a known, limited depth of computation is sufficient for the task at hand such as in privacy-preserving machine learning. Examples include Brakerski-Gentry-Vaikuntanathan (BGV) [30] and Brakerski-Fan-Vercauteren (BFV) [31].
- Somewhat HE (SHE). These schemes allow the evaluation of circuits containing unlimited additions but limited multiplications. They are very useful in scenarios where simple statistical functions, such as mean and standard deviation, need to be computed, as well as in tasks like logistic regression, which are commonly used for predicting the likelihood of desirable or undesirable outcomes. Examples include Boneh-Goh-Nissim (BGN) [32].
- Partially HE (PHE). These schemes allow the evaluation of circuits consisting of only one type of

gate, such as exclusively additions or multiplications. They are useful in scenarios such as in secure voting protocols where the primary operation is to tally votes, which typically involves the addition of encrypted vote values. Examples include the Paillier [33] (addition) and the RSA [34] (multiplication).

### 1.1 HE in the real world

There is a growing trend to update all applications that handle personal or private data to secure them using HE. The real-world applications of HE are galore [35, 29, 36, 37, 38, 39, 40, 41, 42]. It is extremely beneficial in the context of privacy-preserving applications such as Cloud Computing [43] (see Fig. 1), Machine Learning [44], Recommender System [45], Biometric authentication [46], Healthcare [47] and Finance [48].



Figure 1: Privacy-Preserving Cloud Computing; H denotes the HE scheme, m denotes the client's data, f denotes the function supposed to compute on m, the red arrow shows what one needs, and the green arrows show how to do it.

HE [24], particularly FHE [49], is a very active research area. Since Gentry's proposal of first solution [26, 27] several FHE schemes have been designed [30, 31, 50, 51, 28, 29] so far. However, all of them suffer from two major technical problems (compared to traditional cryptography such as *Symmetric Cryptography*) as follows:

- Slow encryption speed. FHEs require more processing power, thus making them slower.
- Large ciphertext expansion. FHEs rely on noise to hide secrets, making the ciphertext-to-plaintext ratio huge.

Consequently, the associated model for privacy-preserving applications (i.e. Fig. 1) faces three major challenges as follows:

- Computation. The client faces computational overload while calculating the homomorphic encryptions (i.e. H.Enc(m)), due to the slow encryption speed.
- Communication. The client faces communication overload while transmitting the large homomorphic ciphertexts (i.e. H.Enc(m)) to the cloud, due to significant ciphertext expansion.
- Verification. Additionally, due to the inherent complexity of FHE operations, the high overhead of
  existing verification techniques, and the lack of optimized, lightweight solutions for verifying encrypted
  computations without compromising security or performance, the real-world applications are not yet
  practical.

So, even though FHE exists theoretically, its widespread public deployment is often debated. As a result, these limitations are often referred to as an engineering challenge. The primary goal of privacy-preserving technologies, such as cloud computing, is to relieve clients from the need to manage their own storage and computational resources. Therefore, addressing the computational and communication overloads on the client side is the top priority, which is the main focus of this article.

To reduce the computational and communication overload on the client side, *Transciphering*, also known as *Hybrid Homomorphic Encryption* (HHE) has been proposed [37, 52]. It's a hybrid framework employing a symmetric cipher along with an FHE, capitalizing symmetric cryptography's fast encryption speed and minimal ciphertext expansion. However, unlike the traditional symmetric ciphers such as AES [16], transciphering requires the symmetric cipher to be *friendly* towards HE, i.e. efficient under HE operations. In other words, transciphering necessitates that symmetric ciphers focus on AND-related metrics, such as multiplicative complexity, multiplicative complexity per encrypted bit, and multiplicative depth, rather than the traditional metrics like encryption/decryption speed, throughput, power consumption, memory footprint, etc. Since then, designing and analyzing these symmetric ciphers, called *HE-Friendly Ciphers* (HEFCs), has emerged as a necessary research area.

### 1.2 Our Contribution

The primary goal of this article is to provide a complete overview of transciphering for HE, with a focus on symmetric ciphers used in them. Since the central aim of privacy-preserving technologies such as cloud computing is to eliminate the need for clients to maintain storage and computational resources themselves, addressing the computational and communication overloads on the client side is more crucial than anything else. As an emerging research domain, a considerable amount of work still needs to be done. In recent years, there has been increased research on transciphering and HEFCs, with many new results coming because of the shortcomings of existing algorithms and new successful attacks on existing strategies. As a result, lots of information is scattered around in the literature while rapid development is going on, making it harder to keep track. So, a comprehensive review is necessary to make progress. Our motivation is to give a uniform view by organizing and collating all this information, thereby fostering future research and development.

To systematize the knowledge, we review transciphering frameworks with a focus on the existing HEFCs. The principal contributions of our work are as follows:

- We contextualize the knowledge of HE from Gentry's breakthrough result to approximate HE such as CKKS. Given the performance bottlenecks of HE schemes, we introduce the transciphering framework for HE. In a client-server transciphering model, we identify three marjor gaps: lack of a standardized HEFC, authentication of client's input data, and verifiability of output returned by the server.
- We systematically present an overview of the existing HEFCs, briefly describing their design rationale, security analysis, and presenting benchmark results. Specifically, we categorize the HEFCs based on their methodologies, providing a solid foundation for an equitable comparison.
- We crystallize research problems to furnish future research directions.

To the best of our knowledge, this SoK explores all the existing transciphering frameworks and HEFCs. The article intends to support enthusiastic readers in understanding the scope of HEFCs, learning the impact of possible attacks on the schemes, and indicating possible directions to design more efficient and secure HEFCs. Hence, our work can promote research on transciphering and the earnest requirement to instantiate HE practically.

### 1.3 Organization of Article

We begin with an introduction to HE and its application to cloud computing in Section 2. Then, we discuss the transciphering framework in Section 3, followed by an overview of HEFCs in Section 4. In the subsequent Sections 4.2-4.3 and 5, we illustrate the design rationale, security analysis, and efficiency of the HEFCs. Finally, we conclude in Section 6 with a discussion of real-world employment and open problems.

# 2 An Overview of HE Schemes

This section briefly discusses a few relevant topics to help readers understand the rest of the article.

### 2.1 Lattice Based Cryptography

Using hard lattice problems to construct cryptographic schemes was first demonstrated in the seminal work of Ajtai [53]. This work uses worst-case instances of Short Integer Solution (SIS) problems to construct one-way functions. Later, in 2005, Regev [54] introduced the *Learning With Errors* (LWE) problem, which is fundamentally solving approximate linear equations and also showed that the hard lattice problem of finding shortest vectors in a lattice can be reduced to it. Remarkably, Regev showed that the average-case LWE instances are as hard as the worst-case instances of the lattice, which constituted the foundation for lattice-based cryptography. Later Lyubashevsky *et al.* introduced a variation of this problem known as *Ring-Learning With Errors* (RLWE) problem, which allows more efficient cryptographic constructions compared to LWE problem.

**RLWE Problem** Let  $R_q = \mathbb{Z}_q[x]/\langle \Phi(x) \rangle$ , where q (¿1) is an integer and  $\Phi(x)$  is a cyclotomic polynomial. Consider a uniform random polynomial  $a \in R_q$  and two polynomials s (secret) and e (error) sampled from a narrow distribution  $\chi$  on  $R_q$ . An RLWE sample is denoted by a tuple  $(a, b) \in R_q^2$ , where  $b = a \cdot s + e$ . Finally, the RLWE problem has the following two variations:

- Search problem: having access to polynomially many RLWE samples, find s.

- Decision problem: distinguish between RLWE samples and uniformly random samples from  $R_a^2$ .

**RLWE Encryption** Based on the RLWE decision problem, we have the following encryption scheme as described in the extended version of [55]. The plaintext space is taken as  $R_t = \mathbb{Z}_t[x]/\langle \Phi(x) \rangle$  (integer t > 1 being the plaintext modulus) and the ciphertext space as  $R_q^2$  (integer q > 1 being the ciphertext modulus). The public key (pk) space is taken as  $R_q^2$  and the secret key (sk) space as  $R_q$ . We denote  $[a]_t$ (resp.  $[a]_q$ ) for some  $a \in R_t$  (resp.  $a \in R_q$ ) as the element in  $R_t$  (resp.  $R_q$ ) obtained by applying mod t(resp. mod q) to all the coefficients of a. The RLWE-based encryption scheme is then defined as follows: – Key Generation (sk, pk)

Sample  $a \leftarrow R_q$  and  $s, e \leftarrow \chi$ 

$$sk = (s)$$
  
 $pk = (b, a)$  where  $b = [-(a \cdot s + e)]_q$ 

- Encryption (RLWE.Enc :  $R_q^2 \times R_t \to R_q^2$ ) Sample  $u, e_1, e_2 \leftarrow \chi$ The encryption of a message  $m \in R_t$ , RLWE.Enc(pk, m) is

$$(c_0, c_1) = ([b \cdot u + e_1 + \lfloor q/t \rfloor \cdot m]_q, [a \cdot u + e_2]_q).$$

- Decryption (RLWE.Dec :  $R_q \times R_q^2 \to R_t$ ) The decryption of a ciphertext  $(c_0, c_1) \in R_q^2$ , RLWE.Dec(sk, c) is

$$m = \left[ \left\lfloor \frac{t \cdot [c_0 + c_1 \cdot s]_q}{q} \right\rfloor \right]_t.$$

### 2.2 FHE: From Gentry to CKKS

HE schemes have been developed using different approaches. Specifically, FHE schemes are often grouped into generations corresponding to their development and the underlying approach [56].

**Pre-FHE** The challenge of designing an FHE scheme was first proposed by Rivest, Adleman, and Dertouzos in 1978 [25]. For almost 30 years, there was no solution. However, some partial results were achieved with the invention of public-key cryptosystems, such as RSA [34] allowing only infinite modular multiplications and the Paillier [33] allowing only infinite modular additions.

**First-generation FHE by Gentry** In 2009, Gentry gave the first design for an FHE [26, 27] based on ideal lattices. His groundbreaking work contains two key ingredients: a *bootstrappable* (i.e. capable of evaluating its own decryption circuit and then at least one more operation) SHE scheme and a *bootstrapping* (see Fig. 2) transformation that converts a bootstrappable SHE scheme into an FHE scheme, through recursive self-embedding.



Figure 2: A graphical representation of bootstrapping.

Bootstrapping (see Fig. 2) is applying the idea of homomorphically evaluating the decryption operation using encryption of the secret key. One begins with a bootstrappable SHE. Suppose we have a ciphertext  $c = Enc_{pk}(m)$ , which is the encryption of message m under the public key pk, and  $Enc_{pk}(sk)$ , encryption of the secret key sk under the same public key. Then c is encrypted under pk to obtain  $Enc_{pk}(c)$ , and homomorphically evaluates the decryption circuit on the inputs  $Enc_{pk}(c)$  and  $Enc_{pk}(sk)$ . Now, since  $Enc_{pk}(c)$  is a fresh ciphertext, homomorphic operations can be performed on  $Enc_{pk}(c)$  as many as the SHE scheme can provide. Now taking f to be the decryption function  $Dec_{sk}$  and homomorphically evaluating the function f on the ciphertext  $Enc_{pk}(m)$ , we get  $Enc_{pk}(f(m))$ , which is  $Enc_{pk}(Dec_{sk}(c)) = Enc_{pk}(m)$ . But this is precisely what we started with! Thus if the SHE scheme has enough multiplicative depth to support the decryption function and one additional operation, then one additional homomorphic operation can be performed on this new ciphertext, and can repeat the procedure *ad infinitum*. In this way, a bootstrappable SHE can be converted into an FHE.

In [57] Gentry *et al.* show that the originally proposed scheme [26, 27] takes about 30 minutes for just a single-bit operation, which makes it impractical for real-world use. Dijk *et al.* [58] proposed another FHE scheme in 2010, based on Gentry's scheme, but instead of ideal lattices over polynomial rings, they used integers with modular arithmetic.

In brief, this generation marks the milestone of having a theoretically feasibles FHE scheme for the first time. The core concept involves the transformation of an SHE scheme into an FHE scheme. However, these schemes are very inefficient and are different from the techniques used today.

**Second-generation FHE** The homomorphic cryptosystems of this generation are derived from techniques developed by Brakerski, Gentry, and Vaikuntanathan (BGV), leading to the development of much more efficient FHEs based on (R)LWE assumptions. In 2011, Brakerski *et al.* proposed the BGV scheme [30], building on techniques of Brakerski-Vaikuntanathan [59]. In 2012, Fan *et al.* proposed the Brakerski-Fan-Vercauteren (BFV) scheme [31], building on Brakerski's scale-invariant cryptosystem [60]. All of the above FHE schemes feature a much slower noise growth during the homomorphic computations than previous schemes. The optimizations in [61, 62, 63], build on the techniques introduced by [64] that enable packing of many plaintext values in a single ciphertext and operating on all these plaintext values in a *Single Instruction Multiple Data* (SIMD) fashion. Many of the advances in this generation cryptosystems were also ported to the cryptosystem over the integers [65, 66].

This generation sees the development of much more efficient schemes than the first generation, which are still widely used and implemented. Here the core idea involves the conversion of an LHE scheme into an FHE scheme.

**Third-generation FHE** In 2013, Gentry *et al.* proposed an asymptotically faster scheme named Gentry-Sahai-Waters (GSW) [50], which uses a different approach than the second-generation schemes. They proposed a new technique for building FHE schemes that avoids an expensive *re-linearization* step in homomorphic multiplication. Brakerski and Vaikuntanathan observed that for certain types of circuits, the GSW cryptosystem features an even slower rate of noise growth, and hence better efficiency and more robust security [67]. Sheriff and Peikert then described a very efficient bootstrapping technique based on this observation [68]. These techniques were further improved to develop efficient ring variants of the GSW cryptosystem: FHEW [51] in 2014 and Torus-FHE (TFHE) [28] in 2016. The FHEW scheme was the first to show that by refreshing the ciphertexts after every single operation (gate bootstrapping), it is possible to reduce the bootstrapping time to a fraction of a second. Specifically, FHEW introduced a new method to compute boolean gates on encrypted data that significantly simplifies bootstrapping and implemented a variant of the bootstrapping procedure [68]. The efficiency of FHEW was further improved by the TFHE scheme, which implements a ring variant of the bootstrapping procedure [69] using a method similar to the one in FHEW. In a way, TFHE can be seen as porting FHEW, which uses both normal LWE and ring-GSW, to the Torus.

**CKKS** In 2016, a novel FHE scheme supporting approximate arithmetic was proposed by Cheon-Kim-Kim-Song (CKKS) [29]. The CKKS scheme includes an efficient rescaling operation that scales down an encrypted message after a multiplication. For comparison, such rescaling requires bootstrapping in the BGV and BFV schemes. The rescaling operation makes CKKS scheme the most efficient method for evaluating polynomial approximations, and is the preferred approach for implementing privacy-preserving machine learning applications.

The advent of CKKS marks the milestone of having an approximate FHE scheme for the first time, which can operate on floating point numbers.

### 2.3 Performance Bottlenecks

From a design perspective, a substantial amount of work has focused on developing asymptotically more efficient schemes. Similarly, lots of efficient software libraries and hardware architectures have been proposed to improve the efficiency of HE schemes. On the positive side, the performance of bootstrappable SHE schemes has increased by several orders of magnitude since the earliest implementations of 2009. However, even after decades of research, there is still considerable overhead in terms of both computational performance and parameter sizes in existing FHE designs that severely limit the practicality and applicability of current implementations. All our current FHE schemes suffer from two main technical problems: slow encryption speed and large ciphertext expansion.

**Slow encryption speed** The encryption/decryption time and the evaluation time of HE schemes are relatively slow compared to conventional encryption schemes such as symmetric cryptography. The most expensive operation in HE is bootstrapping. Although FHE schemes based on GSW, such as TFHE enjoy very fast bootstrapping, still most of the schemes, including CKKS suffer from a very slow bootstrapping operation. When the noise in the ciphertext grows too large due to the evaluation of some function, the bootstrapping operation homomorphically evaluates the decryption function, effectively refreshing the ciphertext while reducing the noise level, thereby ensuring the security and correctness of the computation. The decryption of FHE itself is an expensive operation due to the multiplication of polynomials, so the cost of homomorphically evaluating the decryption function is significantly higher than the cost of regular decryption. This is because bootstrapping typically involves evaluating a more complex function homomorphically, which is computationally intensive.

Large ciphertext expansion A major issue in HE is the huge ciphertext expansion (see Fig. 3), i.e. the ciphertext-plaintext size ratio is huge. For example, in the worst case, an FHE encryption of a single bit with TFHE [?] achieving 128 bits of security could result in a ciphertext size of 2.5KB. This is due to the way HE encryption is done, i.e. the RLWE encryption (see Section 2.1). The plaintexts (resp. ciphertexts) in RLWE encryption are polynomials (resp. polynomial pairs) from the quotient ring of polynomials  $R_t = \mathbb{Z}_t[x]/\langle \Phi(x) \rangle$  (resp.  $R_q = \mathbb{Z}_q[x]/\langle \Phi(x) \rangle$ ), where  $\Phi(x)$  is a cyclotomic polynomial. The most popular choice for expository purposes is to take  $\Phi(x) = x^n + 1$  where *n* is a power of 2. As a result, a plaintext (resp. ciphertext) is of size  $n\lceil \log_2(t)\rceil$  (resp.  $2n\lceil \log_2(q)\rceil$ ) bits, which thereby results in a ciphertext expansion factor of  $2\lceil \log_2(q)\rceil/\lceil\log_2(t)\rceil$ . For many HE applications, *t* is in the range of 16 to 60 bits, and the size of *q* can easily exceed 800 bits, resulting in big expansion factors. Thus, HE schemes require q >> t and, thereby, a large ciphertext expansion.



Figure 3: Large ciphertext expansion; blue (resp. red) colored boxes denote the coefficients of the plaintext (resp. ciphertext) polynomial(s).

Importantly, the plaintext is only decoded in the constant term of the ciphertext polynomial. This is necessary for the proper working of the HE scheme. Therefore, sending t bits of plaintext to the server requires sending 2nq bits of ciphertext to the server. Since  $q \gg t$ , this requires a massive amount of data transfer between the client and the server.

### 2.4 Applications to Privacy-Preserving Outsourced Computation

Outsourcing data for computations (e.g. cloud computing) to a remote server (e.g. cloud) has become a trend across several organizations, industries, governments, and others [11, 12]. However, in doing so, clients often have to blindly trust third parties (e.g. cloud service providers) by sharing their private data in plaintext. Furthermore, the client has limited visibility and control over how their data is stored, processed, or protected once it leaves their local environment. This increases the risk of unauthorized access, data breaches, or misuse, introducing significant data privacy concerns.

HE is the ideal solution for *Privacy-Preserving Outsourced Computation* (PPOC). In a typical usage scenario, there is a client with some confidential data  $\{m_i\}$ , who wishes to delegate the computation of a certain function f, i.e.  $f(\{m_i\})$ , to a remote server. A simple PPOC model (see Fig. 4) employing an HE scheme H would be as follows:

- 1. Client generates public/secret key pair  $(pk^H, sk^H)$  for H
- 2. Client encrypts its data under H and sends the ciphertexts (i.e. Z) to the server
- 3. Server computes the function f on the encrypted data (gets y) and sends the result back to the client 4. Client decrypts y to get the desired result (i.e.  $f(\{m_i\})$ )

However, deploying PPOC using HE is not as simple as it may look. As discussed earlier in Section 2.3, all the HE schemes we know of have slow encryption speeds and large ciphertext expansion. For example, in the worst case, an FHE encryption of a single bit with TFHE [?] achieving 128 bits of security



Figure 4: Privacy-Preserving Outsourced Computation using an HE scheme H.

could result in a ciphertext size of 2.5KB. As a result, despite the simplicity of the above-mentioned model (i.e. Fig. 4), the client faces huge overloads in:

- Computation. The client faces computational overload while calculating the homomorphic encryptions (i.e. Z), due to the slow encryption speed.
- Communication. The client faces communication overload while transmitting the large homomorphic ciphertexts (i.e. Z) to the server, due to significant ciphertext expansion.

These challenges make the model practically unachievable, especially when the client is an embedded device with limited bandwidth, memory, and computing power. The most natural approach then, is to free the client from the burden of complex homomorphic computations. This can be accomplished through the *transciphering* [37], which we will explore further in the sequel.

# 3 Transciphering

Practical implementation of PPOC is hindered especially due to the demerits of HE and the client being a device with limited bandwidth, memory, and computing power (see Section 2.4). To address this, *transciphering* [37], also known as HHE, has been proposed. As the name suggests, it's a hybrid model employing a symmetric cipher along with an FHE scheme. It starts with the client encrypting its data using the symmetric cipher and transmitting those ciphertexts to the server. Upon receiving, the server converts those symmetric ciphertexts into homomorphic ciphertexts, thus enabling themselves to perform meaningful computations on encrypted data. Then, the server does the required computations on the encrypted data and sends the encrypted result back to the client, who can decipher it using the appropriate key. Importantly, this addresses the challenges faced by the client in Figure 4 (see Section 2.4) as follows:

- Computation. Earlier, the client had to encrypt all its data with an HE scheme, which is costly in terms of computational resources such as time and memory (due to the slow encryption speed of HE schemes). However, in transciphering, the client employs a symmetric cipher to encrypt all of its data (except for the symmetric key). Symmetric cryptography facilitates faster encryption speeds compared to HE schemes, resulting in a significant reduction in computational resource usage.
- Communication. Also earlier, the client had to send HE ciphertexts to the server, which is costly in terms of communicational resources such as bandwidth and latency (due to the large ciphertext expansion of HE schemes). Whereas, in transciphering, symmetric ciphertexts are sent to the server. Symmetric cryptography facilitates minimal ciphertext expansion compared to HE schemes, resulting in a significant reduction in communicational resource usage.

However, the problem is not yet solved fully, as the resulting ciphertext that the server computes is still a large HE ciphertext. The solution to this is the dimension reduction technique introduced by Brakerski *et al.* [59]. In particular, the dimension reduction technique converts a ciphertext in  $\mathbb{Z}_q[x]/(x^n+1)$  (where both n and q are large in order to support expressive homomorphisms) to a ciphertext in  $\mathbb{Z}_p[x]/(x^k+1)$ , where both k and p are small. The resulting ciphertext encrypts the same message, although it does not support any further homomorphism. The server then applies this transformation and sends the resulting short ciphertext to the client.

These benefits come with the downside of a higher computational load on the server side, which is typically acceptable in practice, as servers usually have significantly more computing power than clients. With the overheads shifted to the server, the primary goal now is to minimize the server-side overload to a level that is practical for implementation. Till now, two transciphering frameworks are known: one for exact computation [37] and one for approximate computation [52].

### 3.1 Protocols: Theory and Practice

The first scheme (see Fig. 5) is from Naehrig *et al.* [37], a transciphering framework for exact computation. It employs a symmetric cipher S along with an HE scheme H. Suppose, there is a client with some

confidential data  $\{m_i\}$ , who wishes to delegate the computation of a certain function f, i.e.  $f(\{m_i\})$ , to a remote server. The framework works as follows:

- 1. Client generates secret key  $sk^S$  for S and public/secret key pair  $(pk^H, sk^H)$  for H
- 2. Client encrypts  $sk^S$  under H (gets x) and its data under S to get Y.
- 3. Client sends  $pk^H$ , x and Y to the server
- 4. Server encrypts the symmetric ciphertexts (i.e. Y) under H to get X.
- 5. Server converts the symmetric ciphertexts (i.e. Y) to homomorphic ciphertexts (i.e. Z) by homomorphically evaluating decryption circuit of S.
- 6. Server computes the function f on the resulting ciphertexts and sends back the result (i.e. y), encrypted under the H to the client



Figure 5: Transciphering framework for exact computation. H and S refer to homomorphic and symmetric encryption respectively

Although this framework (i.e. Fig. 5) marked a significant step in making PPOC more practical, it has the drawback of not supporting the CKKS scheme. In other words, it cannot be used to encrypt real (or complex) numbers, which is highly beneficial for privacy-preserving machine learning applications. The main reason is that designing an HEFC S operating on real (or complex) numbers is infeasible. If such an HEFC S over the real field were to exist, it would be represented as a real polynomial map. Consequently, any ciphertext would be expressed as a polynomial in the corresponding plaintext and the secret key over  $\mathbb{R}$ . Then, for given plaintext-ciphertext pairs  $(m_i, c_i)$ , an adversary can construct a system of polynomial equations with the unknown key k. The sum of  $||S_k(m_i) - c_i||_2^2$  over the plaintextciphertext pairs forms a real polynomial, where the actual key is the zero of this function. Since this polynomial is differentiable, its zeros can be efficiently found using iterative methods such as the gradient descent algorithm. By using multiple plaintext-ciphertext pairs, the probability of identifying a false key becomes negligible. To overcome this problem, another framework is proposed, which we will discuss below.

The second scheme (see Fig. 6) is from Cho *et al.* [52]. It's a transciphering framework for approximate computation, dubbed RtF (Real-to-Finite-field) framework. It employs a symmetric cipher S and two HE schemes: BFV and CKKS. Suppose, there is a client with some confidential data  $\{m_i\}$ , who wishes to delegate the computation of a certain function f, i.e.  $f(\{m_i\})$ , to a remote server. This framework works as follows:

- 1. Client generates secret key  $sk^S$  for S and public/secret key pairs:  $(pk^{\mathsf{BFV}}, sk^{\mathsf{BFV}})$  for  $\mathsf{BFV}$  and  $(pk^{\mathsf{CKKS}}, sk^{\mathsf{CKKS}})$  for  $\mathsf{CKKS}$ .
- 2. Client encrypts the symmetric key  $sk^S$  under BFV to get x.
- 3. Client encrypts its data under S to get Y.
- 4. Server runs S homomorphically to produce an BFV-encryption of the key stream, whilst the encryption of the message is transformed into an BFV ciphertext. The BFV-encryption of the key stream is then subtracted from the BFV-encryption of the symmetrically encrypted message, producing an BFV-encryption of just the message to get X.
- 5. Server performs an operation termed *half bootstrapping* to transform the BFV ciphertext into a CKKS ciphertext to get Z.

The main idea behind this framework (i.e. Fig. 6) is to combine the BFV and CKKS HE schemes and use a stream cipher S exploiting modular arithmetic in between. It inherits a wide range of usability from the previous transciphering framework (i.e. Fig. 5), such as efficient short message encryption or flexible repacking of data on the server side. Additionally, real numbers can also be encrypted without significant ciphertext expansion or computational overload on the client side. Furthermore, it eliminates



Figure 6: RtF framework for approximate computation; S refer to symmetric encryption, operation termed "Half bootstrapping" transforms BFV ciphertexts into CKKS ciphertexts.

the need for using the complex domain for message spaces (as required in the CKKS scheme) or for any expertise in CKKS parameter settings on the client side.

**Research Problem 1.** Apply the state-of-the-art FHE with transciphering framework to key data analytical tasks such as aggregate statistics, clustering and classification.

Transciphering is a relatively new field and is currently still under development. To employ transciphering HE in practice, we need to address the following problems:

- unsatisfactory performance of existing symmetric ciphers under FHE

- the server cannot verify input data integrity if symmetric encryption is used without authentication

- the client cannot verify function evaluation in the naive protocol

In the remainder of this section, we will summarize the research on the above problems by crystallizing major research areas and presenting a taxonomy of those areas.

### 3.2 HE-Friendly Cipher Design

The major overhead of transciphering is that one needs to homomorphically evaluation the symmetric ciphers, which can be very expensive as symmetric ciphers are computationally complicated functions. The cost of a homomorphic evaluation of several symmetric primitives has been investigated, including several optimized implementations of AES [61, 65, 70], and the lightweight block ciphers Simon [71] and Prince [72]. Since these ciphers have not been designed for the transciphering framework (i.e. different metrics), the performance of the homomorphic evaluation has been shown to be unsatisfactory. This situation leads to a natural question: which symmetric ciphers would be most appropriate for transciphering?

It was observed that this requires the symmetric cipher to be *friendly* towards the corresponding HE (i.e. efficient under HE operations) while still cryptographically strong enough that the server can not learn anything from the symmetrically encrypted data. Hence, the term "*HE-Friendly Cipher* (HEFC)". However, unlike traditional symmetric ciphers, the design of these HEFCs is driven by arithmetic complexity, improving the efficiency of the protocol employing them. A thoroughly treatment of state-of-the-art HEFCs will be given through Sections 4-6.

#### 3.3 Authenticated Transciphering

While reducing computational and communication overload on the client side, transciphering mainly protects data confidentiality, through symmetric encryption (for the encryption step and the transmission), and through homomorphic encryption.

However, it does not ensure data integrity over homomorphically encrypted data during transmission, although data corruption during transmission can lead to incorrect computation and, thereby incorrect decryption and results. To address this, *Authenticated Encryption* (AE) within transciphering referred to as *Authenticated Transciphering* (AT) [73] can be used. However, AE schemes normally have more operations than an encryption scheme since they provide both confidentiality and authentication. Actually, many AE schemes are built by combining an encryption scheme and an authentication algorithm. Thus, adding authentication to transciphering means even larger overhead to the HE application.

As a result, only three results are known till now, Grain128-AEAD over TFHE [73], a hash-based MAC (Message Authentication Code) over TFHE [74], and AES-GCM over CKKS [75].

**Research Problem 2.** Design AEAD for HEFCs. In particular, implementing transciphering using AE over CKKS is challenging due to its approximated nature.

#### 3.4 Verifiable Transciphering

It is common to assume that the server running FHE applications is honest-but-curious, i.e. the server is believed to perform the homomorphic computations correctly but not trusted with access to the confidential plaintext on which the computation is performed. However, this is not always the case in real-life scenarios, as the server can intentionally or accidentally violate the client's trust.

**Example 1.** Suppose an AI-based tool has two paid versions, AI-1 (basic) and AI-2 (advanced and more expensive), and a user wants to use the advanced version. However, the tool uses AI-1 intentionally/accidentally. Without a verification option, the user has no way of confirming the result, forcing them to blindly trust the tool while paying the higher price.

Data corruption during computation or incorrect computations by the server can lead to erroneous decryption. Therefore, clients should have the option of verifying the correctness of computation. In general, verifying the proper working of third-party servers is necessary before deploying transciphering in PPOC.

#### 3.4.1 Verifiable Computation

The trusted server assumption can be removed by adding verifiablitity to existing FHE schemes and thereby constructing *Verifiable FHE* (vFHE) [76].

A common approach for constructing vFHE is to combine an FHE scheme with a Verifiable Computation (VC) scheme which is used to prove the correctness of the homomorphic computations. However, combining these two primitives in an efficient way turns out to be a highly non-trivial task. Namely, VC can usually prove arithmetic circuits whose gates are additions or multiplications over some field  $\mathbb{F}_p$ , while the homomorphic computation that the server wants to prove is performed over polynomial rings.

To overcome this, recent works [77, 78] have studied how to modify the VC protocols to work over rings. While these are more concretely efficient than the constructions mentioned earlier, there is a significant gap between the assumptions made by existing work and the way state-of-the-art FHE schemes are used in practice. To address this, a combination of *Zero Knowledge Proof* (ZKP) and FHE [79, 80, 76] have been proposed. Although promising, these are still far from being practical. It is challenging to bring together modern FHE and ZKP systems, including the mismatch between the large polynomial rings used in most state-of-the-art FHE schemes and the integer fields used in the vast majority of ZKP systems.

### 3.4.2 Implications on Transciphering

While it seems straightforward to adapt the general vFHE schemes to the transciphering framework, the cost quickly becomes prohibitive as the multiplicative depth of the circuit grows. Hence, the following research question is relevant.

**Research Problem 3.** Verifiable transciphering is desirable for practical outsource computation. Instead of combining transciphering with expensive general verifiable computation schemes, is there a way to leverage the symmetric encryption to reduce the overhead of verification?

# 4 An Overview of HE-friendly Ciphers

Traditionally, the primary goal in symmetric cipher design is to reduce the area and latency of hardware/software implementations. However, when a symmetric cipher is combined with an HE scheme in a transciphering framework, the situation is radically different: linear operations come almost for free since they only incur local computation (resp. do not increase the noise much), whereas the bottlenecks are non-linear operations that involve symmetric cryptographic operations and communication between parties (resp. increases the noise considerably). Here, the main difference is that we have a different cost function in an optimization problem. This cost metric suggests a new way of designing a symmetric cipher where the use of non-linear operations is minimized.

All known FHE schemes are *noise-based*, i.e. all homomorphically encrypted ciphertext contains some noise, and each operation on the ciphertexts incurs an increase in the noise. Moreover, in most of the schemes, the noise level grows fast with the multiplicative depth of the circuit to be evaluated [81, 66]. Hence, a symmetric cryptosystem aiming for these types of applications minimizes, foremost the multiplicative depth of the circuit to be evaluated. While the cost of the application-specific homomorphic operations only depends on the multiplicative depth of the cipher, the cost of evaluating the additional decryption circuit itself primarily depends on the number of multiplications. Thus, the number of multiplications in the decryption circuit is also a relevant metric. With this observation, the efficiency of a HEFC is evaluated by three different metrics:

- (i) *Multiplicative complexity* (#ANDs), which is the total number of multiplications (in our case AND gates) per decryption circuit (see [82]),
- (ii) Multiplicative complexity per encrypted bit (#ANDs/bit), which is the total number of ANDs per bit of encrypted text [83] and
- (iii) *Multiplicative depth* (ANDdepth), which is the multiplicative depth of the decryption circuit (see [72]).

Optimizing the above metrics requires new design paradigms for symmetric cryptography. Since then several HEFCs has been proposed, with Table 1 providing a comprehensive overview of the literature from multiple perspectives. As we can see from Table 1, stream ciphers (modes) dominate the *zoo* of HEFCs. Compared to block ciphers, stream ciphers can exhibit a much lower multiplicative depth by making a significant part of the computations independent of the key [83]. The dominance of FLIP-like and Rasta-like ciphers clearly show this trend of design.

The security of a symmetric cipher highly depends on the choice of nonlinear components. It is worth pointing out some nonlinear functions have been emerging in the design of HEFCs. For ciphers with S-boxes in their round function, S-box is assumed to have low multiplicative complexity as well as provide a sufficiently high degree to prevent algebraic attacks. The most popular choices are sparse quadratic S-boxes, which are well understood in lightweight symmetric cryptography. A notable choice is the  $\chi$ -function in Keccak, which is used in Rasta and its variants. When it comes to  $\mathbb{Z}_p$ , the dominant choice is the cubic function  $x \mapsto x^3$ , which exhibits excellent cryptographic properties and minimal multiplicative complexity.

Ciphors	Implemented	Field	Security Construction		Round	References
Cipiters	HE Scheme <sup>†</sup>	Operation	Levels	Construction	Functions	TUTUTEILUES
Trivium	-	$\mathbb{Z}_2$	80	$\mathbf{SC}$	FSR	[84]
Kreyvium	BGV, BFV	$\mathbb{Z}_2$	128	$\mathbf{SC}$	$\mathbf{FSR}$	[85]
FLIP	GSW, BGV	$\mathbb{Z}_2$	128	$\mathbf{SC}$	FP	[86]
FiLIP	T <b>GSW</b> , BGV, TFHE	$\mathbb{Z}_2$	128	$\mathbf{SC}$	$\operatorname{FP}$	[87]
Elisabeth	TFHE	$\mathbb{Z}_{2^k}$	128	$\mathbf{SC}$	$\operatorname{FP}$	[88]
Rasta	BGV, BFV, TFHE	$\mathbb{Z}_2$	80/128/256		ASASA	[83]
Pasta	BGV, BFV, TFHE	$\mathbb{Z}_p$	128			[89]
Dasta	BGV	$\mathbb{Z}_2$	128	PBSC		[90]
Masta	BGV	$\mathbb{Z}_p$	128			[91]
Fasta	BGV	$\mathbb{Z}_2$	128			[92]
Hera	CKKS, BGV	$\mathbb{Z}_p, \mathbb{R}$	128	PBSC	SPN	[52]
Rubato	CKKS	$\mathbb{Z}_{p}^{'},\mathbb{R}$	128	PBSC	SPN+Feistel	[93]
LowMC v3	BGV	$\mathbb{Z}_2$	128	BC	Partial SPN	[94]
Chaghri	BGV	$\mathbb{F}_{2^{63}}$	128	BC	$\operatorname{SPN}$	[95]

Table 1: An overview of HEFCs. SC=Stream Cipher; PBSC = Permutation Based Stream Cipher, BC = Block Cipher, FP=Filter Permutator, SPN=Substitution-Permutation Network

<sup>†</sup> The schemes in **bold** are target HE schemes while the other schemes are used in various benchmarks

The remainder of this section mainly discuss design rationales and security analysis of HEFCs whereas detailed specifications of all ciphers are presented in Appendix B. A brief introduction to symmetric key cryptography and cryptanalytic attacks are shown in Appendices A and C respectively for the sake of completeness.

### 4.1 **BGV/BFV-** Friendly Ciphers

This section introduces ciphers tailored to FHE schemes such as BGV and BFV.

### 4.1.1 LowMC

LowMC [94] is a family of block ciphers proposed by Albrecht *et al.* It aims to achieve low #ANDs and is the first dedicated HEFC. Additionally, it has been used as the underlying block cipher of PICNIC [96], one of the third-round digital signature candidates for NIST PQC standardization procedure [97].

**Rationale** LowMC is a block cipher based on the *Substitution-Permutation Network* (SPN) structure. Notably, it adopts the so-called *partial* SPN construction, i.e., applying S-boxes over only partial state bits of the cipher. The design goal is to minimize #ANDs. A 3-bit S-box with low multiplicative complexity has been used. Moreover, the number of S-boxes applied in parallel is minimized to lower the multiplicative complexity, leaving part of the substitution layer as the identity mapping. To reach security despite low multiplicative complexity, pseudorandomly generated binary matrices are used in the linear layer to introduce a very high degree of diffusion. For design details, refer to Appendix B.3 with Fig. 9 in Appendix B.

**Security Analysis** The security analysis aims to determine the minimal number of rounds required to grant security for a given fixed set of parameters. Hence, the authors provided experimental and theoretical cryptanalysis to determine the minimal number of rounds. So far, LowMC is the most studied HEFC, owing to its innovative design and widespread applicability. It has multiple versions, with changes limited only to the parameter set, which has been updated to keep the security intact in the face of continuously improving cryptanalysis.

The first version, LowMCv0, was circulated at the end of 2014. In early 2015, observations by Khovratovich led to a new version, LowMCv1 [94]. Furthermore, due to the optimized interpolation attack by Dinur *et al.* [98], and new higher-order differential cryptanalysis by Dobraunig *et al.* [99], a revised edition, LowMCv2 [100] was introduced. Eventually, due to the difference enumeration attack by Rechberger *et al.* [101], a subsequent version, LowMCv3 [102] was introduced.

The significance of cryptanalyzing LowMC was elevated by its use in PICNIC [96], which made the single plaintext/ciphertext pair setting an important attack scenario. This is because a successful key recovery attack on LowMC using only a single plaintext and ciphertext is equivalent to retrieving the signing key of PICNIC. As a result, several attacks have been presented till now [103, 104, 105, 106, 107, 108, 109, 110], making some parameters in LowMCv3 still insecure.

#### 4.1.2 Rasta (and variants)

Rasta [83] is a family of stream ciphers proposed by Dobraunig *et al.* It was the first attempt to minimize both the metrics #ANDs/bit and ANDdepth simultaneously. At the same time, the authors also proposed Agrasta, an aggressive version of Rasta with the block size only slightly larger than the security level in bits. Furthermore, several other variants have been proposed over time: Dasta [90] by Hebborn and Leander, Masta [91] by Ha *et al.*, Pasta [89] by Dobraunig *et al.*, and Fasta [92] by Cid *et al.* 

**Rationale** Rasta produces keystreams by applying the key to a cryptographic permutation which exhibits an ASASA (*affine-substitution-affine-substitution-affine*) structure [111]. The core idea of Rasta is to make large parts of the operations, i.e. the permutation nonce-dependent but key-independent. The advantage of key-independent variations is that, on the one hand, the (key-dependent) ANDdepth can be kept very low while at the same time many standard attacks are not applicable to Rasta due to the nonce-dependent variations. Rasta minimizes both ANDdepth and #ANDs/bit by randomly updating the affine layers per round. As a result, the security arguments rely on the provided randomness and the encryption/decryption are potentially slowed down by this randomness generation.

The first variant, Agrasta was proposed to explore the limits of the design space by choosing aggressive parameters and to encourage more cryptanalysis. Since generating the affine layers in each encryption is quite time-consuming in Rasta, Dasta was proposed where the linear layer is replaced with an ever-changing bit permutation and a deterministic linear mapping. Such a construction made Dasta hundreds of times faster than Rasta in the offline settings. The next, Masta was proposed with two main differences from Rasta: using modular arithmetic to support HE schemes over a non-binary plaintext space, and a smaller number of random bits in the affine layers by defining them with finite field multiplication. Whereas, Pasta leverages the structure of BGV and BFV to minimize the homomorphic evaluation latency. Finally, Fasta was proposed with parameters and linear layer especially chosen to allow efficient implementation over the BGV scheme, particularly as implemented in the HElib library. For design details, refer to Appendix B.4 with Fig. 10 in Appendix B. **Security Analysis** The designers have explored various attack vectors before choosing parameters for the instantiations to rule them out conservatively. However, an algebraic attack using low-degree equations by Liu *et al.* in 2021 [112] broke a few instances of Agrasta. In the case of Dasta, the only attack is by Liu *et al.* [112], which applied to Rasta as well. However, the other variants have not been studied that much, and as a result no significant cryptanalysis result exists so far.

### 4.1.3 Chaghri

Chaghri [95] is a block cipher based on SPN structure that has a vector state S consisting of three elements from  $\mathbb{F}_{2^{63}}$ .

**Rationale** Chaghri is designed following the Marvellous design strategy [113] with a specific focus on BGV efficiency metrics. Power mappings over  $\mathbb{F}_{2^{63}}$  are used as the S-boxes while  $3 \times 3$  MDS matrix are exploited as the linear layer. For design details, refer to Appendix B.5 with Fig. 11 in Appendix B.

**Security Analysis** Liu *et al.* showed that a higher-order differential attack on eight rounds could be achieved with time and data complexity of  $2^{38}$ , using a new technique called coefficient grouping [114]. Hence, it indicates that the full eight rounds are far from being secure. Moreover, they have also proposed a modification in the design to avert the attack. In [95], the authors have implemented the modified **Chaghri** using HElib, achieving a throughput of 0.28 seconds-per-bit, which is 63% faster than **AES** in the same setting.

# 4.2 **TFHE-** Friendly Ciphers

This section presents ciphers dedicated to third-generation FHE schemes such as GSW, FHEW and TFHE.

### **4.2.1** Trivium and Kreyvium

Trivium [84] proposed by De Cannière and Preneel, is a stream cipher that was one of the eSTREAM project finalists [115] and is part of the ISO/IEC 29192-3 standard for lightweight stream ciphers. Whereas, Kreyvium [85] is a variant of Trivium, specially designed HEFC by Canteaut *et al.* 

**Rationale** Trivium, based on Feedback Shift Registers (FSRs), is designed to minimize #ANDs aiming for resource-constraint environments such as IoT. It also provides a flexible trade-off between speed and gate count in hardware and reasonably efficient software implementation. Whereas, Kreyvium shares the same internal structure as Trivium but allows for bigger keys of 128 bits, thus providing 128-bit security (instead of 80-bit) with the same ANDdepth, inheriting the same security arguments. For design details, refer to Appendix B.1 with Fig. 7 in Appendix B.

**Security Analysis** Trivium being an eSTREAM finalist, has been studied deeply [116, 117, 118, 119]. Also, there are multiple articles [120, 121, 122, 123, 124, 125] which have shown weakness in reduced rounds of Kreyvium. The best attack till now is by He *et al.* [126] which presents cube attacks on 851 rounds of Trivium and 899 rounds of Kreyvium.

### 4.2.2 FLIP, FiLIP and Elisabeth

FLIP (*family of filter permutators*) [86] and its variants[87, 88] are a family of stream ciphers that are based on a variant of the filter generator construction, but drops the state update function to avoid the algebraic degree increase.

**Rationale** The security of FLIP predominantly derives from the cryptographic properties of the filter function. FLIP employs Direct Sums of Monomials (DSM) as its filter, which takes into account the most common attacks on filter generators.

FiLIP (*family of improved filter permutators*) [87] is a variant of FLIP based on similar design strategy. Compared to FLIP, FiLIP instantiates two function families: DSM and XOR-Thresholds in its filter function.

Elisabeth [88] further extends the designs of FLIP and FiLIP by operating in an additive group such as  $(\mathbb{Z}_{2^k},+)$  rather than extensions of  $\mathbb{F}_2$  in most traditional symmetric ciphers. Thus, Elisabeth enables homomorphic computations in  $\mathbb{Z}_{2^k}$  without expensive conversions. As a result, Elisabeth is well-suited for transciphering with TFHE. For design details, refer to Appendix B.2 with Fig. 8 in Appendix B. **Security Analysis** The initial design of the cipher was proposed by Méaux in [127]. Immediately after that, Duval *et al.* [128] proposed cryptanalysis, revealing weaknesses in its filter function that can be exploited to devise an efficient full key recovery attack based on guess-and-determine techniques. Later, the design was tweaked to prevent the attack, and an updated design [86] was proposed. In [129], Gilbert *et al.* presented several variants of a key-recovery attack on the full Elisabeth-4 that break the 128-bit security claim of that instance of Elisabeth. The most optimized attack out of all those is a chosen-IV attack with a time complexity of  $2^{88}$  elementary operations, a memory complexity of  $2^{54}$  bits and a data complexity of  $2^{41}$  bits. To mitigate the attacks, a few new instances and variants of Elisabeth were proposed in [130].

## 4.3 CKKS- Friendly Ciphers

CKKS [29] is specifically designed to support computation on encrypted real numbers, making it wellsuited for real-world applications involving real-valued data such as privacy-preserving machine learning [131]. We now discuss the symmetric ciphers Hera and Rubato, both of which are specifically designed to be CKKS-friendly.

### 4.3.1 Hera

Hera [52] proposed by Cho *et al.* is a stream cipher specially designed to be CKKS-friendly in the RtF transciphering framework [52].

**Rationale** In the RtF transciphering framework, a stream cipher using modular arithmetic is required as a building block. However, there were only a few ciphers using modular arithmetic [132, 113, 133, 134], and even such algorithms are not suitable for the transciphering framework due to their high multiplicative depths. Recent constructions for HEFCs such as FLIP and Rasta use randomized linear layers in order to reduce the ANDdepth without security degradation. However, this type of ciphers spends too many random bits to generate random matrices, slowing down the overall speed on both the client and the server sides. Instead of generating random matrices, Hera aims to randomize the key schedule algorithm by combining the secret key with a (public) random value for every round. Using a simple randomized key schedule is the main feature of Hera. As a result, Hera requires a smaller number of random bits compared to FLIP and Rasta. For design details, refer to Appendix B.6 with Fig. 12 in Appendix B.

**Security Analysis** In [135], Liu *et al.* presented new algebraic attacks with multiple collisions in the round keys. Specifically, according to the special way to randomize the round keys in Hera, the authors find it possible to peel off the last nonlinear layer by using collisions in the last-round key and a simple property of the power map. In this way, they constructed an overdefined system of equations of a much lower degree in the key, and efficiently solved the system via the linearization technique. However, the primary instantiation of Hera, i.e. the 128-bit security version is not affected by this attack due to the high cost of finding multiple collisions.

### 4.3.2 Rubato

Rubato [93] proposed by Ha *et al.* is a stream cipher specially designed to be CKKS-friendly in the RtF transciphering framework [52].

**Rationale** Rubato follows a novel design strategy of adding noise to increase the algebraic degree of a cipher. With this strategy, the multiplicative complexity of the cipher is significantly reduced, compared to existing HEFCs. Rubato employs building blocks from Hera and Pasta. For linear layers and the key schedule, the style of Hera has been followed. A nonlinear layer whose inverse is of a high degree mitigates algebraic Meet-in-the-Middle attacks. However, due to the unavailability of a quadratic function with the inverse of a high degree over  $\mathbb{Z}_q$ , a cubic S-box has been used in HERA, which leads to a large multiplicative depth. For design details, refer to Appendix B.7 with Fig. 13 in Appendix B.

**Security Analysis** In [136] Grassi *et al.* showed 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.

# 5 Benchmarks and Comparisons of HEFCs

This section conducts a comparative study of HEFCs by presenting benchmarks with certain HE schemes and libraries. It is worth noting that the benchmarks are based on many factors, ranging from the specific HE scheme, to HE library, and to the used data type. Before presenting the performance data, we briefly summarize some widely used HE libraries for benchmarks of HEFCs.

**HE libraries** The last decade has seen the development of a variety of open-source software libraries implementing HE schemes, which have been extensively used by researchers and developers working on HE. The libraries and schemes mentioned in this paper include: HElib [137, 138] with its implementations of BGV and CKKS; Microsoft's SEAL [139] supporting BFV, BGV and CKKS, TFHE [28] implementing TFHE and its two variants Concrete [140] and TFHE-rs [141]. For a more thorough overview of HE libraries, please refer to [142].

We incorporate the performance and efficiency data from a variety of papers based on the following criteria:

- More recent benchmarks are preferable as they represent the state of the art. Usually, the most recent variant of a design has the best performance compared to its predecessors due to continuous improvements. So, we focus on the most efficient variants.
- Comparisons are justified if the benchmarks are based on the same library. So, we compared them
  in the same library.
- We divide the benchmarks into three categories based on the HE schemes involved since most HEFCs only optimize with respect to one type of HE schemes.

Table 2: Comparison of ciphers for BGV/BFV, using SEAL; security level = 128 bit [89, Tables 3 and 9]

Field	Ciphor	1 Block				
rieiu	Orpher	Enc. symm. key	Decompress			
		(s)	(s)			
	LowMC	1.75	613.9			
$\mathbb{Z}_2$	Rasta-6	1.42	88.5			
	Kreyvium	1.84	412.8			
	FiLIP-1280	16.7	1251.6			
	Pasta-3	0.017	9.28			
$\mathbb{Z}_p$	Masta-4	0.058	54.2			
	Hera	0.051	16.6			

### 5.1 BGV/BFV

As discussed in Section 4.1, a lot of HEFCs have been proposed for BGV/BFV. LowMC was implemented for BGV and BFV with HElib in [102]. The benchmark shows an improvement by a factor of 5 compared to AES-128 in computation and communication complexity. In [85], the authors compared the latency and throughput of Trivium and Kreyvium with that of LowMC for BGV and BFV in HElib. The results show that Trivium and Kreyvium have a smaller latency than LowMC, but have a slightly smaller throughput.

BGV and BFV allow for integer plaintexts in  $\mathbb{Z}_q$  with  $q \geq 2$ . Many HEFCs support inputs in  $\mathbb{Z}_p$  to improve the efficiency of transciphering. A more complete and recent benchmark of ciphers for BGV/BFV is included in Table 2 which compares ciphers operating on both  $\mathbb{Z}_2$  and  $\mathbb{Z}_p$ . It turns out that ciphers for  $\mathbb{Z}_p$  outperforms ciphers for  $\mathbb{Z}_2$ .

However, there is a lack of benchmark comparisons of different HEFCs across various HE libraries when applied to different use cases. Consequently, the implications of applying them to any specific use case are not yet fully understood. Specifically, their inefficiency for q > 2 (which is necessary for many use cases, e.g., [36, 143, 144]) has not been realized. Once q is chosen, it cannot be changed without the secret decryption key or bootstrapping (which many major HE libraries still do not support). Therefore, to use one of the HEFCs over  $\mathbb{Z}_2$ , BGV/BFV must be instantiated with q = 2 to evaluate the boolean decryption circuit of these ciphers. This results in evaluating the use case in  $\mathbb{Z}_2$ , which requires constructing binary circuits with significantly greater multiplicative depth to achieve integer arithmetic. Consequently, implementing transciphering in such use cases over integers leads to a substantial performance loss compared to implementing the use case with only HE, without transciphering. [89]

Table 3: Runtime of Trivium and Kreyvium for TFHE, using TFHE-rs library [145, Table 2]

Ciphor	Warm-Up	Latency	Throughput	Transciphering	
Cipiter	(ms)	(ms)	(bit/s)	(ms)	
Trivium	2259	121	529	259	
Kreyvium	2883	150	427	291	

Table 4: Performance comparison of the RtF transciphering framework with 128-bit Rubato to Hera, LWEs- to-RLWE conversion (denoted by LWE) and the CKKS-only environment; parameter N in parentheses implies the dimension of LWE [93]

Ciphor	N	l	Ciphertext		Client		Server		Dita of provision
Cipiter			Size	Expansion	Latency	Throughput	Latency	Throughput	Dits of precision
			(KB)	Ratio	$(\mu s)$	(MB/s)	(s)	(KB/s)	
RtF-Rubato	$2^{16}$	$2^{16}$	0.183	1.26	4.585	31.04	106.4	6.712	18.9
RtF-HERA	$2^{16}$	$2^{16}$	0.055	1.24	1.520	25.26	141.58	5.077	19.1
LWE	$2^{16}(2^{10})$	$2^{10}$	0.007	4.84	21.91	0.051	65.88	0.010	9.3
CKKS	$2^{14}$	$2^{14}$	468	23.25	9656	2.035	1	none	19.1

### 5.2 **TFHE**

The ciphers in Section 4.2 has been mostly implemented with TFHE due to their bootstrapping-friendly design.

While the FLIP and its subsequent variants has been steadily improved in terms of performance. Note that Elisabeth-4 was completely compromised, FiLIP is the most efficient and secure variant of FLIP. In [88], an instance of FiLIP is reported to reach a runtime of 134 ms per bit in Concrete library.

As an ISO standard, Trivium and its variant Kreyvium have been implemented for various HE schemes. Implementing with TFHE in TFHE-rs library [145], Trivium and Kreyvium achieve a transciphering speed of under 300 ms per 64-bit plaintext block, as shown in Table 3. Based on the benchmarks, the authors of [145] suggested that the standardized cipher Trivium and its variant Kreyvium are good enough for transciphering TFHE.

### 5.3 CKKS

A preliminary benchmark of Hera was conducted in the design paper [52], where the authors compared the implementation of Hera with CKKS to those of LWEs-to-RLWE conversions [146] and CKKS itself. It turns out that Hera with CKKS achieves a 23 times smaller ciphertext expansion ratio, 9085 times lower latency and 17.8 times higher throughput on the client side than the CKKS-only environment.

While Rubato was proposed to improve the efficiency of Hera, Rubato enjoys a low multiplicative depth (2 to 5) and a small number of multiplications per encrypted word (2.1 to 6.25) at the cost of slightly larger ciphertext expansion (1.26 to 1.31). In [93], the authors compared benchmarks of Rubato, Hera, implementation of LWEs-to-RLWE conversions [146] and CKKS itself, summarized in Table ??. Compared to Hera within the RtF framework, client-side and server-side throughput is improved by 22.9% and 32.2%, respectively, at the cost of only 1.6% larger ciphertext.

# 6 Real-world Employment of HEFCs

This section identifies the challenges for real-world employment of HEFCs and discusses probable future work directions on transciphering.

**Security analysis** A major concern of using a non-standard symmetric cipher is its security strength. As designing HEFCs is relatively new and not well-understood, the designs tend to be prone to new attacks on their new design elements. We have seen full-round attacks on some versions of LowMC, FLIP, Agrasta, Elisabeth and Chaghri. This reminds us that cryptanalysis is still the main approach for evaluating the security levels of symmetric ciphers.

**Research Problem 4.** More cryptanalysis of the proposed ciphers is required. Due to the deployment of low-degree components, algebraic attacks have many successes in breaking HEFCs. It is still an open problem to propose novel algebraic attacks by leveraging the structural properties of HEFCs. Moreover, it is desirable to see new algebraic techniques combined with statistical cryptanalytic methods, such as linear and differential attacks.

**Benchmarks and new cipher designs** Transciphering homomorphic schemes seems to be a good promise for the future of digital privacy. In the near future, we expect that the area will constantly develop. However, we notice some gaps between theory and practice:

- The lack of diversity and benchmarks of ciphers for approximate HE such as CKKS hinders the application of transciphering with approximate HE.
- Most HEFCs have been created to optimize the efficiency on software platforms. However, for symmetric primitives hardware platforms have a very significant impact on the performance. This is evident from the standard algorithms like AES. But there is no benchmarks on hardware platforms for HEFCs. One reason for this is that researchers are still continuously improving the efficiency of HE on software platforms. Nevertheless, several hardware accelerators have been developed for HE such as BASALISC for BGV [147]. But this has not been reflected in the design of HEFCs. It is an open problem to leverage the hardware architecture in the design of HEFCs.
- While Section 5 focuses on benchmarks of ciphers, it is more relevant in practice to perform benchmark of *use cases* such as privacy-preserving machine learning. Some preliminary works can be found in [89, 88].

Given the above situations, the following research direction could be exploited in the future.

**Research Problem 5.** With a better understanding of the bottlenecks we need to look for new designs, aiming for significant progresses from the state-of-the-art designs. It is desirable to have application-specific and platform-aware designs for the better efficiency.

**Standardization** Standardization of ciphers is the long term goal in the development of symmetric cryptography, which is the foundation for the widespread employment of a symmetric cipher. However, the situation of HEFCs is tricky in many aspects. There are mainly two obstacles for the standardizing HEFCs:

- Since HEFCs are intentionally tailored to HE schemes, a standardization of HE schemes is a prerequisite for that of HEFCs. There has been significant academic and commercial effort towards developing standards for HE. In 2017, an initiative called HomomorphicEncryption.org was launched. As of 2024, there is an ongoing effort to formally standardize FHE schemes by ISO/IEC. The schemes expected to be standardized include BGV, BFV, TFHE and CKKS with their variants [142].
- For NIST standards Ascon and AES, it takes many years of intensive cryptanalysis to gain confidence in their security. But this is not the case for most HEFCs, which again calls for more efforts in cryptanalysis of HEFCs.

## References

- [1] The Guardian. Tech giants may be huge, but nothing matches big data, 2013.
- [2] Softonic. Microsoft Azure suffers the biggest security breach in its history, 2024.
- [3] Economic Times. Bank of America names Infosys US unit for over 57,000 users data leak, 2024.
- [4] WION. India: Complaints of identity theft and fake profiles up by 53.8% in delhi, 2023.
- [5] Economic Times. Stolen data of 600,000 Indians sold on bot markets so far: Study, 2022.
- [6] Global News. Hackers steal children's school photos following a privacy breach, 2024.
- [7] Wikipedia. List of data breaches, 2023.
- [8] Statista. Global Data Creation is About to Explode, 2019.
- [9] OpenAI. ChatGPT, 2022.
- [10] GitHub. Copilot, 2021.
- [11] AAG. The Latest Cloud Computing Statistics, 2024.
- [12] Statista. Global hosting and cloud computing market 2010-2020, 2022.
- [13] European Union. General Data Protection Regulation (GDPR). Online; accessed on 24th May 2025, 2018.
- [14] State of California. California Consumer Privacy Act (CCPA), 2020.

- [15] India. Digital Personal Data Protection (DPDP), 2023.
- [16] Morris Dworkin. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, 2007.
- [17] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, 2018.
- [18] Intel® Software Guard Extensions, May 2024. [Online; accessed 1. Jun. 2024].
- [19] IT Deployment, Field Services, Maintenance, and Hardware Solutions, April 2024. [Online; accessed 1. Jun. 2024].
- [20] Arm Ltd. TrustZone for Cortex-A Arm(R), June 2024. [Online; accessed 1. Jun. 2024].
- [21] Stephan van Schaik, Alex Seto, Thomas Yurek, Adam Batori, Bader AlBassam, Christina Garman, Daniel Genkin, Andrew Miller, Eyal Ronen, and Yuval Yarom. SoK: SGX.Fail: How stuff get eXposed, 2022.
- [22] Yeongjin Jang, Jaehyuk Lee, Sangho Lee, and Taesoo Kim. Sgx-bomb: Locking down the processor via rowhammer attack. In Workshop on System Software for Trusted Execution, pages 1–6. ACM, 2017.
- [23] Jaehyuk Lee, Jinsoo Jang, Yeongjin Jang, Nohyun Kwak, Yeseul Choi, Changho Choi, Taesoo Kim, Marcus Peinado, and Brent ByungHoon Kang. Hacking in darkness: Return-oriented programming against secure enclaves. In USENIX Security Symposium, pages 523–539. USENIX Association, 2017.
- [24] Abbas Acar, Hidayet Aksu, A Selcuk Uluagac, and Mauro Conti. A Survey on Homomorphic Encryption Schemes: Theory and Implementation. ACM Computing Surveys (Csur), 51(4):1–35, 2018.
- [25] Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, pages 169–180. Citeseer, 1978.
- [26] Craig Gentry. A fully homomorphic encryption scheme. Ph.D. thesis, Stanford university, 2009.
- [27] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Symposium on Theory of Computing, pages 169–178. ACM, 2009.
- [28] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachene. Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds. In Advances in Cryptology – ASIACRYPT 2016, volume 10031 of LNCS, pages 3–33, Berlin, Heidelberg, 2016. Springer.
- [29] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In Advances in Cryptology – ASIACRYPT 2017, volume 10624 of LNCS, pages 409–437, Cham, 2017. Springer.
- [30] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully Homomorphic Encryption without Bootstrapping. Cryptology ePrint Archive, Paper 2011/277, 2011.
- [31] Junfeng Fan and Frederik Vercauteren. Somewhat Practical Fully Homomorphic Encryption. Cryptology ePrint Archive, Paper 2012/144, 2012.
- [32] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *Theory of Cryptography*, volume 3378 of *LNCS*, pages 325–341, Berlin, Heidelberg, 2005. Springer.
- [33] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Advances in Cryptology EUROCRYPT '99, volume 1592 of LNCS, pages 223–238, Berlin, Heidelberg, 1999. Springer.
- [34] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. In *Communications of the ACM*, volume 21, pages 120–126, New York, USA, 1978. ACM.
- [35] Jung Hee Cheon, Miran Kim, and Kristin Lauter. Homomorphic computation of edit distance. In Financial Cryptography and Data Security, volume 8946 of LNCS, pages 194–212, Berlin, Heidelberg, 2015. Springer.
- [36] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. GAZELLE: A Low Latency Framework for Secure Neural Network Inference. In USENIX Security Symposium, pages 1651– 1669, USA, 2018. USENIX Association.

- [37] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *Cloud Computing Security Workshop*, pages 113–124, New York, USA, 2011. ACM.
- [38] Josh Benaloh, Melissa Chase, Eric Horvitz, and Kristin Lauter. Patient controlled encryption: ensuring privacy of electronic medical records. In *Proceedings of the 2009 ACM workshop on Cloud computing security*, pages 103–114, Chicago, Illinois, USA, 2009. ACM.
- [39] Kristin Lauter, Adriana López-Alt, and Michael Naehrig. Private computation on encrypted genomic data. In *Progress in Cryptology - LATINCRYPT 2014*, volume 8895 of *LNCS*, pages 3–27, Cham, 2015. Springer.
- [40] Shuang Wang, Yuchen Zhang, Wenrui Dai, Kristin Lauter, Miran Kim, Yuzhe Tang, Hongkai Xiong, and Xiaoqian Jiang. Healer: homomorphic computation of exact logistic regression for secure rare disease variants analysis in gwas. In *Bioinformatics*, volume 32, pages 211–218. Oxford University Press, 2016.
- [41] Miran Kim, Yongsoo Song, and Jung Hee Cheon. Secure searching of biomarkers through hybrid homomorphic encryption scheme. In *iDASH Privacy and Security Workshop 2016*, volume 10, pages 69–76. BMC Medical Genomics, 2017.
- [42] Miran Kim, Yongsoo Song, Shuang Wang, Yuhou Xia, Xiaoqian Jiang, et al. Secure logistic regression based on homomorphic encryption: Design and evaluation. In *JMIR Medical Informatics*, volume 6, page e19, Toronto, Canada, 2018. JMIR Publications.
- [43] Daniele Micciancio. A first glimpse of cryptography's Holy Grail. In Communications of the ACM, volume 53, pages 96–96, New York, USA, 2010. ACM.
- [44] Joon-Woo Lee, HyungChul Kang, Yongwoo Lee, Woosuk Choi, Jieun Eom, Maxim Deryabin, Eunsang Lee, Junghyun Lee, Donghoon Yoo, Young-Sik Kim, et al. Privacy-preserving machine learning with fully homomorphic encryption for deep neural network. In *IEEE Access*, volume 10, pages 30039–30054, USA, 2022. IEEE.
- [45] Shahriar Badsha, Xun Yi, and Ibrahim Khalil. A practical privacy-preserving recommender system. Data Science and Engineering, 1:161–177, 2016.
- [46] Cagatay Karabat, Mehmet Sabir Kiraz, Hakan Erdogan, and Erkay Savas. Thrive: threshold homomorphic encryption based secure and privacy preserving biometric verification system. EURASIP Journal on Advances in Signal Processing, 2015:1–18, 2015.
- [47] Xiaoqiang Sun, Peng Zhang, Mehdi Sookhak, Jianping Yu, and Weixin Xie. Utilizing fully homomorphic encryption to implement secure medical computation in smart cities. *Personal and Ubiquitous Computing*, 21:831–839, 2017.
- [48] Hsin-Tsung Peng, William WY Hsu, Jan-Ming Ho, and Min-Ruey Yu. Homomorphic encryption application on financial cloud framework. In 2016 IEEE Symposium Series on Computational Intelligence (SSCI), pages 1–5. IEEE, 2016.
- [49] Chiara Marcolla, Victor Sucasas, Marc Manzano, Riccardo Bassoli, Frank HP Fitzek, and Najwa Aaraj. Survey on Fully Homomorphic Encryption, Theory, and Applications. *Proceedings of the IEEE*, 110(10):1572–1609, 2022.
- [50] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Advances in Cryptology CRYPTO 2013, volume 8042 of LNCS, pages 75–92, Berlin, Heidelberg, 2013. Springer.
- [51] Léo Ducas and Daniele Micciancio. Fhew: bootstrapping homomorphic encryption in less than a second. In Advances in Cryptology – EUROCRYPT 2015, volume 9056 of LNCS, pages 617–640, Berlin, Heidelberg, 2015. Springer.
- [52] Jihoon Cho, Jincheol Ha, Seongkwang Kim, ByeongHak Lee, Joohee Lee, Jooyoung Lee, Dukjae Moon, and Hyojin Yoon. Transciphering framework for approximate homomorphic encryption. In Advances in Cryptology – ASIACRYPT 2021, volume 13092 of LNCS, pages 640–669, Cham, 2021. Springer.
- [53] Miklós Ajtai. Generating hard instances of lattice problems. In ACM Symposium on Theory of Computing (STOC) '96, pages 99–108, New York, USA, 1996. ACM.
- [54] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Journal of the ACM (JACM), volume 56, pages 1–40, New York, USA, 2009. ACM.

- [55] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Advances in Cryptology-EUROCRYPT 2010: 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30-June 3, 2010. Proceedings 29, volume 6110 of LNCS, pages 1–23, Berlin, Heidelberg, 2010. Springer.
- [56] Vinod Vaikuntanathan. Homomorphic Encryption References. Online. Accessed: 9th Feb 2024, 2020.
- [57] Craig Gentry and Shai Halevi. Implementing gentry's fully-homomorphic encryption scheme. In Advances in Cryptology – EUROCRYPT 2011, volume 6632 of LNCS, pages 129–148, Berlin, Heidelberg, 2011. Springer.
- [58] Marten Van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Advances in Cryptology – EUROCRYPT 2010, volume 6110 of LNCS, pages 24–43, Berlin, Heidelberg, 2010. Springer.
- [59] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. Cryptology ePrint Archive, Paper 2011/344, 2011.
- [60] Zvika Brakerski. Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP. In Advances in Cryptology – CRYPTO 2012, volume 7417 of LNCS, pages 868–886, Berlin, Heidelberg, 2012. Springer.
- [61] Craig Gentry, Shai Halevi, and Nigel P Smart. Homomorphic evaluation of the aes circuit. In Advances in Cryptology – CRYPTO 2012, volume 7417 of LNCS, pages 850–867, Berlin, Heidelberg, 2012. Springer.
- [62] Craig Gentry, Shai Halevi, and Nigel P Smart. Better bootstrapping in fully homomorphic encryption. In *Public Key Cryptography – PKC 2012*, volume 7293 of *LNCS*, pages 1–16, Berlin, Heidelberg, 2012. Springer.
- [63] Craig Gentry, Shai Halevi, and Nigel Smart. Fully homomorphic encryption with polylog overhead. In Advances in Cryptology – EUROCRYPT 2012, volume 7237 of LNCS, pages 465–482, Berlin, Heidelberg, 2012. Springer.
- [64] Nigel P Smart and Frederik Vercauteren. Fully homomorphic simd operations. In Designs, Codes and Cryptography, volume 71, pages 57–81. Springer, 2014.
- [65] Jung Hee Cheon, Jean-Sébastien Coron, Jinsu Kim, Moon Sung Lee, Tancrede Lepoint, Mehdi Tibouchi, and Aaram Yun. Batch fully homomorphic encryption over the integers. In Advances in Cryptology – EUROCRYPT 2013, volume 7881 of LNCS, pages 315–335, Berlin, Heidelberg, 2013. Springer.
- [66] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Scale-invariant fully homomorphic encryption over the integers. In *Public-Key Cryptography – PKC 2014*, volume 8383 of *LNCS*, pages 311–328, Berlin, Heidelberg, 2014. Springer.
- [67] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-Based FHE as Secure as PKE. In ACM Innovations in Theoretical Computer Science (ITCS) 2014, pages 1–12, New York, USA, 2014. ACM.
- [68] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In Advances in Cryptology – CRYPTO 2014, volume 8616 of LNCS, pages 297–314, Berlin, Heidelberg, 2014. Springer.
- [69] Nicolas Gama, Malika Izabachene, Phong Q Nguyen, and Xiang Xie. Structural lattice reduction: Generalized worst-case to average-case reductions and homomorphic cryptosystems. In Advances in Cryptology – EUROCRYPT 2016, volume 9666 of LNCS, pages 528–558, Berlin, Heidelberg, 2016. Springer.
- [70] Yarkin Doroz, Yin Hu, and Berk Sunar. Homomorphic AES Evaluation using NTRU. Cryptology ePrint Archive, Paper 2014/039, 2014.
- [71] Tancrede Lepoint and Michael Naehrig. A Comparison of the Homomorphic Encryption Schemes FV and YASHE. In *Progress in Cryptology – AFRICACRYPT 2014*, volume 8469 of *LNCS*, pages 318–335, Cham, 2014. Springer.
- [72] Yarkın Doröz, Aria Shahverdi, Thomas Eisenbarth, and Berk Sunar. Toward practical homomorphic evaluation of block ciphers using prince. In *Financial Cryptography and Data Security*, volume 8438 of *LNCS*, pages 208–220, Berlin, Heidelberg, 2014. Springer.

- [73] Adda-Akram Bendoukha, Aymen Boudguiga, and Renaud Sirdey. Revisiting Stream-Cipher-Based Homomorphic Transciphering in the TFHE Era. In *International Symposium on Foundations and Practice of Security*, pages 19–33. Springer, 2021.
- [74] Adda Akram Bendoukha, Oana Stan, Renaud Sirdey, Nicolas Quero, and Luciano Freitas. Practical Homomorphic Evaluation of Block-Cipher-Based Hash Functions with Applications. In *Foundations and Practice of Security*, volume 13877 of *LNCS*, pages 88–103, Cham, 2022. Springer.
- [75] Ehud Aharoni, Nir Drucker, Gilad Ezov, Eyal Kushnir, Hayim Shaul, and Omri Soceanu. E2E nearstandard and practical authenticated transciphering. Cryptology ePrint Archive, Paper 2023/1040, 2023.
- [76] Alexander Viand, Christian Knabenhans, and Anwar Hithnawi. Verifiable fully homomorphic encryption. arXiv preprint arXiv:2301.07041, 2023.
- [77] Chaya Ganesh, Anca Nitulescu, and Eduardo Soria-Vazquez. Rinocchio: Snarks for ring arithmetic. Journal of Cryptology, 36(4):41, 2023.
- [78] Alexandre Bois, Ignacio Cascudo, Dario Fiore, and Dongwoo Kim. Flexible and efficient verifiable computation on encrypted data. In *IACR International Conference on Public-Key Cryptography*, volume 12711 of *LNCS*, pages 528–558, Cham, 2021. Springer.
- [79] Enrico Bottazzi. Greco: Fast Zero-Knowledge Proofs for Valid FHE RLWE Ciphertexts Formation. Cryptology ePrint Archive, Paper 2024/594, 2024.
- [80] Shahla Atapoor, Karim Baghery, Hilder V. L. Pereira, and Jannik Spiessens. Verifiable FHE via Lattice-based SNARKs. Cryptology ePrint Archive, Paper 2024/032, 2024.
- [81] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In ACM Transactions on Computation Theory (ToCT), volume 6, pages 1–36, New York, USA, 2014. ACM.
- [82] Joan Boyar, René Peralta, and Denis Pochuev. On the multiplicative complexity of boolean functions over the basis (∧, ⊕,1). In *Theoretical Computer Science*, volume 235, pages 43–57. Elsevier, 2000.
- [83] 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 Advances in Cryptology – CRYPTO 2018, volume 10991 of LNCS, pages 662–692, Cham, 2018. Springer.
- [84] Christophe De Canniere and Bart Preneel. Trivium. In New Stream Cipher Designs: The eS-TREAM Finalists, volume 4986 of LNCS, pages 244–266. Springer, Berlin, Heidelberg, 2008.
- [85] Anne Canteaut, Sergiu Carpov, Caroline Fontaine, Tancrède Lepoint, María Naya-Plasencia, Pascal Paillier, and Renaud Sirdey. Stream ciphers: A practical solution for efficient homomorphicciphertext compression. In *Journal of Cryptology*, volume 31, pages 885–916. Springer, 2018.
- [86] Pierrick Méaux, Anthony Journault, François-Xavier Standaert, and Claude Carlet. Towards Stream Ciphers for Efficient FHE with Low-Noise Ciphertexts. In Advances in Cryptology – EU-ROCRYPT 2016, volume 9665 of LNCS, pages 311–343, Berlin, Heidelberg, 2016. Springer.
- [87] Pierrick Méaux, Claude Carlet, Anthony Journault, and François-Xavier Standaert. Improved Filter Permutators for Efficient FHE: Better Instances and Implementations. In Progress in Cryptology – INDOCRYPT 2019, volume 11898 of LNCS, pages 68–91, Cham, 2019. Springer.
- [88] Orel Cosseron, Clément Hoffmann, Pierrick Méaux, and François-Xavier Standaert. Towards caseoptimized hybrid homomorphic encryption - featuring the elisabeth stream cipher. In Shweta Agrawal and Dongdai Lin, editors, 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, volume 13793 of Lecture Notes in Computer Science, pages 32–67. Springer, 2022.
- [89] Christoph Dobraunig, Lorenzo Grassi, Lukas Helminger, Christian Rechberger, Markus Schofnegger, and Roman Walch. Pasta: A case for hybrid homomorphic encryption. *IACR Transactions* on Cryptographic Hardware and Embedded Systems, 2023(3):30–73, 2023.
- [90] Phil Hebborn and Gregor Leander. Dasta Alternative Linear Layer for Rasta. In IACR Transactions on Symmetric Cryptology – ToSC 2020, volume 2020, pages 46–86, 2020.

- [91] Jincheol Ha, Seongkwang Kim, Wonseok Choi, Jooyoung Lee, Dukjae Moon, Hyojin Yoon, and Jihoon Cho. Masta: An HE-Friendly Cipher Using Modular Arithmetic. In *IEEE Access*, volume 8, pages 194741–194751, USA, 2020. IEEE.
- [92] Carlos Cid, John Petter Indrøy, and Håvard Raddum. FASTA-a stream cipher for fast FHE evaluation. In *Topics in Cryptology – CT-RSA 2022*, volume 13161 of *LNCS*, pages 451–483, Cham, 2022. Springer.
- [93] Jincheol Ha, Seongkwang Kim, ByeongHak Lee, Jooyoung Lee, and Mincheol Son. Rubato: Noisy ciphers for approximate homomorphic encryption. In Advances in Cryptology – EUROCRYPT 2022, volume 13275 of LNCS, pages 581–610, Cham, 2022. Springer.
- [94] Martin Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In Advances in Cryptology – EUROCRYPT 2015, volume 9056 of LNCS, pages 430–454, Berlin, Heidelberg, 2015. Springer.
- [95] Tomer Ashur, Mohammad Mahzoun, and Dilara Toprakhisar. Chaghri a fhe-friendly block cipher. In ACM Computer and Communications Security (CCS), page 139–150, New York, USA, 2022. ACM.
- [96] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-Quantum Zero-Knowledge and Signatures from Symmetric-Key Primitives, 2017.
- [97] NIST. Post-Quantum Cryptography Standardization. Online. Accessed 18th Mar 2024, 2016.
- [98] Itai Dinur, Yunwen Liu, Willi Meier, and Qingju Wang. Optimized Interpolation Attacks on LowMC. In Advances in Cryptology – ASIACRYPT 2015, volume 9453 of LNCS, pages 535–560, Berlin, Heidelberg, 2015. Springer.
- [99] Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. Higher-Order Cryptanalysis of LowMC. In *Information Security and Cryptology - ICISC 2015*, volume 9558 of *LNCS*, pages 87–101, Cham, 2016. Springer.
- [100] Martin Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. Cryptology ePrint Archive, Paper 2016/687, 2016.
- [101] Christian Rechberger, Hadi Soleimany, and Tyge Tiessen. Cryptanalysis of Low-Data Instances of Full LowMCv2. In IACR Transactions on Symmetric Cryptology – ToSC 2018, pages 163–181, 2018.
- [102] Martin Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. FewMul, 2017.
- [103] Subhadeep Banik, Khashayar Barooti, F Betul Durak, and Serge Vaudenay. Cryptanalysis of LowMC instances using single plaintext/ciphertext pair. In *IACR Transactions on Symmetric Cryptology (ToSC)*, volume 2020, pages 130–146, 2020.
- [104] Subhadeep Banik, Khashayar Barooti, Serge Vaudenay, and Hailun Yan. New Attacks on LowMC Instances with a Single Plaintext/Ciphertext Pair. In Advances in Cryptology – ASIACRYPT 2021, volume 13090 of LNCS, pages 303–331, Cham, 2021. Springer.
- [105] Itai Dinur. Cryptanalytic Applications of the Polynomial Method for Solving Multivariate Equation Systems over GF(2). In Advances in Cryptology – EUROCRYPT 2021, volume 12696 of LNCS, pages 374–403, Cham, 2021. Springer.
- [106] Fukang Liu, Takanori Isobe, and Willi Meier. Cryptanalysis of Full LowMC and LowMC-M with Algebraic Techniques. In Advances in Cryptology – CRYPTO 2021, volume 12827 of LNCS, pages 368–401, Cham, 2021. Springer.
- [107] Fukang Liu, Takanori Isobe, and Willi Meier. Low-memory algebraic attacks on round-reduced LowMC, 2021.
- [108] Subhadeep Banik, Khashayar Barooti, Andrea Caforio, and Serge Vaudenay. Memory-Efficient Single Data-Complexity Attacks on LowMC Using Partial Sets. Cryptology ePrint Archive, Paper 2022/688, 2022.
- [109] Fukang Liu, Willi Meier, Santanu Sarkar, and Takanori Isobe. New Low-Memory Algebraic Attacks on LowMC in the Picnic Setting. In *IACR Transactions on Symmetric Cryptology – ToSC 2022*, pages 102–122, 2022.

- [110] Yimeng Sun, Jiamin Cui, and Meiqin Wang. Improved Attacks on LowMC with Algebraic Techniques. Cryptology ePrint Archive, Paper 2023/1718, 2023.
- [111] Alex Biryukov, Charles Bouillaguet, and Dmitry Khovratovich. Cryptographic schemes based on the asasa structure: Black-box, white-box, and public-key. In Advances in Cryptology-ASIACRYPT 2014: 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, ROC, December 7-11, 2014. Proceedings, Part I 20, volume 8873 of LNCS, pages 63–84, Berlin, Heidelberg, 2014. Springer.
- [112] Fukang Liu, Santanu Sarkar, Willi Meier, and Takanori Isobe. Algebraic Attacks on Rasta and Dasta Using Low-Degree Equations. In Advances in Cryptology – ASIACRYPT 2021, volume 13090 of LNCS, pages 214–240, Cham, 2021. Springer.
- [113] Abdelrahaman Aly, Tomer Ashur, Eli Ben-Sasson, Siemen Dhooghe, and Alan Szepieniec. Design of symmetric-key primitives for advanced cryptographic protocols. In *IACR Transactions on Symmetric Cryptology (ToSC)*, volume 2020, pages 1–45, 2020.
- [114] Fukang Liu, Ravi Anand, Libo Wang, Willi Meier, and Takanori Isobe. Coefficient grouping: Breaking chaghri and more. In Advances in Cryptology – EUROCRYPT 2023, volume 14007 of LNCS, pages 287–317, Cham, 2023. Springer.
- [115] ECRYPT. The estream stream cipher project, 2005.
- [116] Pierre-Alain Fouque and Thomas Vannet. Improving Key Recovery to 784 and 799 Rounds of Trivium Using Optimized Cube Attacks. In *Fast Software Encryption – FSE*, volume 8424 of *LNCS*, pages 502–517, Berlin, Heidelberg, 2013. Springer.
- [117] Itai Dinur and Adi Shamir. Cube attacks on tweakable black box polynomials. In Advances in Cryptology - EUROCRYPT 2009, volume 5479 of LNCS, pages 278–299, Berlin, Heidelberg, 2009. Springer.
- [118] Michael Vielhaber. Breaking ONE.FIVIUM by AIDA an Algebraic IV Differential Attack. Cryptology ePrint Archive, Paper 2007/413, 2007.
- [119] Alexander Maximov and Alex Biryukov. Two Trivial Attacks on Trivium. In Selected Areas in Cryptography, volume 4876 of LNCS, pages 36–55, Berlin, Heidelberg, 2007. Springer.
- [120] Abhishek Kesarwani, Dibyendu Roy, Santanu Sarkar, and Willi Meier. New cube distinguishers on nfsr-based stream ciphers. In *Designs, Codes and Cryptography*, volume 88, pages 173–199. Springer, 2020.
- [121] Meicheng Liu. Degree Evaluation of NFSR-Based Cryptosystems. In Advances in Cryptology CRYPTO 2017, volume 10403 of LNCS, pages 227–249, Cham, 2017. Springer.
- [122] Yosuke Todo, Takanori Isobe, Yonglin Hao, and Willi Meier. Cube attacks on non-blackbox polynomials based on division property. In Advances in Cryptology – CRYPTO 2017, volume 10403 of LNCS, pages 250–279, Cham, 2017. Springer.
- [123] Qingju Wang, Yonglin Hao, Yosuke Todo, Chaoyun Li, Takanori Isobe, and Willi Meier. Improved division property based cube attacks exploiting algebraic properties of superpoly. In Advances in Cryptology – CRYPTO 2018, volume 10991 of LNCS, pages 275–305, Cham, 2018. Springer.
- [124] Yuhei Watanabe, Takanori Isobe, and Masakatu Morii. Cryptanalysis of reduced kreyvium. In IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, volume 101, pages 1548–1556. IEICE, 2018.
- [125] Dibyendu Roy, Bhagwan Bathe, and Subhamoy Maitra. Differential Fault Attack on Kreyvium & FLIP. In *IEEE Transactions on Computers*, volume 70, pages 2161–2167, USA, 2021. IEEE.
- [126] Jiahui He, Kai Hu, Hao Lei, and Meiqin Wang. Massive superpoly recovery with a meet-in-themiddle framework – improved cube attacks on trivium and kreyvium. Cryptology ePrint Archive, Paper 2024/342, 2024.
- [127] Pierrick Méaux. Symmetric encryption scheme adapted to fully homomorphic encryption scheme. Coding and Cryptography Days 2015, 2015.
- [128] Sébastien Duval, Virginie Lallemand, and Yann Rotella. Cryptanalysis of the FLIP Family of Stream Ciphers. In Advances in Cryptology – CRYPTO 2016, volume 9814 of LNCS, pages 457– 475, Berlin, Heidelberg, 2016. Springer.

- [129] Henri Gilbert, Rachelle Heim Boissier, Jérémy Jean, and Jean-René Reinhard. Cryptanalysis of Elisabeth-4. In Advances in Cryptology – ASIACRYPT 2023, volume 14440 of LNCS, pages 256–284, Singapore, 2023. Springer.
- [130] Clément Hoffmann, Pierrick Méaux, and François-Xavier Standaert. The patching landscape of elisabeth-4 and the mixed filter permutator paradigm. In *International Conference on Cryptology* in *India*, pages 134–156. Springer, 2023.
- [131] Christina Boura, Nicolas Gama, Mariya Georgieva, and Dimitar Jetchev. Simulating homomorphic evaluation of deep learning predictions. In *International Symposium on Cyber Security Cryptogra*phy and Machine Learning, volume 11527 of LNCS, pages 212–230, Cham, 2019. Springer.
- [132] Martin Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. Mimc: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In *International Conference on the Theory and Application of Cryptology and Information Security*, volume 10031 of *LNCS*, pages 191–219, Berlin, Heidelberg, 2016. Springer.
- [133] Tomer Ashur and Siemen Dhooghe. Marvellous: a stark-friendly family of cryptographic primitives. Cryptology ePrint Archive, Paper 2018/1098, 2018.
- [134] Lorenzo Grassi, Christian Rechberger, Dragos Rotaru, Peter Scholl, and Nigel P Smart. Mpcfriendly symmetric key primitives. In ACM SIGSAC Conference on Computer and Communications Security (CCS) 2016, pages 430–443, New York, USA, 2016. ACM.
- [135] Fukang Liu, Abul Kalam, Santanu Sarkar, and Willi Meier. Algebraic Attack on FHE-Friendly Cipher HERA Using Multiple Collisions. Cryptology ePrint Archive, Paper 2023/1800, 2023.
- [136] 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. In Advances in Cryptology – CRYPTO 2023, volume 14083 of LNCS, pages 305–339, Cham, 2023. Springer.
- [137] Shai Halevi and Victor Shoup. Design and implementation of helib: a homomorphic encryption library. Cryptology ePrint Archive, Paper 2020/1481, 2020.
- [138] Shai Halevi and Victor Shoup. Algorithms in helib. In Advances in Cryptology CRYPTO 2014, volume 8616 of LNCS, pages 554–571, Berlin, Heidelberg, 2014. Springer.
- [139] Microsoft SEAL. https://github.com/Microsoft/SEAL, January 2023. Microsoft Research, Redmond, WA.
- [140] Zama. Concrete: TFHE Compiler that converts python programs into FHE equivalent, 2022. https://github.com/zama-ai/concrete.
- [141] Zama. TFHE-rs: A Pure Rust Implementation of the TFHE Scheme for Boolean and Integer Arithmetics Over Encrypted Data, 2022. https://github.com/zama-ai/tfhe-rs.
- [142] Jean-Philippe Bossuat, Rosario Cammarota, Jung Hee Cheon, Ilaria Chillotti, Benjamin R Curtis, Wei Dai, Huijing Gong, Erin Hales, Duhyeong Kim, Bryan Kumara, et al. Security guidelines for implementing homomorphic encryption. *Cryptology ePrint Archive*, 2024.
- [143] Kelong Cong, Radames Cruz Moreno, Mariana Botelho da Gama, Wei Dai, Ilia Iliashenko, Kim Laine, and Michael Rosenberg. Labeled PSI from Homomorphic Encryption with Reduced Computation and Communication. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, pages 1135–1150, 2021.
- [144] Alexandros Bampoulidis, Alessandro Bruni, Lukas Helminger, Daniel Kales, Christian Rechberger, and Roman Walch. Privately Connecting Mobility to Infectious Diseases via Applied Cryptography. arXiv preprint arXiv:2005.02061, 2020.
- [145] Thibault Balenbois, Jean-Baptiste Orfila, and Nigel Smart. Trivial transciphering with trivium and tfhe. In Proceedings of the 11th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, pages 69–78, 2023.
- [146] Wen-jie Lu, Zhicong Huang, Cheng Hong, Yiping Ma, and Hunter Qu. Pegasus: Bridging polynomial and non-polynomial evaluations in homomorphic encryption. In 2021 IEEE Symposium on Security and Privacy (SP), pages 1057–1073, USA, 2021. IEEE.

- [147] Robin Geelen, Michiel Van Beirendonck, Hilder Vitor Lima Pereira, Brian Huffman, Tynan McAuley, Ben Selfridge, Daniel Wagner, Georgios Dimou, Ingrid Verbauwhede, Frederik Vercauteren, et al. Basalisc: programmable hardware accelerator for bgv fully homomorphic encryption. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2023(4):32–57, 2023.
- [148] David A. McGrew and John Viega. The security and performance of the galois/counter mode (GCM) of operation. In Anne Canteaut and Kapalee Viswanathan, editors, Progress in Cryptology - INDOCRYPT 2004, 5th International Conference on Cryptology in India, Chennai, India, December 20-22, 2004, Proceedings, volume 3348 of Lecture Notes in Computer Science, pages 343-355. Springer, 2004.
- [149] Phillip Rogaway. Authenticated-encryption with associated-data. In Vijayalakshmi Atluri, editor, Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, Washington, DC, USA, November 18-22, 2002, pages 98–107. ACM, 2002.
- [150] Christina Boura and Maria Naya-Plasencia. Symmetric Cryptography, Volume 1: Design and Security Proofs. John Wiley & Sons, 2023.
- [151] Dilara Toprakhisar and Tomer Ashur. A Comparative Study of Vision and AES in FHE Setting. In 41st WIC Symposium on Information Theory and Signal Processing in the Benelux, SITB2021, page 110, Eindhoven, Netherlands, 2021. TU Eindhoven.
- [152] Kaisa Nyberg. Differentially uniform mappings for cryptography. In Advances in Cryptology EUROCRYPT '93, volume 765 of LNCS, pages 55–64, Berlin, Heidelberg, 1993. Springer.
- [153] Doreen Hertel. A Note on the Kasami Power Function. Cryptology ePrint Archive, Paper 2005/436, 2005.
- [154] Hans Dobbertin. Almost Perfect Nonlinear Power Functions on  $GF(2^n)$ : The Niho Case. In *Information and Computation*, volume 151, pages 57–72. Elsevier, 1999.
- [155] IACR. Lowmc cryptanalysis challenge, 2020.
- [156] R Radheshwar, Meenakshi Kansal, Pierrick Méaux, and Dibyendu Roy. Differential Fault Attack on Rasta and FiLIP<sub>DSM</sub>. In *IEEE Transactions on Computers*, volume 72, pages 2418–2425, USA, 2023. IEEE.
- [157] Jung Hee Cheon, Jinhyuck Jeong, Joohee Lee, and Keewoo Lee. Privacy-preserving computations of predictive medical models with minimax approximation and non-adjacent form. In *Financial Cryptography and Data Security*, pages 53–74. Springer, 2017.
- [158] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. In Journal of CRYPTOLOGY, volume 4, pages 3–72. Springer, 1991.
- [159] Thomas Jakobsen and Lars R Knudsen. The interpolation attack on block ciphers. In Fast Software Encryption, volume 1267 of LNCS, pages 28–40, Berlin, Heidelberg, 1997. Springer.
- [160] Bruno Buchberger. Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal. Ph. D. Thesis, Math. Inst., University of Innsbruck, 1965.
- [161] Xuejia Lai. Higher order derivatives and differential cryptanalysis. In Communications and Cryptography: Two Sides of One Tapestry, volume 276 of SECS, pages 227–233, Boston, MA, 1994. Springer.
- [162] Lars Knudsen and David Wagner. Integral cryptanalysis. In Fast Software Encryption (FSE), volume 2365 of LNCS, pages 112–127, Berlin, Heidelberg, 2002. Springer.
- [163] Fukang Liu, Santanu Sarkar, Gaoli Wang, Willi Meier, and Takanori Isobe. Algebraic meet-in-themiddle attack on LowMC. In International Conference on the Theory and Application of Cryptology and Information Security, pages 225–255. Springer, 2022.
- [164] Christoph Dobraunig, Farokhlagha Moazami, Christian Rechberger, and Hadi Soleimany. Framework for faster key search using related-key higher-order differential properties: applications to Agrasta. In *IET Information Security*, volume 14, pages 202–209, USA, 2020. John Wiley & Sons, Inc.
- [165] Jonathan J Hoch and Adi Shamir. Fault analysis of stream ciphers. In Cryptographic Hardware and Embedded Systems (CHES), pages 240–253. Springer, 2004.

- [166] W. Diffie and M.E. Hellman. Special Feature Exhaustive Cryptanalysis of the NBS Data Encryption Standard. In *Computer*, volume 10, pages 74–84, 1977.
- [167] Mitsuru Matsui. Linear Cryptanalysis Method for DES Cipher. In Advances in Cryptology EUROCRYPT '93, volume 765 of LNCS, pages 386–397, Berlin, Heidelberg, 1993. Springer.
- [168] Lars R Knudsen. Truncated and higher order differentials. In Fast Software Encryption FSE, volume 1008 of LNCS, pages 196–211, Berlin, Heidelberg, 1995. Springer.
- [169] Statista. Number of internet users worldwide 2022. Online. Accessed 25th Feb 2024, 2023.
- [170] Microsoft. What is ElectionGuard? Online. Accessed 29th Feb 2024, 2020.
- [171] MarketsandMarkets. Cloud Computing Market Size, Share, Growth Drivers, Opportunities & Statistics. Online. Accessed 2nd Mar 2024, 2024.
- [172] Wonkyung Jung, Sangpyo Kim, Jung Ho Ahn, Jung Hee Cheon, and Younho Lee. Over 100x faster bootstrapping in fully homomorphic encryption through memory-centric optimization with gpus. In *IACR Transactions on Cryptographic Hardware and Embedded Systems*, volume 2021, pages 114–148. IACR, 2021.
- [173] Christoph Dobraunig, Lorenzo Grassi, Anna Guinet, and Daniël Kuijsters. Ciminion: symmetric encryption based on toffoli-gates over large finite fields. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, volume 12697 of LNCS, pages 3–34, Cham, 2021. Springer.
- [174] NIST. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. Online. Accessed 18th Mar 2024, 2015.
- [175] Massimo Chenal and Qiang Tang. On Key Recovery Attacks Against Existing Somewhat Homomorphic Encryption Schemes. In Progress in Cryptology-LATINCRYPT 2014: Third International Conference on Cryptology and Information Security in Latin America Florianópolis, Brazil, September 17–19, 2014 Revised Selected Papers 3, volume 8895 of LNCS, pages 239–258, Cham, 2015. Springer.
- [176] Ilaria Chillotti, Nicolas Gama, and Louis Goubin. Attacking FHE-based applications by software fault injections. Cryptology ePrint Archive, Paper 2016/1164, 2016. https://eprint.iacr.org/ 2016/1164.
- [177] Dario Fiore, Rosario Gennaro, and Valerio Pastro. Efficiently Verifiable Computation on Encrypted Data. In Computer and Communications Security (CCS) '14, pages 844–855. ACM, 2014.
- [178] Bhuvnesh Chaturvedi, Anirban Chakraborty, Ayantika Chatterjee, and Debdeep Mukhopadhyay. A Practical Full Key Recovery Attack on TFHE and FHEW by Inducing Decryption Errors. Cryptology ePrint Archive, Paper 2022/1563, 2022. https://eprint.iacr.org/2022/1563.
- [179] Michael Walter. On side-channel and cvo attacks against the and fhew. Cryptology ePrint Archive, Paper 2022/1722, 2022. https://eprint.iacr.org/2022/1722.
- [180] Bhuvnesh Chaturvedi, Anirban Chakraborty, Ayantika Chatterjee, and Debdeep Mukhopadhyay. Demystifying the comments made on "a practical full key recovery attack on the and the by inducing decryption errors". Cryptology ePrint Archive, Paper 2022/1741, 2022. https://eprint. iacr.org/2022/1741.
- [181] Dan Boneh, Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. SIAM Journal on Computing, 36(5):1301–1328, 2007.
- [182] Jake Loftus, Alexander May, Nigel P Smart, and Frederik Vercauteren. On cca-secure somewhat homomorphic encryption. In Selected Areas in Cryptography: 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers 18, volume 7118 of LNCS, pages 55–72, Berlin, Heidelberg, 2012. Springer.
- [183] Zengpeng Li, Steven D Galbraith, and Chunguang Ma. Preventing adaptive key recovery attacks on the gsw levelled homomorphic encryption scheme. In *Provable Security: 10th International Conference, ProvSec 2016, Nanjing, China, November 10-11, 2016, Proceedings 10*, volume 10005 of *LNCS*, pages 373–383, Cham, 2016. Springer.
- [184] Junzuo Lai, Robert H Deng, Changshe Ma, Kouichi Sakurai, and Jian Weng. Cca-secure keyedfully homomorphic encryption. In Public-Key Cryptography-PKC 2016: 19th IACR International Conference on Practice and Theory in Public-Key Cryptography, Taipei, Taiwan, March 6-9, 2016, Proceedings, Part I, volume 9614 of LNCS, pages 70–98, Berlin, Heidelberg, 2016. Springer.

- [185] Keita Emura, Goichiro Hanaoka, Koji Nuida, Go Ohtake, Takahiro Matsuda, and Shota Yamada. Chosen ciphertext secure keyed-homomorphic public-key cryptosystems. *Designs, Codes and Cryp*tography, 86(8):1623–1683, 2018.
- [186] Biao Wang, Xueqing Wang, and Rui Xue. Cca1 secure fhe from pio, revisited. Cybersecurity, 1:1–8, 2018.
- [187] Keita Emura. On the security of keyed-homomorphic pke: preventing key recovery attacks and ciphertext validity attacks. *IEICE Transactions on Fundamentals of Electronics, Communications* and Computer Sciences, 104(1):310–314, 2021.
- [188] Shingo Sato, Keita Emura, and Atsushi Takayasu. Keyed-fully homomorphic encryption without indistinguishability obfuscation. In *International Conference on Applied Cryptography and Network Security*, volume 13269 of *LNCS*, pages 3–23, Cham, 2022. Springer.
- [189] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In Advances in Cryptology-CRYPTO 2010: 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings 30, volume 6223 of LNCS, pages 465–482, Berlin, Heidelberg, 2010. Springer.
- [190] Rosario Gennaro and Daniel Wichs. Fully homomorphic message authenticators. In International Conference on the Theory and Application of Cryptology and Information Security, volume 8270 of LNCS, pages 301–320, Berlin, Heidelberg, 2013. Springer.
- [191] Dario Catalano and Dario Fiore. Practical homomorphic macs for arithmetic circuits. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, volume 7881 of LNCS, pages 336–352, Berlin, Heidelberg, 2013. Springer.
- [192] Dario Fiore, Anca Nitulescu, and David Pointcheval. Boosting verifiable computation on encrypted data. In Public-Key Cryptography-PKC 2020: 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4-7, 2020, Proceedings, Part II 23, volume 12111 of LNCS, pages 124–154, Cham, 2020. Springer.
- [193] Sylvain Chatel, Christian Knabenhans, Apostolos Pyrgelis, and Jean-Pierre Hubaux. Verifiable encodings for secure homomorphic analytics. arXiv preprint arXiv:2207.14071, 2022.
- [194] Shafi Goldwasser, Yael Tauman Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run turing machines on encrypted data. In Advances in Cryptology-CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II, volume 8043 of LNCS, pages 536–553, Berlin, Heidelberg, 2013. Springer.
- [195] Christian Mouchet, Juan Troncoso-Pastoriza, Jean-Philippe Bossuat, and Jean-Pierre Hubaux. Multiparty Homomorphic Encryption from Ring-Learning-with-Errors. Proceedings on Privacy Enhancing Technologies, 2021(4):291–311, 2021.

# A A Brief Overview of Symmetric Key Cryptography

This section provides a concise introduction to symmetric cryptography. A more in-depth and updated exploration can be found in [150].

### A.1 Stream Ciphers

A (synchronous) stream cipher consists of an internal state  $\mathbf{S}$ , a state initialization function G, a state update function F, and an output function H. For an input (k, IV), the following operations are performed:

- 1. During the initialization phase, the initial state  $\mathbf{S}_0$  is generated by  $\mathbf{S}_0 = G(k, IV)$ .
- 2. At clock t, the keystream bit/word  $z_t$  is produced by

$$z_t = H(k, \mathbf{S}_t), \, \mathbf{S}_{t+1} = F(k, \mathbf{S}_t).$$

3. The message  $m_t$  is encrypted by  $c_t = m_t \oplus z_t$  to obtain the ciphertext  $c_t$ .

Similarly, decryption is performed by  $m_t = c_t \oplus z_t$ . The triple (G, F, H) is called a *keystream generator*, which generates the keystream from the input (k, IV).

### A.2 Block Ciphers

A block cipher is a keyed permutation that operates on fixed-length bit strings. For any fixed key k, a keyed permutation  $E(k, \cdot)$  and a decryption function  $D(k, \cdot)$  can be defined such that D(k, E(k, m)) = m for any message m. Most block ciphers used in practice, like AES, are constructed as iterated mappings based on *round functions*. More specifically, AES and AES-like ciphers have a round function consisting of an addition with a round key, followed by a substitution layer (the S-box layer) and a linear permutation layer (the P-layer). The S-box layer is usually implemented as a parallel application of several, not necessarily identical, small S-boxes.

### A.3 Authenticated Encryption

Authenticated encryption (AE) simultaneously ensures confidentiality and data integrity of messages between the sender and the receiver. Most existing AE schemes also allow the authentication of a public string, the associated data, along with the message. These AE schemes are also called AE with associated data (AEAD) [149]. For instance, AES in Galois Counter Mode (GCM) [148] is a widely used AEAD cryptosystem.

# **B** Details of Ciphers

### B.1 Trivium and Kreyvium

Trivium [84] (see Fig. 7) is a synchronous stream cipher with an 80-bit secret key (K) and an 80-bit initial value (IV). Its internal structure consists of three shift registers comprising 288 bits, ten XOR gates, and three AND gates for feedback. Each of these three shift registers is composed of 93, 84, and 111 bits, respectively. The K and the IV are loaded in the internal register, along with some prefixed constants. At each round, a bit is shifted into the three shift registers using a non-linear combination of taps from that and one other register; one bit of output is produced. After the first 1152 clock cycles, the cipher generates a valid pseudorandom bit sequence. Finally, the ciphertext is the result of XOR of the generated bit sequence and plaintext.

Kreyvium [85] is a variant of Trivium with a 128-bit secret key and a 128-bit initial value. It maintains an internal state consisting of three shift registers similar to Trivium, but with different lengths (128 bits, 128 bits, and 256 bits). Moreover, it incorporates additional mixing steps within the update functions for the internal state registers.



Figure 7: Trivium

### B.2 FLIP, FiLIP and Elisabeth

FLIP [86] proposed by Méaux *et al.* is based on the Filter Permutator (FP) paradigm (see Fig. 8). FP is composed of four parts: a register where the key is stored, a Pseudo Random Number Generator (PRNG) initialized with a public IV, a generator of wire-cross permutations, and a filter function that produces the keystream. For a security parameter  $\lambda$ , to encrypt  $m \leq 2^{\lambda}$  bits under a secret key  $K \in \mathbb{F}_2^n$  (such that  $w_H(K) = n/2$ ), the public parameters of the PRNG are chosen and then the following process is executed for each keystream bit  $s_i$  (for  $i \in [m]$ ):

- The PRNG is updated, its output determines the *n*-bit permutation  $P_i$  at time i,

- the keystream bit  $z_i$  is computed as  $z_i = F(P_i(K))$ .



Figure 8: Permutator paradigms: (a) for FLIP, (b) for FiLIP and (c) for Elisabeth;  $m_i$  denotes plaintext;  $z_i$  denotes the keystream;  $c_i$  denotes the ciphertext.

FiLIP [87] proposed by Méaux *et al.* is based on the Improved Filter Permutator (IFP) paradigm (see Fig. 8). The IFP contains the four parts of the filter permutator (FP) and two generators, one to select subsets of the key and one to generate fixed-size binary strings called whitenings. The main differences of these modifications to the FP paradigm are that the key register is bigger than the input size of the filter function and that a pseudorandom binary string is added to the input of the filter functions. For a security parameter  $\lambda$ , to encrypt  $m \leq 2\lambda$  bits under a secret key  $K \in \mathbb{F}_2^N$ , the public parameters of the PRNG are chosen. Then the following process is executed for each keystream bit  $z_i$  (for  $i \in [m]$ ):

- The PRNG is updated, its output determines the subset  $S_i$  of *n*-out-of-*N* elements, the permutation
  - $P_i$  from n to n elements at time i, and the whitening  $w_i$ , that is a n-size binary vector,
- the subset is applied to the key, and the permutation is applied,
- the whitening is added, and the keystream bit  $z_i$  is computed,  $z_i = F(P_i(S_i(K)) + w_i)$ .

Elisabeth [88] proposed by Cosseron *et al.* is based on Group Filter Permutator (GFP) paradigm (see Fig. 8), a generalization of the improved filter permutator where  $\mathbb{G} = \mathbb{F}_2$ . The XOR is replaced by the addition of  $\mathbb{G}$  and the Boolean function by a function from  $\mathbb{G}^n$  to  $\mathbb{G}$ . Here, each key stream symbol belongs to an additive group  $\mathbb{G}$  and is derived from a large key by a filtering function that operates on  $\mathbb{G}$ . Unlike for traditional filtered-LFSR stream ciphers, the filtering function used in Elisabeth varies at each clock, most notably in the values of additive constants called masks. The group filter permutator is defined by a group  $\mathbb{G}$  with operation noted +, a forward secure PRNG, a key size N, a subset size n, and a filtering function f from  $\mathbb{G}^n$  to  $\mathbb{G}$ . To encrypt m elements of  $\mathbb{G}$  under a secret key  $K \in \mathbb{G}^N$ , the public parameters of the PRNG are chosen and then the following process is executed for each key-stream  $s_i$  (for  $i \in [m]$ ):

- The PRNG is updated, its output determines a subset, a permutation, and a length-n vector of  $\mathbb{G}$ .
- the subset  $S_i$  is chosen, as a subset of n elements over N; the permutation  $P_i$  (a re-ordering) from n to n elements is chosen; the vector, called whitening and denoted by  $w_i$ , from  $\mathbb{G}^n$  is chosen,
- the key-stream element  $s_i$  is computed as  $s_i = f(P_i(S_i(K)) + w_i)$

### B.3 LowMC

The block cipher LowMC [94] over  $\mathbb{Z}_2$ , starts with a key whitening, followed by applying the round function r times; r depends on the chosen parameter set. A single round comprises three layers: SboxLayer, AffineLayer, and KeyAddition as shown in Fig. 9. SboxLayer is an m-fold parallel application of the same 3-bit S-box on the first 3m bits of the state. If the block size n > 3m, then for the remaining n - 3m bits, the SboxLayer is the identity. AffineLayer starts with a multiplication of the state with a pseudorandomly generated invertible binary matrix, followed by an addition of the state with a pseudorandomly generated binary vector. Finally, KeyAddition adds the state with the round key, generated by multiplying pseudorandomly generated binary matrices of full rank with the master key.



Figure 9: The round function of LowMC

### B.4 Rasta

The stream cipher Rasta [83] over  $\mathbb{F}_2$ , takes a key  $k \in \mathbb{F}_2^n$ , a nonce nc, a counter i and a message block  $m \in \mathbb{F}_2^n$  as input and returns a keystream  $z \in \mathbb{F}_2^n$  as output. The nonce and counter is then fed to an *Extendable Output Function* (XOF) to generate  $n \times n$  matrices  $M_{j,nc,i}$  and round constants  $rc_{j,nc,i} \in \mathbb{F}_2^n$  for  $j = 0, \ldots, r$ . The state of Rasta is  $\mathbb{F}_2^n$ , i.e. a vector of n elements from  $\mathbb{F}_2$ . The process of Rasta with r rounds is defined as (see Fig. 10):

 $\mathsf{Rasta}[k, nc, i] = (A_{r,nc,i} \circ S \circ A_{r-1,nc,i} \circ S \circ \cdots \circ A_{1,nc,i} \circ S \circ A_{0,nc,i}(k)) \oplus k$ 

where affine layer  $A_{i,nc,i}$  for  $j = 0, \ldots, r$  is defined as:

$$A_{j,nc,i}(x) = M_{j,nc,i} \cdot x \oplus rc_{j,nc,i}$$

and nonlinear layer S for j = 0, ..., n - 1 is defined as:

 $S(x_0, \ldots, x_{n-1}) = (y_0, \ldots, y_{n-1})$ 

$$y_j = x_j \oplus x_{j+2 \pmod{n}} \oplus x_{j+1 \pmod{n}} x_{j+2 \pmod{n}}$$



Figure 10: The r round Rasta construction; operations in the box with dotted (resp. thick) lines are public (resp. secret).

#### B.5 Chaghri

The block cipher Chaghri [95] over  $\mathbb{F}_{2^{63}}$ , takes a secret key  $k \in \mathbb{F}_{2^{63}}^3$  and a plaintext m as input. The key is then fed into the key scheduling algorithm to get 2r + 1 round keys, say  $rk_0, \ldots, rk_{2r} \in \mathbb{F}_{2^{63}}^3$ . The state of Chaghri S is  $\mathbb{F}_{2^{63}}^3$ , i.e. a vector of three elements from  $\mathbb{F}_{2^{63}}$ . The process of Chaghri with r rounds is defined as:

$$\mathsf{Chaghri}[k,m] = RF[r] \circ \cdots \circ RF[1] \circ ARK[0]$$

where the *i*-th round function RF[i] for i = 1, 2, ..., r is defined as (see Fig. 11):

$$RF[i] = ARK[2i] \circ G^{-1} \circ B^{-1} \circ M \circ ARK[2i-1] \circ G^{-1} \circ B^{-1} \circ M$$

where the nonlinear function  $G(x) : \mathbb{F}_{2^{63}} \to \mathbb{F}_{2^{63}}$  is defined as  $G(x) = x^{2^{3^2}+1}$ , the affine transform  $B(x) : \mathbb{F}_{2^{63}} \to \mathbb{F}_{2^{63}}$  is defined as  $B(x) = c_1 x^{2^{56}} + c_2 x^4 + c_3 x + c_4 \ (c_1, c_2, c_3, c_4 \in \mathbb{F}_{2^{63}}$  are constants), the linear transform  $M : \mathbb{F}_{2^{63}}^3 \to \mathbb{F}_{2^{63}}^3$  is a  $3 \times 3$  matrix and the add round key function ARK[j] for  $j = 0, \ldots, 2r$  and  $x \in \mathbb{F}_{2^{63}}^3$  is defined as  $ARK[j](x) = x + rk_j$ .





### B.6 Hera

The stream cipher Hera [52] over  $\mathbb{Z}_t$ , for  $\lambda$ -bit security takes a secret key  $k \in \mathbb{Z}_t^{16}$ , a nonce  $nc \in \{0, 1\}^{\lambda}$  as input and returns a keystream  $k_{nc} \in \mathbb{Z}_t^{16}$  as output. The state of Hera is  $\mathbb{Z}_t^{16}$ , which can also be viewed as a  $4 \times 4$  matrix over  $\mathbb{Z}_t$ . The process of Hera with r rounds is defined as:

$$\mathsf{Hera}[k, nc] = Fin[k, nc, r] \circ RF[k, nc, r-1] \circ \cdots \circ RF[k, nc, 1] \circ ARK[k, nc, 0]$$

where the *i*-th round function RF[k, nc, i] for i = 1, 2, ..., r - 1 is defined as (see Fig. 12):

$$RF[k, nc, i] = ARK[k, nc, i] \circ Cube \circ MixRows \circ MixColumns$$

and the final round function Fin[k, nc, r] is defined as:

 $Fin[k,nc,r] = ARK[k,nc,r] \circ MixRows \circ MixColumns \circ Cube \circ MixRows \circ MixColumns.$ 

The nonce is fed to the underlying XOF that outputs an element in  $(\mathbb{Z}_t^{16})^{r+1}$ , say  $rc = (rc_0, \ldots, rc_r)$ . Then the add round key function ARK[k, nc, i] for  $i = 0, 1, \ldots, r$  and  $x \in \mathbb{Z}_t^{16}$  is defined as:

$$ARK[k, nc, i](x) = x + k \bullet rc_i$$

where • (resp. +) denotes component-wise multiplication (resp. addition) modulo t. The linear map MixRows (resp. MixColumns) multiplies a certain  $4 \times 4$  matrix to each row (resp. column) of the state. Whereas, the nonlinear map  $Cube(x) = (x_0^3, \ldots, x_{15}^3)$  for  $x = (x_0, \ldots, x_{15}) \in \mathbb{Z}_t^{16}$ .



Figure 12: The round function of Hera; operations in the box with dotted (resp. thick) lines are public (resp. secret). "MC" and "MR" represent *MixColumns* and *MixRows*, respectively.

#### B.7 Rubato

The stream cipher Rubato [93] over  $\mathbb{Z}_q$ , for  $\lambda$ -bit security and a prime number q takes a secret key  $k \in \mathbb{Z}_q^n$ , a nonce  $nc \in \{0, 1\}^{\lambda}$  as input, and returns a keystream  $k_{nc} \in \mathbb{Z}_q^l$  as output for some l < n. The state of Rubato is  $\mathbb{Z}_q^n$ , which can also be viewed as a  $v \times v$  matrix over  $\mathbb{Z}_q$  where  $n = v^2$ . The process of Rubato with r rounds is defined as:

 $\mathsf{Rubato}[k, nc] = AGN \circ Fin[k, nc, r] \circ RF[k, nc, r-1] \circ \cdots \circ RF[k, nc, 1] \circ ARK[k, nc, 0]$ 

where the *i*-th round function RF[k, nc, i] for i = 1, 2, ..., r - 1 is defined as (see Fig. 13):

 $RF[k, nc, i] = ARK[k, nc, i] \circ Feistel \circ MixRows \circ MixColumns$ 

and the final round function Fin[k, nc, r] is defined as:

 $Fin[k, nc, r] = Tr_{n,l} \circ ARK[k, nc, r] \circ MixRows \circ MixColumns \circ Feistel \circ MixRows \circ MixColumns$ 

where  $Tr_{n,l}$  is the truncation of the last n-l words, i.e.  $Tr_{n,l}(x) = (x_1, \ldots, x_l)$  for  $x = (x_1, \ldots, x_n) \in \mathbb{Z}_q^n$ . The nonce is fed to the underlying XOF that outputs an element in  $(\mathbb{Z}_q^n)^{r+1}$ , say  $rc = (rc_0, \ldots, rc_r)$ . Then the add round key function ARK[k, nc, i] for  $i = 0, 1, \ldots, r$  and  $x \in \mathbb{Z}_q^n$  is defined as:

$$ARK[k, nc, i](x) = x + k \bullet rc_i$$

where • (resp. +) denotes component-wise multiplication (resp. addition) modulo q. The add Gaussian noise function AGN(x) for  $x = (x_1, \ldots, x_l) \in \mathbb{Z}_q^l$  is defined as:

$$AGN(x) = (x_1 + e_1, \dots, x_l + e_l)$$

where  $e_1, \ldots, e_l$  are sampled independently from a one-dimensional discrete Gaussian distribution  $D_{\alpha q}$  with zero mean and variance  $(\alpha q)^2/2\pi$ . The linear map MixRows (resp. MixColumns) multiplies a certain  $v \times v$  matrix to each row (resp. column) of the state. Whereas, the nonlinear map  $Feistel(x) = (x_1, x_2 + x_1^2, \ldots, x_n + x_{n-1}^2)$  for  $x = (x_1, \ldots, x_n) \in \mathbb{Z}_q^n$ .



Figure 13: The round function of Rubato; operations in the box with dotted (resp. thick) lines are public (resp. secret). "MC" and "MR" represent *MixColumns* and *MixRows*, respectively.

## C Attacks on HEFC

In this section, we give a rough idea of some significant attacks on symmetric ciphers, categorized into three main categories: statistical attacks, algebraic and structural attacks.

#### C.1 Statistical Attacks

They are performed by exploiting statistical biases in the output of a cipher or some intermediate values and/or abnormal correlation with the inputs. With the appropriate input, an attacker tries to find anomalous statistical behavior in the output and recover secret information related to the original plaintext or the secret key. These statistical attacks usually require a large data complexity to exploit this undesired statistical behavior.

Linear Cryptanalysis. It exploits linear relationships between plaintext, ciphertext, and key bits to deduce information about the secret key. In 1993, Matsui [167] applied this technique to analyze the block cipher Data Encryption Standard (DES). There are two parts to it. The first is to construct linear equations relating plaintext, ciphertext, and key bits that have a high bias, i.e., whose probabilities of holding are as close as possible to 0 or 1. The second is to use these linear equations and known plaintext/ciphertext pairs to derive the key bits.

Differential Cryptanalysis. It is the study of how differences in information input can affect the resultant difference in the output. In 1990, Biham and Shamir [158] were the first to apply this technique to the DES. Essentially, it examines differences (differentials) between pairs of plaintexts and their corresponding ciphertexts to deduce patterns and reveal information about the secret key.

#### C.2 Algebraic and Structural Attacks

As its name indicates, they are performed by exploiting algebraic techniques such as equation-solving algorithms and zero-sum properties. The attacker views the problem of recovering the secret information by modeling using some algebraic system, where the structure and solvability of the system depend on the internal design of a cipher.

Gröbner Basis Attack. It involves using Gröbner basis theory to analyze and simplify systems of polynomial equations derived from the cipher's operations. The concept behind the attack is to consider

a ciphertext c and solve the system P(x) - c = 0 for the plaintext x by searching a Gröbner basis of the ideal generated by the polynomials P(x) - c. The main cost of the Gröbner basis attack lies in the computation of the Gröbner basis. A notable approach for computing the Gröbner basis is the Buchberger algorithm [160].

Interpolation Attack. It targets the low degree of the polynomial description of a block cipher. The attacker reconstructs the polynomial description using plaintext/ciphertext pairs by means of polynomial interpolation. In 1997, Jakobsen and Knudsen [159] introduced this attack and demonstrated its effectiveness against certain ciphers with statistically strong S-boxes.

*Higher-Order Differential Cryptanalysis.* It extends the principles of differential cryptanalysis by analyzing differences in the higher-order differentials (changes in differences) between plaintext pairs and their corresponding ciphertexts to gain insights into the cryptographic algorithm and potentially recover the secret key. Lai [161] and Knudsen [168] initiated the field of higher-order differential cryptanalysis, each contributing crucial elements to its development and application.

*Cube Attack.* It was formally introduced in 2008 by Dinur and Shamir [117] as an extension of higher order differential cryptanalysis and AIDA [118]. Cube attacks exploit more subtle properties than higher-order differential properties. By carefully choosing the summing cube, one tries to get reduced polynomials with detectable properties such as (non)constanceness, linearity, balanceness, and neutrality.