IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN XXXX-XXXX, Vol. 0, No. 0, pp. 1–31. DOI:XXXXXXXX

SoK: FHE-Friendly Symmetric Ciphers and Transciphering

Chao Niu[1], Benqiang Wei[1], Zhicong Huang[1], Zhaomin Yang[1], Cheng Hong[1]*, Meiqin Wang[2] and Tao Wei[1]

Ant Group[1]
{niuchao.niu,weibenqiang.wbq,zhicong.hzc,yangzhaomin.yzm,vince.hc,lenx.wei}@antgroup.com
Shandong University, Jinan, China[2]
{mqwang}@sdu.edu.cn

Abstract. Fully Homomorphic Encryption (FHE) enables computation on encrypted data without decryption, demonstrating significant potential for privacy-preserving applications. However, FHE faces several challenges, one of which is the significant plaintext-to-ciphertext expansion ratio, resulting in high communication overhead between client and server. The transciphering technique can effectively address this problem by first encrypting data with a space-efficient symmetric cipher, then converting symmetric ciphertext to FHE ciphertext without decryption.
Numerous FHE-friendly symmetric ciphers and transciphering methods have been developed by researchers, each with unique advantages and limitations. These often require extensive knowledge of both symmetric cryptography and FHE to fully grasp, making comparison and selection among these schemes challenging. To address this, we conduct a comprehensive survey of over 20 FHE-friendly symmetric ciphers and transciphering methods, evaluating them based on criteria such as security level, efficiency, and compatibility. We have designed and executed experiments to benchmark the performance of the feasible combinations of symmetric ciphers and transciphering methods across various application scenarios. Our findings offer insights into achieving efficient transciphering tailored to different task contexts. Additionally, we make our example code available open-source, leveraging state-of-the-art FHE implementations.

Keywords: Transciphering · Fully homomorphic encryption · Symmetric cipher

1 Introduction

Since Gentry's blueprint [Gen09], fully homomorphic encryption (FHE) has been viewed as one of the most promising techniques for privacy-preserving solutions. Initially considered purely theoretical, FHE has rapidly evolved over recent years. Notable advancements include efficient modern FHE schemes like BGV [BGV14], BFV [Bra12, FV12], CGGI [CGGI16a, CGGI17], and CKKS [CKKS17]. These schemes have been implemented in open-source libraries and applied in various privacy-preserving scenarios, from breached password detection [edg] to secure genome analysis [BGPG20], and even secure classification with deep neural networks [LLL+].

There is a common perception that FHE involves prohibitively high computational costs. However, recent advancements in algorithm optimization [CLOT21, BMTPH21, OPP23] and hardware acceleration [JKA+21, KKK+22, KKC+23] suggest that these costs

*Corresponding author: Cheng Hong (vince.hc@antgroup.com)

can be significantly reduced. For instance, while directly running ResNet20 inference under FHE might take thousands of seconds, utilizing efficient approximation algorithms alongside application-specific integrated circuits (ASICs) can accomplish this task in just 30 milliseconds [PKK+23].

In addition to its high computational cost, FHE faces another significant, albeit often overlooked, challenge: the large size of its ciphertexts. To facilitate homomorphic computations on the underlying plaintext, FHE requires additional space in the ciphertext domain, leading to a substantial ciphertext expansion rate. Indeed, the CGGI scheme necessitates several kilobytes to encrypt just one bit. On the other hand, schemes like BFV, BGV, and CKKS incorporate SIMD (Single Instruction Multiple Data) operations [SV14], allowing multiple messages to be encoded into a single plaintext, which makes them more space-efficient than CGGI. However, even with SIMD operations, the ciphertext expansion rate remains notably high in typical applications.

FHE-based applications commonly operate in a client-server model where clients encrypt and send their data to the cloud servers. The high ciphertext expansion ratio significantly increases communication overhead, posing a barrier to the widespread adoption of FHE. Communication resources are not only more costly but also less scalable than computational resources. To address this issue, a technique known as transciphering[1] has been proposed. The fundamental concept of transciphering can be summarized as follows:[2]

(1) (One-time setup) The client sets up an FHE system and a symmetric encryption system and sends the FHE-encrypted symmetric secret key to the server.

(2) After the setup, instead of encrypting its data with FHE, the client could encrypt the data under symmetric encryption and send the symmetric ciphertext to the server.

(3) To support homomorphic computations, the server needs to homomorphically compute the symmetric decryption circuit, transforming the symmetric ciphertext into homomorphic ciphertext.

Since symmetric ciphers typically feature lower ciphertext expansion ratio and faster encryption efficiency, transciphering can kill two birds with one stone: it minimizes both communication costs and computational overhead on the client side. The primary challenge of transciphering lies in the high computational complexity on the server side, which is determined by the size and depth of the symmetric decryption circuit. Conventional symmetric ciphers like DES and AES are considered unfriendly for transciphering due to their complex decryption circuits. In response, researchers have focused on designing FHE-friendly symmetric ciphers (FHE-SCs) with simpler decryption circuit to reduce transciphering cost.

Currently, a variety of FHE-SCs with innovative designs have been developed. Many of these operate within the Boolean domain to reduce the per-bit encryption cost. Notable examples include LowMC [ARS+15], Kreyvium [CCF+16], FLIP [MJSC16], FiLIP [MCJS19], Rasta [DEG+18], and Dasta [HL20]. Additionally, variants tailored for specific FHE schemes have been introduced, such as Fasta [CIR22] (a variant of Rasta for HElib), Elisabeth [CHMS22, HMS23] (a variant of FiLIP), and the recently proposed FRAST [CCH+24] for CGGI. Another approach involves designing FHE-SCs with plaintext domains over prime fields $\mathbb{F}_p$, including Masta [HKC+20], Pasta [DGH+23], HERA [CHK+21], Rubato [HKL+22], and the more recent $\mathrm{Yu}_p\mathrm{X}$ [LLC+24]. Furthermore, ciphers defined over Galois extension fields, such as Chaghri [AMT22] and $\mathrm{Yu}_2\mathrm{X}$ [LLC+24], have been developed.

[1] Also known as Hybrid Homomorphic Encryption (HHE) in [NLV11, CHMS22].

[2] Methods that convert one FHE scheme to another [BGGJ20, LHH+21] are occasionally termed "transciphering", but they aim for enhanced homomorphic properties rather than a reduced expansion ratio, and thus fall outside our discussion.

In addition to substituting traditional symmetric ciphers with FHE-SCs, researchers have also invested significant effort into optimizing the transciphering algorithms themselves, aiming to homomorphically evaluate symmetric decryption circuits more efficiently. Early work by [GHS12, CCF+18] explored transciphering with the BGV scheme, while [CHK19] employed the CLT scheme [CLT14]. More recent studies [CDPP22, BPR24, TCBS23, WLW+24] have introduced efficient transciphering methods tailored for the CGGI scheme. Cho et al. [CHK+21] proposed a RtF framework which first converts real numbers to BFV and then to CKKS, while Aharoni et al. [ADE+23] directly translates AES ciphertexts to CKKS ciphertexts by exploiting the CKKS scheme.

1.1 Research Questions

Given the extensive variety of FHE-SCs and transciphering methods available, one might ask: *If one wants to implement transciphering in a specific application, which symmetric cipher and transciphering method should be chosen?* Unfortunately, the answer is not straightforward. It depends on several factors, including resource constraints on the client side, the plaintext size to be encrypted, and the nature of the subsequent FHE computational tasks. This Systematization of Knowledge paper will explore these considerations in detail and aim to achieve the following objectives:

- Providing insights to symmetric cryptographers from the FHE perspective, aiding in the design of more efficient FHE-SCs.
- Providing insights to FHE cryptographers from the symmetric encryption perspective, assisting in the selection of appropriate FHE-SCs and the development of more efficient transciphering methods.
- Providing comprehensive analysis and benchmarks of the current state-of-the-art transciphering technologies, detailing their capabilities and limitations. These results could serve as a resource for potential users interested in evaluating the applicability of these technologies for their specific needs.

We begin by outlining the technical backgrounds of FHE and transciphering. Subsequently, we categorize existing FHE-SCs into four groups based on their algorithmic properties, individually analyzing their security and compatibility with various transciphering methods. We evaluate the performance of each feasible combination of symmetric cipher and transciphering method, providing recommendations tailored to different application scenarios based on our experimental findings. Additionally, we uncover several intriguing observations that could inform the future design of FHE-SCs and transciphering methods. Our example code and documentation for all test cases are open-sourced at https://github.com/AntCPLab/awesome-transciphering.

1.2 Related Work

The Pasta design document [DGH+23] compiles benchmarks for several FHE-SCs. However, it lacks experiments involving state-of-the-art CKKS-related transciphering methods and does not discuss AES transciphering, which is a prominent topic in this field. Additionally, the document considers a relatively simple FHE application consisting of matrix operations.

Transciphering scenarios for secure multi-party computation (MPC), which transform a symmetric ciphertext into secret-shared plaintexts, have also been explored in [AGR+16, AABS+20, DGGK21, BPA+23]. These scenarios differ from FHE transciphering in several ways. Firstly, the threat model requires the client's secret key to be secret-shared among the servers in MPC, necessitating trust that the servers will not collude. Secondly, XOR and addition operations are computationally inexpensive in MPC but not in FHE. Additionally,

studies such as [GKR+21, BGK+21] investigate Zero-Knowledge Proofs (ZKP)-friendly ciphers. Christian Rechberger delivered an excellent talk [Rec] summarizing the history of FHE/MPC/ZK-friendly symmetric ciphers at STAP 2023.

Concurrent work. Two very recent works also propose a survey on transciphering. In [ANM25], the authors analyze the security properties of various transciphering schemes and conduct extensive experiments on Pasta, HERA, and Rubato. Meanwhile [TKLP25] also provides a study and categorization of existing transciphering methods; although they do not perform experiments themselves, they incorporate data from existing papers.

1.3 Organization

Section 2 introduces background knowledge pertinent to this paper. Section 3 outlines our survey methodology. Section 4 compares the features of different symmetric ciphers under transciphering. Section 5 presents the experimental results. Section 6 discusses our observations.

2 Preliminaries

In this section, we describe the necessary background knowledge of FHE and transciphering.

2.1 Fully Homomorphic Encryption

The concept of Homomorphic Encryption was introduced by Rivest et al. [RAD+78] in 1978. However, a true Fully Homomorphic Encryption (FHE) scheme that allows arbitrary computations on encrypted data was not realized until Gentry's breakthrough in 2009 [Gen09]. Today, numerous open-source FHE libraries are available within the community, including HElib [HS20], Microsoft SEAL [SEA23], PALISADE/OpenFHE [BAB+22], Lattigo [lat23], TFHE [CGGI16b], and Concrete [Zam22]. These libraries are maintained by world-class FHE experts and provide state-of-the-art implementations of the BFV, BGV, CKKS, and CGGI schemes.

Informally, an FHE scheme comprises the following four probabilistic polynomial time (PPT) algorithms:

- **KeyGen.** Generate the keys $(\mathsf{sk}, \mathsf{pk}, \mathsf{evk}) \leftarrow \mathrm{KeyGen}(1^\lambda)$. Given the security parameter λ, output the public key pk, secret key sk, and evaluation key evk.
- **Encrypt.** Given a plaintext $\mathsf{m} \in \mathbb{Z}_p$ and the public key pk, output its ciphertext $\mathsf{ct} = \mathrm{Encrypt}(\mathsf{m}, \mathsf{pk})$.
- **Decrypt.** Given a ciphertext ct and the secret key sk, output the corresponding plaintext $\mathsf{m} = \mathrm{Decrypt}(\mathsf{ct}, \mathsf{sk})$.
- **Evaluate.** Given two ciphertexts ct^0 and ct^1, a function $\mathcal{F} : \mathbb{Z}_p^* \to \mathbb{Z}_p$, and the evaluation key evk, output a new ciphertext $\mathsf{ct}' = \mathrm{Evaluate}(\mathsf{ct}^0, \mathsf{ct}^1, \mathcal{F}, \mathsf{evk})$ that decrypts to the evaluation of the underlying plaintexts of ct^0 and ct^1: $m' = \mathcal{F}(\mathsf{m}^0, \mathsf{m}^1)$.

Some FHE schemes such as BFV/BGV/CKKS additionally support the following SIMD functions:

- **SIMD Encrypt.** Given a vector $\vec{\mathsf{m}} \in \mathbb{Z}_p^N$ and the public key pk, output the ciphertext $\mathsf{ct} = \mathrm{SIMDEncrypt}(\vec{\mathsf{m}}, \mathsf{pk})$.

- **SIMD Decrypt.** Given a SIMD encrypted ciphertext ct and the secret key sk, output the corresponding plaintext vector $\vec{\mathsf{m}} = \mathrm{SIMDDecrypt}(\mathsf{ct}, \mathsf{sk})$.

- **SIMD Evaluate.** Given two SIMD encrypted ciphertexts ct^0 and ct^1, a function $\mathcal{F} : \mathbb{Z}_p^* \to \mathbb{Z}_p$, and the evaluation key evk, output a new ciphertext $\mathsf{ct}' = \mathrm{SIMDEvaluate}(\mathsf{ct}^0, \mathsf{ct}^1, \mathcal{F}, \mathsf{evk})$ that decrypts to a vector $\vec{\mathsf{m}}'$ representing the pointwise evaluation of the underlying plaintexts of ct^0 and ct^1:

$$\vec{m}'_i = \mathcal{F}(\vec{\mathsf{m}}^0_i, \vec{\mathsf{m}}^1_i) \quad \text{for all } i \in \{1, 2, \ldots, N\}.$$

  Note that FHE schemes also support homomorphic evaluations between ciphertext and plaintext. When the context is clear, we may abbreviate the operations as $\mathrm{Evaluate}(\mathsf{ct}^0, \mathsf{m}^1, \mathcal{F}, \mathsf{evk})$ or $\mathrm{SIMDEvaluate}(\mathsf{ct}^0, \vec{\mathsf{m}}^1, \mathcal{F}, \mathsf{evk})$.

Typically, FHE schemes support only a limited evaluation depth, and exceeding this limit results in a decryption error. In order to implement the computation of arbitrary multiplication depths, expensive bootstrapping procedure is required:

- **Bootstrapping.** Given a ciphertext ct that has exhausted its evaluation depth and an evaluation key evk, output a refreshed ciphertext $\mathsf{ct}' = \mathsf{Boot}(\mathsf{ct}, \mathsf{evk})$, which decrypts to the same plaintext as ct but restores the ability to support further evaluations.

FHE schemes[3] and their expansion factors. The security of the BGV [BGV14] and BFV [Bra12, FV12] schemes relies on the hardness assumption of the Learning With Errors (LWE) or Ring Learning With Errors (RLWE) problem. RLWE-based schemes enable SIMD operations, with ciphertexts represented as polynomial pairs with coefficients modulo q: $\mathsf{ct} \in R^2_{q,N}$, where $R_{q,N} = \mathbb{Z}_q[X]/(X^N + 1)$.

In the SIMD setting, a message vector $\vec{\mathsf{m}} \in \mathbb{Z}_p^N$ is encoded as a polynomial over R_p, and then encrypted as ciphertext over R^2_q, leading to an initial expansion rate of $2 \log q / \log p$.[4] However, the ciphertext expansion rate could worsen in practice. In the BGV/BFV schemes, homomorphic functions $\mathcal{F}$ operate over the field modulo p. Yet, real-world applications typically work on integers or real numbers. To ensure meaningful results, p must be large enough to accommodate the largest possible calculation output. Consequently, the original plaintext often occupies only a fraction of p, leading to a much higher real ciphertext expansion rate, frequently reaching tens or even hundreds.

The CKKS scheme [CKKS17] operates like BFV but is significantly faster. The key distinction is that CKKS does not preserve the plaintext modulus p during homomorphic operations. Instead, it employs a rescaling step to prevent overflow. The downside of this approach is that, without the "gap" between q and p, CKKS cannot distinguish small RLWE noise from the plaintext, resulting in approximate decryption rather than the exact decryption provided by BGV/BFV. Despite this, the ciphertext expansion rate of CKKS (with SIMD) remains in the range of several tens [JKLS18, HHCP18, HHW+21].

The CGGI scheme [CGGI16a] is notable for its highly efficient bootstrapping process, which is two to three orders of magnitude faster than those of the BGV, BFV, or CKKS schemes. Initially designed for binary operations, recent advancements [CJP21] have extended CGGI to support multi-bit functional bootstrapping. However, CGGI's rapid bootstrapping is incompatible with SIMD packing, in other words, a CGGI ciphertext over $\mathbb{Z}_q^N$ can encrypt only a single plaintext message $\mathbb{Z}_p$. As a result, CGGI's ciphertext expansion rate is orders of magnitude higher than that of other FHE schemes.

[3] We only introduce widely-used FHE schemes and omit older ones, such as the CLT scheme [CLT14] based on integers.

[4] This expansion rate can be optimized to $\log q / \log p$ [CLR17].

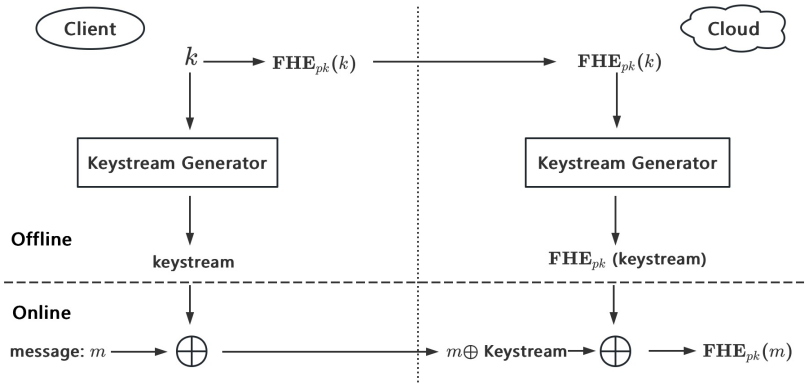

Figure 1: Transciphering framework based on stream cipher.

2.2 Transciphering Routines

SIMD transciphering. For transciphering to BFV, BGV, or CKKS, the SIMD property can be utilized to process multiple symmetric ciphertexts in parallel. Specifically, let the block size of the symmetric cipher be b, and let S_j^i denote the i^{th} value within the j^{th} block. The symmetric ciphertexts for N blocks can be represented as N vectors:

$$\vec{S_j} = (S_j^0, S_j^1, \ldots, S_j^{b-1}), \text{ where } 0 \leq j < N.$$

These N vectors can be transposed into b vectors, where each vector can be treated as an FHE plaintext:

$$\vec{\mathsf{m}}^i = (S_0^i, S_1^i, \ldots, S_{N-1}^i), \text{ where } 0 \leq i < b.$$

Since values at the same index across different blocks are typically processed similarly, SIMD operations can be applied to each $\vec{\mathsf{m}}^i$, thereby handling all N blocks simultaneously. This method is also referred to as "row packing" in [CHK+21], and it offers higher throughput compared to other packing methods. Therefore, we have chosen this method for our experiments.

The first layer of the transciphering circuit performs SIMD ciphertext-plaintext evaluations on $\vec{\mathsf{m}}^i$ using the FHE-encrypted secret key, resulting in b FHE ciphertexts. Subsequent layers execute further SIMD evaluations on these b ciphertexts to complete the transciphering process. SIMD plays a crucial role in achieving high transciphering throughput. However, its effectiveness relies on symmetric encryption modes that allow parallel block processing. For instance, modes like Cipher Block Chaining (CBC), where each block depends on the previous one, are inherently sequential and thus incompatible with SIMD-based transciphering.

Offline-online paradigm. In the introduction, we described the transciphering process as "homomorphically evaluating the decryption circuit" for simplicity. In practice, more efficient methods are often employed. One commonly used approach leverages a stream cipher [CCF+18]. Notably, block ciphers can also be adapted to function as stream ciphers by operating in counter (CTR) mode. In this framework, the plaintext is masked—typically using an XOR operation—with a pseudorandom keystream.

This approach splits the transciphering process into an offline (preprocessing) phase and an online phase, as shown in Fig. 1. In the offline phase, the server uses the FHE-encrypted secret key to homomorphically generate an encrypted pseudorandom keystream, which is data-independent and can be computed in advance. In the online phase, the server combines the precomputed keystream with the symmetric ciphertext, removing the mask and converting it into an FHE ciphertext of the original plaintext. By offloading intensive

tasks to the offline phase, this strategy enhances the online process, making it ideal for low-latency scenarios.

3 Survey Methodology

Rather than listing individual works, we categorize related symmetric ciphers and transciphering methods based on their inherent properties, enabling a clearer comparison of their design principles and efficiency.

3.1 Classification of Symmetric Ciphers

The message format of a symmetric cipher plays a critical role in ensuring seamless integration with FHE schemes. Different FHE schemes are optimized for specific types of operations. For example, schemes like CGGI are designed to efficiently handle binary (bit-wise) operations, whereas schemes such as BGV and BFV are better suited for operations on larger word sizes modulo p.

Given these considerations, we classify existing symmetric ciphers employed in transciphering into four distinct groups based on their message format and compatibility with different FHE schemes. This categorization helps in selecting the most appropriate cipher for a given application and FHE scheme.

1) FHE-friendly $\mathbb{Z}_2$ Ciphers: This category includes ciphers such as Kreyvium, FiLIP-like stream ciphers, and LowMC, Rasta-like block ciphers. These ciphers are specifically designed to minimize the number of Boolean gates in their encryption/decryption functions, which are formulated as Boolean circuits. Their design makes them highly compatible with FHE schemes that rely on efficient evaluation of Boolean operations.

2) $\mathbb{F}_p$ Ciphers: This group consists of ciphers like Masta, Pasta, HERA, and Rubato, where encryption and decryption functions are defined over arithmetic modulo a prime number p. These ciphers are well-suited for FHE schemes that operate on modular arithmetic, such as BGV or BFV, which can process message in $\mathbb{F}_p$ efficiently.

3) Standard Ciphers: This group encompasses widely used and standardized symmetric ciphers, with AES serving as the most notable example. Most standard ciphers are $\mathbb{Z}_2$ ciphers, but their design does not account for FHE compatibility. Consequently, their encryption and decryption functions typically involve a higher number of gates compared to FHE-friendly $\mathbb{Z}_2$ ciphers in Group 1. Despite this, their widespread adoption and cryptographic robustness make them an important consideration for transciphering.

4) $\mathbb{F}_{2^n}$ Ciphers: This category includes ciphers whose operations and plaintexts are defined over the finite field $\mathbb{F}_{2^n}$, such as Chaghri and Yu$_2$X. AES can also be interpreted over $\mathbb{F}_{2^n}$ [GHS12]. However, an important limitation is that transciphering over $\mathbb{F}_{2^n}$ can only be used for FHE computations that operate on $\mathbb{F}_{2^n}$. Most practical applications operate on numeric or Boolean values, and currently, there is no efficient method to convert elements from $\mathbb{F}_{2^n}$ into these formats under FHE. While a few studies have explored $\mathbb{F}_{2^n}$-based applications, such as GHASH [MV04], their practicality remains limited. Since this Systematization of Knowledge (SoK) focuses on transciphering for practical FHE tasks, we choose not to dedicate significant attention to transciphering methods over $\mathbb{F}_{2^n}$.

3.2 Classification of Transciphering Methods

We categorize existing transciphering methods into the following four groups, of which only the latter three are deemed sufficiently efficient.

Old methods. In the early days of FHE, homomorphically evaluating the AES circuit was often considered as a "benchmark" for assessing FHE performance. However, compared to

the more recent transciphering methods, these early efforts are consistently outperformed and thus less practical for current applications.

- **Transciphering to BGV scheme:** BGV support extended SIMD operations on vectors in $\mathbb{F}_{2^n}$. This allows $\mathbb{F}_{2^n}$ or $\mathbb{Z}_2$ ciphers[5] to use BGV for SIMD transciphering [GHS12, AMT22]. As mentioned in Section 3.1, due to the infeasibility of $\mathbb{F}_{2^n}$ for subsequent computations, We do not delve extensively into methods along this line in our SoK.

- **Transciphering to Vanilla FHE schemes:** Cheon et al. [CCK+13] and Coron et al. [CLT14] tested the AES evaluation based on the vDGHV [VDGHV10] scheme, while Doröz et al. [DHS16] evaluated the AES circuit using the LTV [LATV12] scheme. These FHE schemes themselves are usually not efficient enough to complete useful tasks.

CGGI-based methods. The CGGI-based scheme supports Boolean plaintext domains by representing bits within a specific interval of the torus, enabling the evaluation of any binary gate via a single bootstrapping operation, referred to as gate bootstrapping. From this perspective, CGGI is particularly well-suited for symmetric ciphers whose functionality can be represented as binary circuits.

In addition to gate bootstrapping (GBS), CGGI supports programmable bootstrapping (PBS) and circuit bootstrapping (CBS), enabling the evaluation of arbitrary functions in FHE and LHE modes, respectively. The computational cost ranking is as follows: GBS $<$ PBS $<$ CBS. GBS is lighter than PBS due to smaller parameters, while CBS, which uses PBS as a sub-procedure, is roughly ten times more expensive than PBS. All these bootstrapping methods rely on the CMUX gate as the fundamental operation.

The ciphertext formats in CGGI-based schemes include three types: LWE, RLWE and GGSW. The first two ciphertext types can be viewed as specific instances of GLWE ciphertext $c \in R_{q,N}^{k+1}$, where LWE corresponds to the case when $N = 1$ and RLWE corresponds to $k = 1$. For GGSW ciphertext, $C \in R_{q,N}^{\ell(k+1)\times(k+1)}$, where ℓ is the length of the decomposition, it essentially consists of multiple GLWE ciphertexts, which encrypts the secret key of the LWE to serve as bootstrapping key. The homomorphic multiplication GGSW $\times$ GLWE $\rightarrow$ GLWE is the basic unit of blind rotation. Typically, the size of a GGSW ciphertext is several times larger than a single GLWE ciphertext. Efficient conversion between these ciphertext types is supported by CGGI-based schemes, further enhancing the usability of the framework. For more details, refer to [CGGI20].

These capabilities make CGGI a powerful tool for evaluating symmetric ciphers, especially those involving S-boxes, such as AES [TCBS23, BPR24, WWL+23, WLW+24].

While CGGI-based methods typically offer lower latency, they tend to have lower throughput when compared to other FHE schemes, primarily due to the absence of SIMD support in CGGI. This trade-off arises from the inherent design choices prioritizing fast execution over parallelism.

Real-to-Finite-field (RtF) framework. The concept of the Real-to-Finite-field (RtF) framework [CHK+21] involves designing a symmetric cipher where the ciphertext domain is modulo p, matching the plaintext domain of BFV, thereby facilitating an efficient transciphering process into BFV ciphertext. Recognizing CKKS's exceptional performance in privacy-preserving applications, the RtF framework further proposes converting the BFV ciphertext into CKKS ciphertext, enabling efficient and secure computation over real numbers under encryption.

[5] A $\mathbb{Z}_2$ element can be viewed as an $\mathbb{F}_{2^n}$ element utilizing only one coefficient.

Binary-to-Real (BtR) framework. Symmetric ciphers usually have an avalanche effect, that is a single-bit error will lead to totally different decryption results. Given CKKS's inherent approximation nature, directly transciphering symmetric ciphertext into CKKS ciphertext is challenging. Fortunately, as proposed in the BLEACH framework [DMPS24], CKKS can be used to encrypt binary plaintext by representing a bit $x \in \{0, 1\}$ as a real number $x + \epsilon$ for some small noise $|\epsilon| \ll 1$. This allows for implementing binary gates such as AND ($x \wedge y$ as $x \cdot y$) and XOR ($x \bigoplus y$ as $(x - y)^2$) through arithmetic circuits. A recent work XBOOT [NHY+25] further optimizes XOR evaluations by using addition with lazy reduction instead of costly multiplications. The growth of the error term ϵ during these operations could be addressed by employing step functions [CKK20] or bootstrapping. These approaches enable efficient CKKS transciphering when the symmetric cipher can be modeled as a binary circuit. We denote such transciphering method as Binary-to-Real (BtR).

4 Symmetric Ciphers Used in Transciphering

In this section, we analyze the proposed symmetric ciphers used in transciphering by groups. We describe the structure and security of these ciphers, and discuss their suitability for transciphering. An overview is given in Table 1.

Table 1: Overview of symmetric ciphers *vs* transciphering.

Transciphering method	Message format	FHE-friendly $\mathbb{Z}_2$ ciphers	AES	$\mathbb{F}_p$ ciphers
CGGI-based	Binary	●	◐	○
RtF	Numeric	○	○	●
BtR	Binary	○	●	○

(○), (◐), and (●) denote the compatibility level of the ciphers for each transciphering method, ranging from least compatible to compatible (○ < ◐ < ●).

4.1 FHE-friendly $\mathbb{Z}_2$ Ciphers

This section includes some newly designed FHE-friendly symmetric ciphers over $\mathbb{Z}_2$.

4.1.1 Stream Ciphers

Kreyvium. In a recent exploration, the standardized cipher Trivium [RB08] and its variant Kreyvium [CCF+16] have been considered as candidates for transciphering [BOS23]. Trivium is one of the seven stream ciphers recommended by the eSTREAM project.[6] It features a small number of nonlinear operations in the warm-up phase[7]. However, Trivium offers only an 80-bit security level, which is considered marginal in current standards. Kreyvium, designed as a variant of Trivium, elevates this security to 128 bits with similar AND depth and FHE performance.

These NLFSR[8]-based stream ciphers typically require fewer Boolean gates compared to block ciphers. This efficiency is due to the fact that only a small number of Boolean gates are employed at each step, making them particularly well-suited for transciphering with the CGGI method. Additionally, after the warm-up phase, these ciphers can generate

[6]The eSTREAM project [DLSK05] is a competition run via a European project, between 2004 and 2008, to identify new stream ciphers.

[7]To produce k bits of output, one has to run the update function $1152 + k$ times as it requires to discard the first 1152 bits of output.

[8]A Non-Linear Feedback Shift Register (NLFSR)-based stream cipher is a type of stream cipher that utilizes a non-linear feedback shift register as its core component to generate a pseudorandom keystream.

a single keystream bit with a low cost. This characteristic leads to improved throughput in homomorphic evaluations, making them advantageous in FHE.

The Filter Permutator (FP) ciphers. Filter Permutators (FPs) were introduced by Méaux et al. at Eurocrypt 2016 [MJSC16], alongside the FLIP stream cipher. However, FPs were later found to have significant vulnerabilities [DLR16]. To address these issues, Improved Filter Permutators (IFPs) and the FiLIP cipher were proposed [MCJS19], incorporating lightweight subset selection and whitening steps that are compatible with CGGI-based FHE schemes.

Limitations. The early versions of the FP cipher family exhibited poor performance due to the numerous gate bootstrapping operations required to generate a single bit, with each gate bootstrapping containing several hundred CMUX gates.

Improved throughput. Subsequent research built on multi-value PBS, which enables the operation of arbitrary functions on multiple bits during a single PBS execution. This approach led to the development of Elisabeth-4, which defines the plaintext domain in $\mathbb{Z}_{16}$ and improves throughput by outputting 4 bits per evaluation. However, Elisabeth-4 was later broken by an algebraic attack [GBJR23]. To address these vulnerabilities, Hoffmann et al. [HMS23] introduced patched designs, including Elisabeth-b, Gabriel, and Margrethe, which aim to enhance security while preserving the efficiency of FHE. However, these ciphers incur at least double the TFHE evaluation cost compared to Elisabeth. They still suffer from the expensive cost of PBS.

Bootstrapping-free. A notable advancement came with the introduction of FiLIP-144 [HMR20, CDPP22], which incorporates a newly proposed filtering function called XOR-Threshold. This function eliminates the need for gate bootstrapping, replacing it with much cheaper CMUX gates. Typically, FiLIP-144 can generate one bit of keystream at a cost of less than one gate bootstrapping.

Combined optimization. By combining a larger plaintext domain ($\mathbb{Z}_{16}$) with the bootstrapping-free technique, an optimized implementation of Margrethe was introduced by [AGHM24]. In their evaluation, only inexpensive CMUX gates and homomorphic addition were utilized, resulting in excellent performance.

4.1.2 Block Ciphers

LowMC and Rasta family. LowMC, proposed at Eurocrypt 2015, minimizes multiplicative complexity by using randomly generated matrices in the linear layer, reducing AND depth and improving transciphering performance [ARS+15]. However, various attacks have prompted updates, resulting in the current version, LowMC_{v3} [DEM16, DLMW15, RST18].

The Rasta family [DEG+18] uses a quadratic nonlinear layer and randomized linear matrices to improve throughput but incurs higher ANDs per bit. While Dasta [HL20] reduces randomness by using bit-permutations, it doesn't improve homomorphic performance. Fasta [CIR22] optimizes for homomorphic evaluation in HElib but is limited by its fixed parameters.

Limitations. LowMC and Rasta were designed to minimize multiplication depth and implemented on the BGV scheme where XOR operations were considered "free". To reduce AND depth without compromising security, these $\mathbb{Z}_2$ block ciphers introduced many gates at each level. However, when evaluated with CGGI-based methods, the XOR gates are not free and incur expensive bootstrappings, causing significant transciphering latency and limiting performance.

FRAST. To address the limitations of LowMC and Rasta, FRAST [CCH+24] was proposed, which leverages the power of multi-value PBS with the plaintext domain defined in $\mathbb{Z}_{16}$. By using $\mathbb{Z}_{16}$, FRAST can process the 4-bit S-box with a single PBS evaluation, significantly reducing the number of PBS compared to earlier schemes.

A key advancement in FRAST is the use of GGSW-encrypted symmetric keys. Specifically, the CMUX gate is applied, which takes the GGSW-encrypted symmetric key as input and processes the resulting GLWE ciphertext from the S-box PBS. This approach, referred to as double-blind rotation, efficiently integrates the symmetric key with several inexpensive CMUX gates, thus reducing computational overhead and improving overall performance.

4.1.3 Summary of FHE-friendly $\mathbb{Z}_2$ Ciphers.

Security. FHE-friendly block ciphers mitigate statistical attacks by incorporating randomized elements during each encryption, such as randomized linear matrices in LowMC and Rasta or randomized S-boxes in FRAST. Despite these safeguards, the limited multiplicative depth inherent to FHE-compatible encryption systems produces low-degree equations for secret key recovery, rendering them susceptible to algebraic attacks [LSMI21].

Traditional stream ciphers, such as Kreyvium and Trivium, face state-of-the-art attacks called monomial prediction techniques [HST+21, HHLW24]. These ciphers benefit from extensive community-driven optimization efforts, with current security records for Trivium and Kreyvium remaining at 851 and 899 rounds, respectively [HHLW24].

Filter Permutator (FP) ciphers derive their security from the cryptographic strength of the underlying one-way function [Gol00, BY03]. Researchers should stay informed about recent advancements in bit-fixing correlation attacks [FLLL24], which have compromised certain FiLIP PRGs, particularly those incorporating the XOR-Threshold function.

In summary, while cryptanalysis methods for $\mathbb{Z}_2$ ciphers are well-developed, newly proposed techniques still pose significant security threats. Continued efforts from the research community are essential to bolster confidence in these ciphers and should remain a priority.

Transciphering efficiency. Most works combine FHE-SCs over $\mathbb{Z}_2$ with CGGI-based transciphering, aiming to reduce the number of bootstrapping operations during evaluation. This is primarily achieved through two techniques: 1) GGSW-based LUT: This technique uses GGSW-encrypted ciphertexts as input indices, selecting low-noise plaintext outputs with minimal use of cheap CMUX gates; 2) Leveraging multi-value PBS: By expanding the plaintext domain, multiple bits can be updated in a single PBS operation, thereby reducing the total number of expensive PBS operations.

A limitation of using GGSW-encrypted symmetric keys is the increased key size, as GGSW ciphertexts are several orders of magnitude larger than LWE ciphertexts. Nevertheless, these optimizations, when combined with the advanced features of CGGI schemes, result in significant improvements in overall performance.

Despite these significant improvements, CGGI-based methods still face low throughput issues due to the absence of SIMD support. The BtR framework, designed for SIMD evaluation of boolean circuits, could serve as a potential solution for efficiently transciphering large volumes of Z_2 ciphers.

4.2 $\mathbb{F}_p$ Ciphers for Word-Wise HE

The FHE schemes BGV, BFV, and CKKS efficiently handle arithmetic operations, like addition and multiplication, on multiple data simultaneously. The gap between boolean symmetric ciphers and FHE operations on prime numbers has inspired the design of FHE-SCs defined in $\mathbb{F}_p$.

Masta and Pasta. Masta [HKC+20], an $\mathbb{F}_p$ variant of the $\mathbb{Z}_2$ cipher Rasta, aims to improve throughput by replacing the bit-based structure of $\mathbb{Z}_2$ with larger primes in $\mathbb{F}_p$. The approach helps to scale better in terms of arithmetic efficiency. However, a key

challenge arises from the randomized matrices used during evaluation, which result in quadratically increasing plaintext-ciphertext multiplications. This quadratic complexity leads to a performance bottleneck, as the number of homomorphic plaintext-ciphertext multiplications grows significantly with larger block size, limiting the overall efficiency.

Pasta [DGH+23] addresses some of these issues by reducing the latency compared to Masta. This is achieved through a block-splitting strategy, which halves the matrix size from $2t$ to t. While this reduces the processing time for each individual block, the throughput doesn't improve significantly because the total output is also halved, thus limiting the performance gain.

Pasta_{v2} [GLR+24] improves upon Pasta by reducing the randomness required in the cipher. Only the first round of the matrix multiplication is randomized, while subsequent matrices are fixed. This reduces the complexity of random matrices generation and improves client-side performance. However, despite these improvements, the inherent inefficiency in the homomorphic evaluation remains, as the underlying issue of quadratic plaintext-ciphertext multiplication persists.

Limitations. A common limitation across Masta, Pasta, and Pasta_{v2} is the reliance on matrix multiplication in their linear layers to enhance diffusion and security. While this is effective for cryptographic strength, it introduces significant overhead in SIMD evaluations. Specifically, the quadratic increase in plaintext-ciphertext multiplications during such evaluations leads to a bottleneck in performance.

Randomized key schedule: HERA and Rubato. HERA [CHK+21], proposed alongside the RtF framework, first transciphers data into BFV ciphertext and then converts it into CKKS ciphertext. HERA enhances security through component-wise multiplication between a randomized vector and the symmetric key, reducing plaintext-ciphertext multiplication to linear growth. It also reduces the block size to $t = 16$ for better client-side performance, but this increases multiplication depth to 15 and consumes more server-side noise budget.

Rubato [HKL+22] reduces multiplication depth by introducing Gaussian noise after encryption, which lowers server-side costs but increases client-side overhead. This addition of Gaussian noise effectively combines the hardness assumption of Learning With Errors (LWE) with the security of symmetric ciphers, significantly enhancing the overall security of the system.

Limitations. Notably, an algebraic attack on HERA, which exploits multiple collisions in round keys, was proposed in [LKSM24]. This attack can peel off the last non-linear layer of the cipher, allowing a full-round attack on its 192/256-bit security claim. While the 128-bit security claim remains unaffected, this highlights the inherent trade-off between security and performance. For Rubato, a potential full-round attack [GMAH+23] exists when p is non-prime. Even if Rubato assumes p to be prime, its security still requires further investigation.

Another limitation of Rubato is its reliance on the CKKS scheme, which leads to a ciphertext expansion ratio (CER) greater than 1:1, typically around 1:1.26. Specifically, an additional scaling factor δ is required to convert floating-point data into integers for transciphering in the BFV scheme, resulting in a CER of p/δ. This expansion arises from the accuracy rounding similar to the CKKS scheme when dealing with floating-point values. While this design improves security and server-side performance, it limits the input to a workload that is less sensitive to small deviations.

4.2.1 Security of $\mathbb{F}_p$ Ciphers.

Although ciphers based on $\mathbb{F}_p$ are efficient for encryption, their security is not as well understood as that of $\mathbb{Z}_2$ ciphers. Building on the core concept of Rasta, FHE-SCs in $\mathbb{F}_p$ introduces randomness into the update function to mitigate chosen-plaintext attacks [BS91,

Knu95, BBS99, DS09], meaning that distinct encryption functions are applied to each quire. However, the primary threat to $\mathbb{F}_p$ ciphers arises from algebraic attacks, in which an adversary treats the secret key as an unknown variable and constructs multiple equations to solve for it.

Cryptanalysts employ linearization attacks [DEG+18], Gröbner basis attacks [BPW06], and interpolation attacks [JK97], all of which involve generating equations from the ciphertext. These equations typically take the form of polynomials with high-degree monomials. Symmetric ciphers counteract such attacks by increasing the polynomial density, either by adding more monomials or raising the degree of the terms. To illustrate this, we focus on linearization attacks to demonstrate how the parameters of $\mathbb{F}_p$ ciphers influence security.

How the parameters of $\mathbb{F}_p$ ciphers impact security. A linearization attack consists of two primary phases: the equation-building phase (where equations are collected from the output) and the equation-solving phase, which determines the overall time complexity. The time complexity is estimated as $\mathcal{O}\left(\binom{t+d}{d}^{\omega}\right)$ field operations, where t is the number of variables, d is the degree of each equation, and $2 \leq \omega < 3$ represents the Gaussian elimination constant.

If the time complexity of attacking an $\mathbb{F}_p$ cipher, given specific parameters, is lower than the claimed security level, the cipher is considered compromised. Typically, the security level is set to 128 bits. Hybrid approaches can reduce the number of variables by guessing some of them before building the equations, thus lowering the complexity. These attacks have a lower-bounded time complexity of $\mathcal{O}(p^j\binom{t-j+d}{d}^{\omega})$ when j primes are guessed.

Optimization can be achieved by directly constructing low-degree equations, significantly reducing time complexity based on binomial combinations. This is often accomplished by peeling off the first or last non-linear layer. However, such attacks involve complex procedures and additional costs, such as guessing parts of the output block or creating key schedule collisions [GLR+24, LKSM24]. Countermeasures like truncation in Pasta and Rubato, or key whitening—where the master key is XORed into the block at the end of the encryption process—help mitigate these risks in ciphers like Masta and HERA.

In summary, increasing the block size t and the algebraic degree of the cipher's update function enhances security, though at the cost of performance. The prime size p of each element in the block bolsters security by raising the complexity of guessing each element, but it slightly impacts server-side performance due to greater noise budget consumption during transciphering. The design of $\mathbb{F}_p$ ciphers is still evolving, and the interaction between cryptanalysis techniques and countermeasures adds complexity to both the design process and the selection of optimal parameters. Ultimately, $\mathbb{F}_p$ ciphers require further validation to establish confidence in their security.

4.3 AES and Related Transciphering Works

While AES is generally not considered FHE-friendly due to its exceptionally high multiplication depth, it remains a top choice for transciphering applications because of its high encryption efficiency and widespread adoption.

AES with BGV. In 2012, [GHS12] designed AES-BGV transciphering. Since the BGV scheme supports SIMD and AES can be treated within $\mathbb{F}_{2^n}$, which is highly compatible with BGV, [GHS12] achieved a remarkable throughput of two seconds per AES block. However, as discussed in previous sections, elements in $\mathbb{F}_{2^n}$ are infeasible for subsequent FHE computations. As a result, AES-BGV transciphering is primarily considered an FHE benchmark rather than a practical solution for transciphering in real-world workloads.

AES with CGGI. CGGI supports efficient gate bootstrapping, which can be used to evaluate AES by replacing all the Boolean gates in the AES circuit. However, this approach leads to extremely high computational latency [MG21]. Recently, AES evaluation strategies based on PBS and CBS computation modes have been developed, offering significantly lower latency.

- **PBS-based evaluation:** Homomorphic computation of AES based on programmable bootstrapping technique has been continuously proposed and optimized [SMK22, TCBS23, BPR24]. Trama et al. [TCBS23] adapted four basic functions of AES to lookup table operations in order to be compatible with PBS. In particular, they analyzed the effect of different message spaces on the efficiency of PBS to determine the optimal parameters for homomorphic computation of AES.

- **CBS-based evaluation:** Wei et al. [WWL+23] proposed the first efficient AES homomorphic computation method based on circuit bootstrapping mode. Specifically, they present a cheap Sbox leveled lookup table using CMux gates combined with hybrid packing technique. Moreover, the circuit bootstrapping is used to implement ciphertext conversion to complete the whole AES full-round evaluation. Subsequent works [WLW+24, WWL+24] further optimize the computational cost of circuit bootstrapping, thus further improving the evaluation latency of AES.

AES with BtR. The BtR framework enhances Boolean operations and performance in CKKS. IBM Research [ADE+23] introduced a transciphering method using BtR for the homomorphic AES evaluation. They fully leveraged SIMD to process thousands of blocks in parallel, boosting throughput.

While AES was inefficient for bit-wise transciphering like CGGI, its linear layers (rotations and XORs) are well-suited for BtR. Each block bit maps to a ciphertext, making bit permutation free through index changes. This efficiency arises from SIMD-friendly, bitsliced implementations [MN07, Kön08, KS09], enabling parallel processing and high throughput in BtR.

4.4 Authenticated Encryption in Transciphering

Authenticated Encryption (AE) such as AES-GCM and ASCON not only protects data confidentiality but also ensures integrity, and has been widely adopted in symmetric encryption. [BBS21] proposed the idea of transciphering on authenticated encrypted ciphertext for the first time. [BPR24] implemented ASCON transciphering in CGGI. [ADE+23] implemented transciphering for AES-GCM and ASCON using BtR.

In traditional AE, the decryptor uses the symmetric key to verify the ciphertext integrity. However in transciphering, the server has no direct access to the symmetric key. [ADE+23] suggests letting the server perform FHE computations on the transciphered data, regardless of whether the data has been compromised. Then, the server transmits both the result and an encrypted bit indicating the validity of the ciphertext to the client. The client will only accept the result if this decrypted bit is true. This would waste computational resources if the symmetric ciphertext were compromised. An alternative strategy is sending verification data to the client and waiting for a success signal before proceeding, which increases communication rounds and is less optimal in the FHE scenario, especially for the resource-limited client. On the contrary, simply having the client sign the symmetric ciphertext achieves similar goal as AE, but avoids all these drawbacks.

We suggest that the necessity of AE in the context of transciphering is still in doubt, and requires further examination.

5 Experiments

In this section, we conduct three sets of experiments. First, we evaluate the efficiency of various FHE-SCs when paired with different transciphering methods. Next, we design specific FHE applications and pair them with various transciphering solutions to assess their practical applicability. Lastly, we measure the encryption cost of different FHE-SCs from the client's perspective.

The experiments were conducted on a machine with Intel Xeon CPU operating at 2.70 GHz and 256 GB of RAM. All the numbers were obtained using a single thread[9].

All the symmetric ciphers we used satisfy $\geq$ 128-bit security as evaluated in their corresponding papers. Early versions of many FHE-SCs (e.g., FLIP, Elisabeth-4, Chaghri, LowMC, AgRasta) are broken [DLR16, LAW+23, LIM21, GBJR23, LSMI21] and excluded from the experiments. Transciphering to BGV is not tested due to the reasons mentioned in Section 3.2.

5.1 Parameters Descriptions

All word-wise FHE-based transciphering (RtF and CKKS) used the same modulus chain and polynomial degree for consistency, with $\log_2(QP) \leq 1555$ bits, ensuring at least 128-bit security level for BFV and CKKS. We then remark the parameters of the symmetric ciphers as follows:

- **Stream Ciphers:** We benchmark the stream ciphers by measuring latency until the 128-bit keystream is output. For throughput, the warm-up costs of Trivium and Kreyvium are excluded. FiLIP-(DSM/TX)[10] and Margrethe with (k, m) denote the key size and the input size of the nonlinear filtering function.
- **Block Ciphers:** LowMC, Rasta and AES with parameters (r, t), denote the number of rounds and block size, respectively.
- $\mathbb{F}_p$ **ciphers** with parameters (r, t, m, p) denote the number of rounds, the block size, the output size, and the plaintext's bit-size, respectively. The cipher block is measured in units of $\mathbb{F}_p$.

5.2 Transciphering Benchmarks

We list the latency and throughput of transciphering different symmetric ciphers with corresponding methods in Table 2 and illustrated in Fig. 2. Two popular benchmark criteria are reported: **latency** (the amount of time required before outputting the first block of transciphering results, the lower the better.) and **throughput** (the amortized amount of symmetric ciphertext transformed per second, the higher the better). We include both online and offline costs but exclude the one-time setup cost.

Benchmark for $\mathbb{Z}_2$ ciphers with CGGI. As shown in Fig. 2 (left) and Table 2, $\mathbb{Z}_2$ block ciphers LowMC, Dasta and Rasta suffer from poor performance in CGGI-based transciphering, characterized by high latency and low throughput. This is primarily due to their use of multiple XOR gates.

[9] For multi-threading, CGGI-based schemes generally show better scalability because of its smaller memory footprint (e.g., we observe a 30x speedup for Kreyvium-CGGI transciphering using 64 threads, versus 20x for AES-CKKS). We opt to report single-threaded performance due to varying multi-threading support across different FHE libraries.

[10] Direct Sum Monomial (DSM) [MJSC16] and XOR-Threshold [HMR20] type of filtering functions with different input size, respectively.

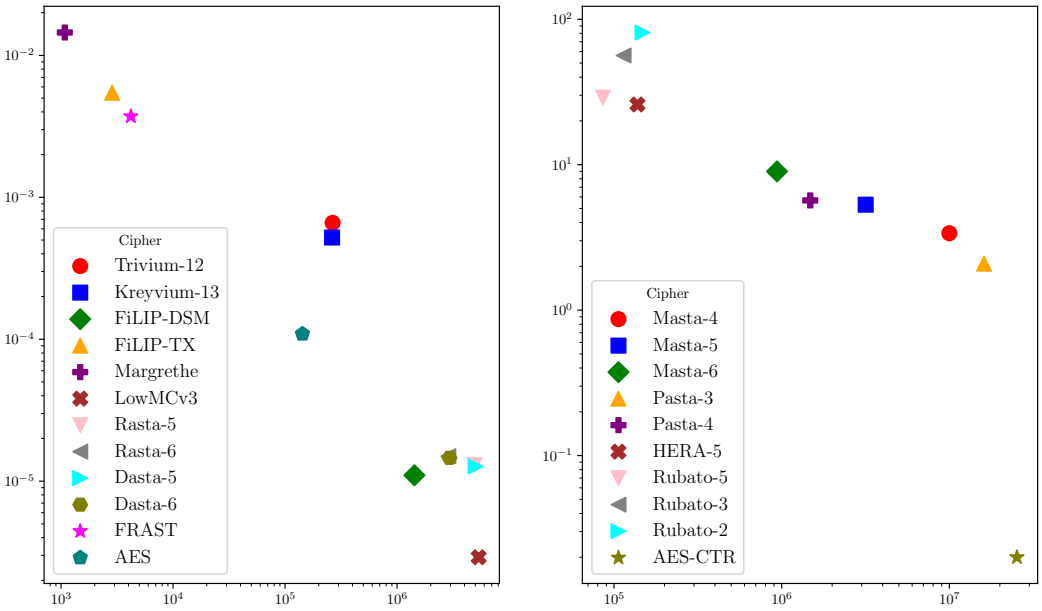

Figure 2: Throughput (KB/s in Vertical axis) and Latency (ms in Horizontal axis) of various FHE-SCs on server-side.

Methods like FiLIP-TX, FRAST, and Margrethe demonstrate good performance, benefiting from optimized transciphering which reduces the need for bootstrapping. Interestingly, AES is not the worst one. It performs moderately in both latency and throughput due to a special design of CBS-based method.

The best performer in this group, Margrethe, makes efficient use of the GGSW-based LUT, as described in Sect. 4.1.1. This results in significantly lower computational costs, thanks to the combined optimization of a larger plaintext domain, $\mathbb{Z}_{16}$, and bootstrapping-free strategy.

However, the improved performance of FiLIP-TX, Margrethe, and FRAST all comes at the cost of a large key size, formatted as GGSW ciphertexts. This aspect will be discussed in the **Key size** section.

Benchmark for ciphers with RtF and BtR. As shown in Fig. 2 (right) and Table 2, Masta and Pasta are less efficient than HERA and Rubato due to the numerous $\mathsf{pt} \times \mathsf{ct}$ homomorphic operations required by Masta and Pasta's random matrix multiplication. The number of $\mathsf{pt} \times \mathsf{ct}$ operations is quadratically related to their block size, implying that variants with larger block sizes perform even worse.

In contrast, HERA and Rubato avoid using random matrices in their round functions. They instead introduce randomness into the key schedule via component-wise multiplication between a randomized vector and the symmetric key, leading to a linear increase in $\mathsf{pt} \times \mathsf{ct}$ operations.

Rubato outperforms HERA by reducing the multiplicative depth. This improvement is achieved without compromising security, by introducing Gaussian noise at the final stage of the symmetric encryption process. This can be viewed as adding an extra layer of security based on the LWE hardness problem to the encryption scheme. And versions of Rubato with fewer rounds demonstrate better performance.

For AES transciphering combined with BtR, the SIMD features of the CKKS scheme are leveraged, resulting in better throughput compared to the CGGI-based method. However,

Table 2: Throughput and latency of various FHE-SCs were tested in a single-threaded environment. Throughput is measured in kilobytes of messages transciphered per second. $\mathbb{F}_p$ ciphers are benchmarked in a leveled homomorphic encryption (LHE) mode, where the BFV-to-CKKS conversion is excluded.

Cipher(Parameters)	Schemes	Type	Evaluation	Latency (ms)	Throughput (KB/s)	Key Size (MB)
Trivium-12		$\mathbb{Z}_2$	PBS[BOS23]	2.66×10^5	6.61×10^{-4}	10.3
Kreyvium-13		$\mathbb{Z}_2$	PBS[BOS23]	2.63×10^5	5.21×10^{-4}	10.7
FiLIP-DSM(4096,1280)	CGGI	$\mathbb{Z}_2$	GBS[DGH+23]	1.43×10^6	1.10×10^{-5}	41.8
FiLIP-TX(16384,144)		$\mathbb{Z}_2$	CMUX[MPP24]	2.86×10^3	5.46×10^{-3}	215
Margrethe(2048,308)		$\mathbb{Z}_{16}$	CMUX[AGHM24]	1.08×10^3	1.45×10^{-2}	128
LowMCv3(14,196)		$\mathbb{Z}_2$	GBS[DGH+23]	5.36×10^6	2.91×10^{-6}	11.2
Rasta(5,525)		$\mathbb{Z}_2$	GBS[DGH+23]	4.94×10^6	1.30×10^{-5}	13.8
Rasta(6,351)		$\mathbb{Z}_2$	GBS[DGH+23]	2.87×10^6	1.50×10^{-5}	12.4
Dasta(5,525)	CGGI	$\mathbb{Z}_2$	GBS[DGH+23]	5.03×10^6	1.27×10^{-5}	13.8
Dasta(6,351)		$\mathbb{Z}_2$	GBS[DGH+23]	2.94×10^6	1.46×10^{-5}	12.4
FRAST(40,128)		$\mathbb{Z}_{16}$	PBS[CCH+24]	4.20×10^3	3.72×10^{-3}	11.8/460‡
AES(10,128)		$\mathbb{Z}_2$	CBS[WWL+23]	1.43×10^5	1.09×10^{-4}	1500
Masta(4,128,128,33-bit)		$\mathbb{F}_p$	LHE	1.00×10^7	3.38	5051
Masta(5,64,64,33-bit)		$\mathbb{F}_p$	LHE	3.17×10^6	5.31	3451
Masta(6,32,32,33-bit)†		$\mathbb{F}_p$	LHE	9.38×10^5	8.99	2651
Pasta(3,256,128,33-bit)		$\mathbb{F}_p$	LHE	1.61×10^7	2.09	8251
Pasta(4,64,32,33-bit)	BFV	$\mathbb{F}_p$	LHE	1.48×10^6	5.68	3451
HERA(5,16,16,28-bit)		$\mathbb{F}_p$	LHE[CHK+21]	1.38×10^5	25.95	2251
Rubato(5,16,12,26-bit)		$\mathbb{F}_p$	LHE[HKL+22]	0.86×10^5	28.94	2251
Rubato(3,36,32,25-bit)		$\mathbb{F}_p$	LHE[HKL+22]	1.14×10^5	56.33	2751
Rubato(2,64,60,25-bit)		$\mathbb{F}_p$	LHE[HKL+22]	1.48×10^5	81.07	3451
AES(10,128)	BGV	$\mathbb{Z}_2$	LHE[GHS12]	1.98×10^5	9.5×10^{-3}	10.5
AES-CTR(10,128)††	CKKS	$\mathbb{Z}_2$	FHE	2.53×10^7	0.02	3387

(†): The custom cipher version to clarify the effect of the matrix scale.
(‡): The symmetric key is first compressed into GLWE format, and then converted from GLWE to GGSW on the server. The terms GLWE/GGSW denote the corresponding key sizes before and after the conversion, respectively.
(††): We implemented the CKKS transciphering from [ADE+23] in Lattigo, as the original code was not available.

in BtR, each slot encrypts only a single bit, making it less efficient than RtF, which encrypts a value over $\mathbb{F}_p$ per slot.

Key size. The encrypted symmetric key and evaluation key are transferred from the client to the server during a one-time setup. To improve efficiency, several methods include optimizations that increase the transciphering efficiency at the cost of increased key size. To facilitate a more accurate apples-to-apples comparison, we also include a rough estimation of the total size of the encrypted symmetric and evaluation keys as part of the benchmark.

A fresh LWE ciphertext, with no homomorphic operations applied, can compress its random values over $\mathbb{Z}_q^N$ into a seed, reducing the LWE sample size to $\log q$ (the modulus size). Other ciphertext formats based on CGGI can also benefit from this technique. In this work, we account for the key size reduction enabled by this optimization. The symmetric keys are encrypted in multiple formats across different schemes, resulting in different key sizes. We will briefly introduce these methods below.

- **LWE Symmetric Key:** The evaluation methods involve gate bootstrapping (GBS) and programmable bootstrapping (PBS), requiring the symmetric key to be encrypted

as an LWE sample. The size of this encryption can be estimated as $k \times \log q$, where k is the total number of key bits and $\log q$ is the LWE modulus size.

- **GGSW Symmetric Key:** The symmetric key is encrypted in the lifted form of GGSW, i.e., GGSW(X^k), which can be evaluated within a single CMUX gate. However, the storage required for one GGSW ciphertext is tens of thousands of times larger than that of an LWE ciphertext.
- **RLWE Symmetric Key:** In RtF and BtR, each ciphertext involves two polynomials with N coefficients. Suppose $N = 65,536$, the size of the ciphertext at a multiplication depth d is approximately 12 MB for AES-CTR when $d = 12$, and 25 MB for RtF benchmarks.

In the CBS-based AES implementation from [WWL+23], the increase in key size arises from a different reason. They also applies the GGSW-based CMUX tree LUT to eliminate the expensive S-box evaluation cost, outputting an LWE ciphertext. However, unlike the FP-based stream cipher, AES requires additional evaluation, which necessitates an expensive CBS to refresh the noise while converting the LWE ciphertext into GGSW format. As a result, the computation bottleneck shifts from S-box evaluation to CBS evaluation. Furthermore, the increase in key size also stems from the CBS evaluation key.

The optimized Margrethe implementation in [AGHM24] leverages the technique of first sending the GLWE-packed symmetric key, then transforming them from GLWE to GGSW using the method described in [CCR19]. This approach reduces the communication cost for transferring the symmetric key during the setup phase. To enable a more comprehensive comparison, we also include the key size of the expanded symmetric key in GGSW format, as the conversion time from GLWE to GGSW is not accounted for.

For the CKKS bootstrapping key, we estimate the key size using the Lattigo library, which is about 1851 MB.

5.3 Application Performances

In this section, we choose two widely used FHE applications as benchmarks to evaluate the performance of transciphering across different scenarios: the Allelic chi-square test in secure genome-wide association studies (GWAS) [BGPG20] and privacy-preserving convolutional neural network (CNN) inference [LLL+]. Both workloads are computationally intensive. We didn't choose smaller workloads because, in such cases, the end-to-end server-side cost would be dominated by the transciphering efficiency, which has already been listed in Table 2. Only ciphers compatible with SIMD FHE schemes are considered, as other FHE approaches would be orders of magnitude slower for these computational expensive tasks.

Allelic chi-square test. We perform Pearson's chi-square test for independence using the CKKS scheme on the transciphered result. The task involves calculating the sum of squared differences between observed and expected frequencies. The input consists of data from 200 individuals, each containing 16,384 Single-Nucleotide Polymorphisms (SNPs), with each SNP represented as $\{0, 1, 2\}$ using 2 bits of information. The total size of the plaintext data is approximately 800 KB.

Privacy-preserving CNN inference. We evaluate the performance of CNN inference for the ResNet20 model using the CKKS scheme on the transciphered CIFAR-10 dataset. CNNs involve complex computations with numerous linear convolution layers, as well as non-linear ReLU layers, which are approximated by high-degree polynomials. The input data consists of images with dimensions $32 \times 32 \times 3$, represented as 32-bit floating-point values, amounting to 12 KB per image. We implemented the CNN inference algorithm from [LLL+] in Lattigo.

Gap between application performance. The performance of transciphering along with applications are shown in Table 3. The HERA and Rubato $\mathbb{F}_p$ ciphers still achieve highest performance, but compared to the orders of magnitude advantage they show against AES in Table 2, their gaps here become less significant. This is due to workload limitations that cap the end-to-end performance.

Specifically, the 2-bit data format in chi-square test restricts the effective use of the slots. As a result, the capacity of each $\mathbb{F}_p$ slot for RtF is underutilized, encrypting only a 2-bit value. For secure CNN inference, the computational bottleneck lies in the numerous layers of ReLU functions, which can cost thousands of seconds. Finally, the expensive BFV-CKKS halfboot requirement [CHK+21] of $\mathbb{F}_p$ ciphers further reduce the advantage of $\mathbb{F}_p$ ciphers.

In summary, if the application workload is complex, the efficiency of transciphering becomes less important, and AES could still be an acceptable choice.

Table 3: End-to-end time cost of applications, running with single thread.

Cipher(Parameters)	Schemes	Chi-Square (s)	ResNet-20 (s)
Masta(4,128,128,33-bit)	BFV/CKKS	1.10×10^4	1.10×10^4
Masta(5,64,64,33-bit)		4.19×10^3	4.26×10^3
Masta(6,32,32,33-bit)		2.90×10^3	2.02×10^3
Pasta(3,256,128,33-bit)		1.71×10^4	1.72×10^4
Pasta(4,64,32,33-bit)		3.99×10^3	2.57×10^3
HERA(5,16,16,28-bit)	BFV/CKKS	1.57×10^3	1.22×10^3
Rubato(5,16,12,26-bit)		1.45×10^3	1.17×10^3
Rubato(3,36,32,25-bit)		1.25×10^3	1.20×10^3
Rubato(2,64,60,25-bit)		1.17×10^3	1.23×10^3
AES-CTR(10,128)	CKKS	5.06×10^4	2.63×10^4

5.4 Client Side Performances

The efficiency of symmetric encryption itself is crucial because clients might have limited computational resources. We re-ran the symmetric encryption code from each study in our environment. Pasta_{v2} is a variant of Pasta with similar transciphering performance but significantly improved client-side efficiency.

As shown in Table 4, AES offers the best client-side encryption performance. It's worth noting that the current AES implementation (AES-SW) does not utilize any platform-specific optimization such as the AES-NI (Advanced Encryption Standard New Instructions) acceleration, which will lead to even better performance if implemented.

$\mathbb{F}_p$ ciphers generally outperform $\mathbb{Z}_2$ ciphers because they require less calls to Pseudo-Random Number Generators (PRNGs). One PRNG call in $\mathbb{F}_p$ ciphers generates a word, whereas in $\mathbb{Z}_2$ ciphers, it produces only a single bit. Dasta, FRAST and Pasta_{v2} are designed with fewer random elements, which reduces the need for calls to PRNGs, making them more efficient than others in their respective groups.

Although Rubato is highly efficient for transciphering, its client-side performance is poor due to the inclusion of Gaussian noise. This noise addition can be seen as a combination of expensive LWE encryption and symmetric encryption.

6 Observations

Concluding from the results above, we obtain the following useful observations:

Table 4: Client side performance of various ciphers. Note that due to the high speed of client-side encryption, the measurements in this table differ from those in previous tables: Latency is measured in CPU cycles, and throughput (Thrp.) is measured in CPU cycles per byte (the lower the better).

Cipher (Parameters)	Type	Latency (Cycles)	Thrp. (C/B)
Trivium-12	$\mathbb{Z}_2$	4.89×10^4	3.02×10^3
Kreyvium-13	$\mathbb{Z}_2$	5.27×10^4	3.29×10^3
FiLIP-DSM(4096,1280)	$\mathbb{Z}_2$	2.56×10^7	1.60×10^6
FiLIP-TX(16384,144)	$\mathbb{Z}_2$	1.01×10^7	6.31×10^5
Margrethe(2048,308)	$\mathbb{Z}_{16}$	2.81×10^6	1.76×10^5
LowMC_{v3}(14,196)	$\mathbb{Z}_2$	1.73×10^8	7.05×10^6
Rasta(5,525)	$\mathbb{Z}_2$	1.81×10^8	2.75×10^6
Rasta(6,351)	$\mathbb{Z}_2$	4.54×10^7	1.03×10^6
Dasta(5,525)	$\mathbb{Z}_2$	1.70×10^6	2.59×10^4
Dasta(6,351)	$\mathbb{Z}_2$	7.76×10^5	1.77×10^4
FRAST(40,128)	$\mathbb{Z}_{16}$	1.47×10^5	0.92×10^4
Masta(4,128,128,33-bit)	$\mathbb{F}_p$	2.28×10^6	4.33×10^3
Masta(5,64,64,33-bit)	$\mathbb{F}_p$	7.25×10^5	2.75×10^3
Masta(6,32,32,33-bit)	$\mathbb{F}_p$	2.25×10^5	1.71×10^3
Pasta(3,128,64,33-bit)	$\mathbb{F}_p$	4.56×10^6	1.73×10^4
Pasta(4,64,32,33-bit)	$\mathbb{F}_p$	4.70×10^5	3.57×10^3
Pasta_{v2}(3,128,64,33-bit)	$\mathbb{F}_p$	2.16×10^6	8.20×10^3
Pasta_{v2}(4,64,32,33-bit)	$\mathbb{F}_p$	1.92×10^5	1.46×10^3
HERA(5,16,16,28-bit)	$\mathbb{F}_p$	2.86×10^4	5.11×10^2
Rubato(5,16,12,26-bit)	$\mathbb{F}_p$	1.22×10^7	3.14×10^5
Rubato(3,36,32,25-bit)	$\mathbb{F}_p$	1.37×10^7	1.36×10^5
Rubato(2,64,60,25-bit)	$\mathbb{F}_p$	1.48×10^7	7.89×10^5
AES-SW(10,128)	$\mathbb{Z}_2$	1.86×10^3	1.16×10^2

AES-SW is evaluated using the open-source implementation from GitHub (https://github.com/kokke/tiny-AES-c).

(1) Data shape is the key factor in transciphering. By data shape, we refer to the size and format of data required by FHE applications. For applications dealing with small data size, it's preferable to select a $\mathbb{Z}_2$ symmetric cipher in conjunction with CGGI transciphering. With larger data sets, $\mathbb{F}_p$ symmetric ciphers offer the best performance when matched with the RtF framework.

The boundary between "small" and "large" data size is roughly several kilobytes. As the data volume grows, one would need to run multiple instances of $\mathbb{Z}_2$ ciphers due to CGGI's lack of SIMD capabilities, potentially making $\mathbb{F}_p$ ciphers a more attractive option.

Additionally, a few $\mathbb{Z}_2$ symmetric ciphers can also achieve reasonable efficiency for larger data sets when paired with the BtR framework. The key distinction is that BtR supports both binary and arithmetic circuits, whereas RtF is more optimized for arithmetic circuits. For instance, if Hamming distance (which involves binary circuits) needs to be computed after transciphering, BtR might be a better choice.

(2) $\mathbb{F}_p$ ciphers excel in large-scale transciphering, but concerns persist regarding their security or client-side complexity. The primary issue is that $\mathbb{F}_p$ ciphers are relatively new, and their resilience against algebraic attacks has not been thoroughly explored. They are potentially more vulnerable for two reasons: they feature fewer unknown variables since these variables are counted on a word-wise basis, and their arithmetic operations might inadvertently introduce exploitable algebraic structures.

For efficient transciphering, the primary design strategy for $\mathbb{F}_p$ ciphers is to incorporate more operations that are inexpensive in a homomorphic context at the server side. However, these operations might not be friendly at the client side. For instance, Rubato employs Gaussian noise, but this adds additional computational load on the client. A client with constrained communication resources is often also limited in computational capabilities, such as an IoT device. Achieving a balance between efficiency, security, and resource usage is challenging and necessitates a comprehensive analysis of the cipher's structure and security considerations.

(3) New designs of $\mathbb{Z}_2$ ciphers and corresponding transciphering methods are required. CGGI serves as effective solutions for transciphering over $\mathbb{Z}_2$ ciphers. However, the existing FHE-friendly $\mathbb{Z}_2$ ciphers are predominantly designed to reduce the depth or number of AND gates, often neglecting the XOR gate, which has the same complexity[11] with AND gate in CGGI. As a result, these designs, such as LowMC, would be even less efficient than AES for transciphering. Therefore, new designs of $\mathbb{Z}_2$ ciphers that feature fewer XOR gates or utilize other enhanced transciphering methods would be advantageous.

(4) Standard symmetric cipher (AES) is not too bad for transciphering. Surprisingly, despite extensive efforts in designing new FHE-SCs, AES remains a viable option. It performs moderately in small-scale transciphering (refer to Figure 2) and is approximately one order of magnitude slower than the best-performing schemes on larger, more time-consuming tasks (refer to Table 3). Moreover, AES offers significantly superior performance on the client side (refer to Table 4). Given the well-established security of AES, users might prefer to continue using it. This situation challenges FHE-SC researchers: they must outperform AES by a considerable margin to alter the status quo, taking into account both client and server-side costs.

Acknowledgements

We would like to express our sincere gratitude to the anonymous reviewers for their valuable comments and insightful suggestions, which have significantly contributed to enhancing the quality of this paper.

This work is supported by Ant Group. Meiqin Wang is supported by the National Key R&D Program of China (Grant No. 2024YFA1013000, 2023YFA1009500), the National Natural Science Foundation of China (Grant No. 62032014, U2336207), Department of Science Technology of Shandong Province (No. SYS202201), Quan Cheng Laboratory (Grant No. QCLZD202301, QCLZD202306).

[11] A recent work [CLW+24] could reduce the cost of XOR gates in CGGI; however, they require more complex handling of AND gates.

References

[AABS+20] Abdelrahaman Aly, Tomer Ashur, Eli Ben-Sasson, Siemen Dhooghe, and Alan Szepieniec. Design of symmetric-key primitives for advanced cryptographic protocols. IACR Transactions on Symmetric Cryptology, pages 1–45, 2020.

[ADE+23] Ehud Aharoni, Nir Drucker, Gilad Ezov, Eyal Kushnir, Hayim Shaul, and Omri Soceanu. E2e near-standard and practical authenticated transciphering. Cryptology ePrint Archive, Paper 2023/1040, 2023. https://eprint.iacr.org/2023/1040.

[AGHM24] Diego F. Aranha, Antonio Guimarães, Clément Hoffmann, and Pierrick Méaux. Secure and efficient transciphering for fhe-based MPC. IACR Cryptol. ePrint Arch., page 1702, 2024.

[AGR+16] 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, pages 191–219. Springer, 2016.

[AMT22] Tomer Ashur, Mohammad Mahzoun, and Dilara Toprakhisar. Chaghri - A fhe-friendly block cipher. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, pages 139–150. ACM, 2022.

[ANM25] Hossein Abdinasibfar, Camille Nuoskala, and Antonis Michalas. The hhe land: Exploring the landscape of hybrid homomorphic encryption. Cryptology ePrint Archive, 2025.

[ARS+15] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In Elisabeth Oswald and Marc Fischlin, editors, Advances in Cryptology - EUROCRYPT 2015, volume 9056 of Lecture Notes in Computer Science, pages 430–454. Springer, 2015.

[BAB+22] Ahmad Al Badawi, Andreea Alexandru, Jack Bates, Flavio Bergamaschi, David Bruce Cousins, Saroja Erabelli, Nicholas Genise, Shai Halevi, Hamish Hunt, Andrey Kim, Yongwoo Lee, Zeyu Liu, Daniele Micciancio, Carlo Pascoe, Yuriy Polyakov, Ian Quah, Saraswathy R.V., Kurt Rohloff, Jonathan Saylor, Dmitriy Suponitsky, Matthew Triplett, Vinod Vaikuntanathan, and Vincent Zucca. Openfhe: Open-source fully homomorphic encryption library. Cryptology ePrint Archive, Paper 2022/915, 2022. https://eprint.iacr.org/2022/915.

[BBS99] Eli Biham, Alex Biryukov, and Adi Shamir. Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. In Advances in Cryptology—EUROCRYPT'99, pages 12–23. Springer, 1999.

[BBS21] Adda-Akram Bendoukha, Aymen Boudguiga, and Renaud Sirdey. Revisiting stream-cipher-based homomorphic transciphering in the TFHE era. In Foundations and Practice of Security - 14th International Symposium, FPS 2021,, volume 13291 of Lecture Notes in Computer Science, pages 19–33. Springer, 2021.

[BGGJ20] Christina Boura, Nicolas Gama, Mariya Georgieva, and Dimitar Jetchev. Chimera: Combining ring-lwe-based fully homomorphic encryption schemes. Journal of Mathematical Cryptology, 14(1):316–338, 2020.

[BGK+21] Mario Barbara, Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, Markus Schofnegger, Roman Walch, and T Graz. Reinforced concrete: Fast hash function for zero knowledge proofs and verifiable computation. IACR Cryptol. ePrint Arch., 2021:1038, 2021.

[BGPG20] Marcelo Blatt, Alexander Gusev, Yuriy Polyakov, and Shafi Goldwasser. Secure large-scale genome-wide association studies using homomorphic encryption. Proceedings of the National Academy of Sciences, 117(21):11608–11613, 2020.

[BGV14] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT), 6(3):1–36, 2014.

[BMTPH21] Jean-Philippe Bossuat, Christian Mouchet, Juan Troncoso-Pastoriza, and Jean-Pierre Hubaux. Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pages 587–617. Springer, 2021.

[BOS23] Thibault Balenbois, Jean-Baptiste Orfila, and Nigel P. Smart. Trivial transciphering with trivium and TFHE. In Proceedings of the 11th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, pages 69–78. ACM, 2023.

[BPA+23] Amit Singh Bhati, Erik Pohle, Aysajan Abidin, Elena Andreeva, and Bart Preneel. Let's go eevee! a friendly and suitable family of aead modes for iot-to-cloud secure computation. In Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, pages 2546–2560, 2023.

[BPR24] Nicolas Bon, David Pointcheval, and Matthieu Rivain. Optimized homomorphic evaluation of boolean functions. IACR Trans. Cryptogr. Hardw. Embed. Syst., 2024(3):302–341, 2024.

[BPW06] Johannes Buchmann, Andrei Pyshkin, and Ralf-Philipp Weinmann. Block ciphers sensitive to gröbner basis attacks. In Topics in Cryptology–CT-RSA 2006: The Cryptographers' Track at the RSA Conference 2006, pages 313–331. Springer, 2006.

[Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In Annual Cryptology Conference, pages 868–886. Springer, 2012.

[BS91] Eli Biham and Adi Shamir. Differential cryptanalysis of des-like cryptosystems. Journal of CRYPTOLOGY, 4:3–72, 1991.

[BY03] Mihir Bellare and Bennet S. Yee. Forward-security in private-key cryptography. In Marc Joye, editor, Topics in Cryptology - CT-RSA 2003, The Cryptographers' Track at the RSA Conference 2003, San Francisco, CA, USA, April 13-17, 2003, Proceedings, volume 2612 of Lecture Notes in Computer Science, 2003.

[CCF+16] Anne Canteaut, Sergiu Carpov, Caroline Fontaine, Tancrède Lepoint, María Naya-Plasencia, Pascal Paillier, and Renaud Sirdey. Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. In Fast Software Encryption - 23rd International Conference, FSE 2016, volume 9783 of Lecture Notes in Computer Science, pages 313–333. Springer, 2016.

[CCF+18] Anne Canteaut, Sergiu Carpov, Caroline Fontaine, Tancrède Lepoint, María Naya-Plasencia, Pascal Paillier, and Renaud Sirdey. Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. Journal of Cryptology, 31(3):885–916, 2018.

[CCH+24] Mingyu Cho, Woohyuk Chung, Jincheol Ha, Jooyoung Lee, Eun-Gyeol Oh, and Mincheol Son. FRAST: tfhe-friendly cipher based on random s-boxes. IACR Trans. Symmetric Cryptol., 2024(3):1–43, 2024.

[CCK+13] 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, pages 315–335. Springer, 2013.

[CCR19] Hao Chen, Ilaria Chillotti, and Ling Ren. Onion ring ORAM: efficient constant bandwidth oblivious RAM from (leveled) TFHE. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, pages 345–360. ACM, 2019.

[CDPP22] Kelong Cong, Debajyoti Das, Jeongeun Park, and Hilder VL Pereira. Sortinghat: Efficient private decision tree evaluation via homomorphic encryption and transciphering. In Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, pages 563–577, 2022.

[CGGI16a] 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, pages 3–33. Springer, 2016.

[CGGI16b] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Tfhe: Fast fully homomorphic encryption library. Online: https://tfhe.github.io/tfhe, 2016.

[CGGI17] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster packed homomorphic operations and efficient circuit bootstrapping for TFHE. In Advances in Cryptology - ASIACRYPT 2017, volume 10624 of Lecture Notes in Computer Science, pages 377–408. Springer, 2017.

[CGGI20] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Tfhe: fast fully homomorphic encryption over the torus. Journal of Cryptology, 33(1):34–91, 2020.

[CHK19] Jung Hee Cheon, Kyoohyung Han, and Duhyeong Kim. Faster bootstrapping of fhe over the integers. In International Conference on Information Security and Cryptology, pages 242–259. Springer, 2019.

[CHK+21] 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 Lecture Notes in Computer Science, pages 640–669. Springer, 2021.

[CHMS22] Orel Cosseron, Clément Hoffmann, Pierrick Méaux, and François-Xavier Standaert. Towards case-optimized hybrid homomorphic encryption - featuring the elisabeth stream cipher. In Shweta Agrawal and Dongdai Lin, editors, Advances in Cryptology - ASIACRYPT 2022, volume 13793 of Lecture Notes in Computer Science, pages 32–67. Springer, 2022.

[CIR22] Carlos Cid, John Petter Indrøy, and Håvard Raddum. FASTA - A stream cipher for fast FHE evaluation. In Steven D. Galbraith, editor, Topics in Cryptology - CT-RSA 2022, volume 13161 of Lecture Notes in Computer Science, pages 451–483. Springer, 2022.

[CJP21] Ilaria Chillotti, Marc Joye, and Pascal Paillier. Programmable bootstrapping enables efficient homomorphic inference of deep neural networks. In Cyber Security Cryptography and Machine Learning: 5th International Symposium, CSCML 2021, pages 1–19. Springer, 2021.

[CKK20] Jung Hee Cheon, Dongwoo Kim, and Duhyeong Kim. Efficient homomorphic comparison methods with optimal complexity. In Advances in Cryptology–ASIACRYPT 2020, pages 221–256. Springer, 2020.

[CKKS17] JungHee Cheon, Andrey Kim, Miran Kim, and YongSoo Song. Homomorphic encryption for arithmetic of approximate numbers. In ASIACRYPT, volume 10624 of LNCS, pages 409–437. Springer, 2017.

[CLOT21] Ilaria Chillotti, Damien Ligier, Jean-Baptiste Orfila, and Samuel Tap. Improved programmable bootstrapping with larger precision and efficient arithmetic circuits for tfhe. In Advances in Cryptology–ASIACRYPT 2021, pages 670–699. Springer, 2021.

[CLR17] Hao Chen, Kim Laine, and Peter Rindal. Fast private set intersection from homomorphic encryption. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pages 1243–1255, 2017.

[CLT14] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Scale-invariant fully homomorphic encryption over the integers. In Public-Key Cryptography–PKC 2014, pages 311–328. Springer, 2014.

[CLW+24] Chunling Chen, Xianhui Lu, Ruida Wang, Zhihao Li, Xuan Shen, and Benqiang Wei. Free-xor gate bootstrapping. IACR Cryptol. ePrint Arch., page 1703, 2024.

[DEG+18] Christoph Dobraunig, Maria Eichlseder, Lorenzo Grassi, Virginie Lallemand, Gregor Leander, Eik List, Florian Mendel, and Christian Rechberger. Rasta: A cipher with low anddepth and few ands per bit. In Hovav Shacham and Alexandra Boldyreva, editors, Advances in Cryptology - CRYPTO 2018, volume 10991 of Lecture Notes in Computer Science, pages 662–692. Springer, 2018.

[DEM16] Christoph Dobraunig, Maria Eichlseder, and Florian Mendel. Higher-order cryptanalysis of lowmc. In Information Security and Cryptology-ICISC 2015, pages 87–101. Springer, 2016.

[DGGK21] 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, pages 3–34. Springer, 2021.

[DGH+23] Christoph Dobraunig, Lorenzo Grassi, Lukas Helminger, Christian Rechberger, Markus Schofnegger, and Roman Walch. Pasta: A case for hybrid homomorphic encryption. IACR Trans. Cryptogr. Hardw. Embed. Syst., 2023(3):30–73, 2023.

[DHS16] Yarkın Doröz, Yin Hu, and Berk Sunar. Homomorphic aes evaluation using the modified ltv scheme. Designs, Codes and Cryptography, 80:333–358, 2016.

[DLMW15] Itai Dinur, Yunwen Liu, Willi Meier, and Qingju Wang. Optimized interpolation attacks on lowmc. In International Conference on the Theory and Application of Cryptology and Information Security, pages 535–560. Springer, 2015.

[DLR16] Sébastien Duval, Virginie Lallemand, and Yann Rotella. Cryptanalysis of the flip family of stream ciphers. In Annual International Cryptology Conference, pages 457–475. Springer, 2016.

[DLSK05] Jana Dittmann, Andreas Lang, Martin Steinebach, and Stefan Katzenbeisser. Ecrypt-european network of excellence in cryptology. aspekte der sicherheit von mediendaten. Sicherheit 2005, Sicherheit–Schutz und Zuverlässigkeit, 2005.

[DMPS24] Nir Drucker, Guy Moshkowich, Tomer Pelleg, and Hayim Shaul. BLEACH: cleaning errors in discrete computations over CKKS. J. Cryptol., 37(1):3, 2024.

[DS09] Itai Dinur and Adi Shamir. Cube attacks on tweakable black box polynomials. In Advances in Cryptology-EUROCRYPT 2009, pages 278–299. Springer, 2009.

[edg] Password monitor: Safeguarding passwords in microsoft edge. https://shorturl.at/Jh1UZ.

[FLLL24] Ximing Fu, Mo Li, Shihan Lyu, and Chuanyi Liu. Bit-fixing correlation attacks on goldreich's pseudorandom generators. IACR Cryptol. ePrint Arch., page 1594, 2024.

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, 2012.

[GBJR23] Henri Gilbert, Rachelle Heim Boissier, Jérémy Jean, and Jean-René Reinhard. Cryptanalysis of elisabeth-4. In Advances in Cryptology - ASIACRYPT 2023, volume 14440 of Lecture Notes in Computer Science, pages 256–284. Springer, 2023.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Proceedings of the forty-first annual ACM symposium on Theory of computing, pages 169–178, 2009.

[GHS12] Craig Gentry, Shai Halevi, and Nigel P Smart. Homomorphic evaluation of the aes circuit. In Annual Cryptology Conference, pages 850–867. Springer, 2012.

[GKR+21] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. Poseidon: A new hash function for {Zero-Knowledge} proof systems. In 30th USENIX Security Symposium (USENIX Security 21), pages 519–535, 2021.

[GLR+24] Lorenzo Grassi, Fukang Liu, Christian Rechberger, Fabian Schmid, Roman Walch, and Qingju Wang. Minimize the randomness in rasta-like designs: How far can we go? Cryptology ePrint Archive, 2024.

[GMAH+23] 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 Annual International Cryptology Conference, pages 305–339. Springer, 2023.

[Gol00] Oded Goldreich. Candidate one-way functions based on expander graphs. Electron. Colloquium Comput. Complex., TR00-090, 2000.

[HHCP18] Kyoohyung Han, Seungwan Hong, Jung Hee Cheon, and Daejun Park. Efficient logistic regression on large encrypted data. Cryptology ePrint Archive, 2018.

[HHLW24] Jiahui He, Kai Hu, Hao Lei, and Meiqin Wang. Massive superpoly recovery with a meet-in-the-middle framework: Improved cube attacks on trivium and kreyvium. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pages 368–397. Springer, 2024.

[HHW+21] Zhicong Huang, Cheng Hong, Chenkai Weng, Wen-jie Lu, and Hunter Qu. More efficient secure matrix multiplication for unbalanced recommender systems. IEEE Transactions on Dependable and Secure Computing, 20(1):551–562, 2021.

[HKC+20] Jincheol Ha, Seongkwang Kim, Wonseok Choi, Jooyoung Lee, Dukjae Moon, Hyojin Yoon, and Jihoon Cho. Masta: an he-friendly cipher using modular arithmetic. IEEE Access, 8:194741–194751, 2020.

[HKL+22] 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 Lecture Notes in Computer Science, pages 581–610. Springer, 2022.

[HL20] Phil Hebborn and Gregor Leander. Dasta - alternative linear layer for rasta. IACR Trans. Symmetric Cryptol., 2020(3):46–86, 2020.

[HMR20] Clément Hoffmann, Pierrick Méaux, and Thomas Ricosset. Transciphering, using filip and TFHE for an efficient delegation of computation. In Progress in Cryptology - INDOCRYPT 2020, volume 12578 of Lecture Notes in Computer Science, pages 39–61. Springer, 2020.

[HMS23] Clément Hoffmann, Pierrick Méaux, and François-Xavier Standaert. The patching landscape of elisabeth-4 and the mixed filter permutator paradigm. In Progress in Cryptology - INDOCRYPT 2023, volume 14459 of Lecture Notes in Computer Science, pages 134–156. Springer, 2023.

[HS20] Shai Halevi and Victor Shoup. Design and implementation of helib: a homomorphic encryption library. Cryptology ePrint Archive, 2020.

[HST+21] Kai Hu, Siwei Sun, Yosuke Todo, Meiqin Wang, and Qingju Wang. Massive superpoly recovery with nested monomial predictions. In Mehdi Tibouchi and Huaxiong Wang, editors, Advances in Cryptology - ASIACRYPT 2021, volume 13090 of Lecture Notes in Computer Science, pages 392–421. Springer, 2021.

[JK97] Thomas Jakobsen and Lars R Knudsen. The interpolation attack on block ciphers. In Fast Software Encryption: 4th International Workshop, FSE'97, pages 28–40. Springer, 1997.

[JKA+21] 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. IACR Transactions on Cryptographic Hardware and Embedded Systems, pages 114–148, 2021.

[JKLS18] Xiaoqian Jiang, Miran Kim, Kristin Lauter, and Yongsoo Song. Secure outsourced matrix computation and application to neural networks. In Proceedings of the 2018 ACM SIGSAC conference on computer and communications security, pages 1209–1222, 2018.

[KKC+23] Jongmin Kim, Sangpyo Kim, Jaewan Choi, Jaiyoung Park, Donghwan Kim, and Jung Ho Ahn. Sharp: A short-word hierarchical accelerator for robust and practical fully homomorphic encryption. In Proceedings of the 50th Annual International Symposium on Computer Architecture, pages 1–15, 2023.

[KKK+22] Sangpyo Kim, Jongmin Kim, Michael Jaemin Kim, Wonkyung Jung, John Kim, Minsoo Rhu, and Jung Ho Ahn. Bts: An accelerator for bootstrappable fully homomorphic encryption. In Proceedings of the 49th annual international symposium on computer architecture, pages 711–725, 2022.

[Knu95] Lars R Knudsen. Truncated and higher order differentials. In Fast Software Encryption: Second International Workshop Leuven, Belgium, December 14–16, 1994 Proceedings 2, pages 196–211. Springer, 1995.

[Kön08] Robert Könighofer. A fast and cache-timing resistant implementation of the aes. In Cryptographers' Track at the RSA Conference, pages 187–202. Springer, 2008.

[KS09] Emilia Käsper and Peter Schwabe. Faster and timing-attack resistant aes-gcm. In International Workshop on Cryptographic Hardware and Embedded Systems, pages 1–17. Springer, 2009.

[lat23] Lattigo v5. Online: https://github.com/tuneinsight/lattigo, November 2023. EPFL-LDS, Tune Insight SA.

[LATV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Proceedings of the forty-fourth annual ACM symposium on Theory of computing, pages 1219–1234, 2012.

[LAW+23] Fukang Liu, Ravi Anand, Libo Wang, Willi Meier, and Takanori Isobe. Coefficient grouping: Breaking chaghri and more. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pages 287–317. Springer, 2023.

[LHH+21] 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. IEEE, 2021.

[LIM21] Fukang Liu, Takanori Isobe, and Willi Meier. Cryptanalysis of full lowmc and lowmc-m with algebraic techniques. In Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, pages 368–401. Springer, 2021.

[LKSM24] Fukang Liu, Abul Kalam, Santanu Sarkar, and Willi Meier. Algebraic attack on fhe-friendly cipher hera using multiple collisions. IACR Transactions on Symmetric Cryptology, 2024(1):214–233, 2024.

[LLC+24] Fen Liu, Yongqiang Li, Huiqin Chen, Lin Jiao, Ming Luo, and Mingsheng Wang. Yux: Finite field multiplication based block ciphers for efficient fhe evaluation. IEEE Transactions on Information Theory, 2024.

[LLL+] Eunsang Lee, Joon-Woo Lee, Junghyun Lee, Young-Sik Kim, Yongjune Kim, Jong-Seon No, and Woosuk Choi. Low-complexity deep convolutional neural networks on fully homomorphic encryption using multiplexed parallel convolutions. In International Conference on Machine Learning, ICML 2022.

[LSMI21] Fukang Liu, Santanu Sarkar, Willi Meier, and Takanori Isobe. Algebraic attacks on rasta and dasta using low-degree equations. In International Conference on the Theory and Application of Cryptology and Information Security, pages 214–240. Springer, 2021.

[MCJS19] Pierrick Méaux, Claude Carlet, Anthony Journault, and François-Xavier Standaert. Improved filter permutators for efficient FHE: better instances and implementations. In Feng Hao, Sushmita Ruj, and Sourav Sen Gupta, editors, Progress in Cryptology - INDOCRYPT 2019, volume 11898 of Lecture Notes in Computer Science, pages 68–91. Springer, 2019.

[MG21] Kalikinkar Mandal and Guang Gong. Homomorphic evaluation of lightweight cipher boolean circuits. In Esma Aïmeur, Maryline Laurent, Reda Yaich, Benoît Dupont, and Joaquín García-Alfaro, editors, Foundations and Practice of Security, volume 13291 of Lecture Notes in Computer Science, pages 63–74. Springer, 2021.

[MJSC16] Pierrick Méaux, Anthony Journault, François-Xavier Standaert, and Claude Carlet. Towards stream ciphers for efficient FHE with low-noise ciphertexts. In Advances in Cryptology - EUROCRYPT 2016, volume 9665 of Lecture Notes in Computer Science, pages 311–343. Springer, 2016.

[MN07] Mitsuru Matsui and Junko Nakajima. On the power of bitslice implementation on intel core2 processor. In Cryptographic Hardware and Embedded Systems-CHES 2007: 9th International Workshop, Vienna, Austria, September 10-13, 2007. Proceedings 9, pages 121–134. Springer, 2007.

[MPP24] Pierrick Méaux, Jeongeun Park, and Hilder V. L. Pereira. Towards practical transciphering for FHE with setup independent of the plaintext space. IACR Commun. Cryptol., 1(1):20, 2024.

[MV04] David A McGrew and John Viega. The security and performance of the galois/counter mode (gcm) of operation. In International Conference on Cryptology in India, pages 343–355. Springer, 2004.

[NHY+25] Chao Niu, Zhicong Huang, Zhaomin Yang, Yi Chen, Liang Kong, Cheng Hong, and Tao Wei. Xboot: Free-xor gates for ckks with applications to transciphering. Cryptology ePrint Archive, 2025.

[NLV11] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In Proceedings of the 3rd ACM workshop on Cloud computing security workshop, pages 113–124, 2011.

[OPP23] Hiroki Okada, Rachel Player, and Simon Pohmann. Homomorphic polynomial evaluation using galois structure and applications to bfv bootstrapping. In International Conference on the Theory and Application of Cryptology and Information Security, pages 69–100. Springer, 2023.

[PKK+23] Jaiyoung Park, Donghwan Kim, Jongmin Kim, Sangpyo Kim, Wonkyung Jung, Jung Hee Cheon, and Jung Ho Ahn. Toward practical privacy-preserving convolutional neural networks exploiting fully homomorphic encryption. arXiv preprint arXiv:2310.16530, 2023.

[RAD+78] Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. On data banks and privacy homomorphisms. Foundations of secure computation, 4(11):169–180, 1978.

[RB08] Matthew J. B. Robshaw and Olivier Billet, editors. New Stream Cipher Designs - The eSTREAM Finalists, volume 4986 of Lecture Notes in Computer Science. Springer, 2008.

[Rec] Christian Rechberger. On the history of fhempczk-friendly symmetric crypto. https://who.paris.inria.fr/Leo.Perrin/rescale/slides/Christian-STAP-2023.pdf.

[RST18] Christian Rechberger, Hadi Soleimany, and Tyge Tiessen. Cryptanalysis of low-data instances of full lowmcv2. IACR Transactions on Symmetric Cryptology, 2018(3):163–181, 2018.

[SEA23] Microsoft SEAL (release 4.1). https://github.com/Microsoft/SEAL, January 2023. Microsoft Research, Redmond, WA.

[SMK22] Roy Stracovsky, Rasoul Akhavan Mahdavi, and Florian Kerschbaum. Faster evaluation of aes using tfhe. Poster Session, FHE.Org - 2022, 2022. Available at https://rasoulam.github.io/data/poster-aes-tfhe.pdf.

[SV14] Nigel P Smart and Frederik Vercauteren. Fully homomorphic simd operations. Designs, codes and cryptography, 71:57–81, 2014.

[TCBS23] Daphné Trama, Pierre-Emmanuel Clet, Aymen Boudguiga, and Renaud Sirdey. A homomorphic aes evaluation in less than 30 seconds by means of tfhe. In Proceedings of the 11th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, pages 79–90, 2023.

[TKLP25] Indranil Thakur, Angshuman Karmakar, Chaoyun Li, and Bart Preneel. A survey on transciphering and symmetric ciphers for homomorphic encryption. Cryptology ePrint Archive, 2025.

[VDGHV10] Marten Van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Advances in Cryptology–EUROCRYPT 2010, pages 24–43. Springer, 2010.

[WLW+24] Benqiang Wei, Xianhui Lu, Ruida Wang, Kun Liu, Zhihao Li, and Kunpeng Wang. Thunderbird: Efficient homomorphic evaluation of symmetric ciphers in 3gpp by combining two modes of tfhe. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2024(3):530–573, 2024.

[WWL+23] Benqiang Wei, Ruida Wang, Zhihao Li, Qinju Liu, and Xianhui Lu. Fregata: Faster homomorphic evaluation of AES via TFHE. In Information Security - 26th International Conference, ISC 2023, volume 14411 of Lecture Notes in Computer Science, pages 392–412. Springer, 2023.

[WWL+24] Ruida Wang, Yundi Wen, Zhihao Li, Xianhui Lu, Benqiang Wei, Kun Liu, and Kunpeng Wang. Circuit bootstrapping: Faster and smaller. In Advances in Cryptology - EUROCRYPT 2024, volume 14652 of Lecture Notes in Computer Science, pages 342–372. Springer, 2024.

[Zam22] Zama. Concrete: TFHE Compiler that converts python programs into FHE equivalent, 2022. https://github.com/zama-ai/concrete.