The HHE Land: Exploring the Landscape of Hybrid Homomorphic Encryption *

Hossein Abdinasibfar[1], Camille Nuoskala[1] and Antonis Michalas[1,2]

[1] Tampere University, Tampere, Finland,
{hossein.abdinasibfar,camille.nuoskala,antonios.michalas}@tuni.fi
[2] RISE Research Institutes of Sweden, Gothenburg, Sweden

Abstract. Hybrid Homomorphic Encryption (HHE) is considered a promising solution for key challenges that emerge when adopting Homomorphic Encryption (HE). In cases such as communication and computation overhead for clients and storage overhead for servers, it combines symmetric cryptography with HE schemes. However, despite a decade of advancements, enhancing HHE usability, performance, and security for practical applications remains a significant stake. This work contributes to the field by presenting a comprehensive analysis of prominent HHE schemes, focusing on their performance and security. We implemented three superior schemes–PASTA, HERA, and Rubato–using the Go programming language and evaluated their performance in a client-server setting. To promote open science and reproducibility, we have made our implementation publicly available on GitHub. Furthermore, we conducted an extensive study of applicable attacks on HHE schemes, categorizing them under algebraic-based, differential-based, linear-based, and LWE-based attacks. Our security analysis revealed that while most existing schemes meet theoretical security requirements, they remain vulnerable to practical attacks. These findings emphasize the need for improvements in practical security measures, such as defining standardized parameter sets and adopting techniques like noise addition to counter these attacks. This survey provides insights and guidance for researchers and practitioners to design and develop secure and efficient HHE systems, paving the way for broader real-world adoption.

Keywords: Applied Cryptography · HE-friendly Ciphers · Homomorphic Encryption · Hybrid Homomorphic Encryption · Lattigo

1 Introduction

The versatile nature of Homomorphic Encryption (HE)—combined with the wide range of its applications—has rendered it one of the most significant fields of research in recent years. In 1978, one year after the introduction of RSA [RSA78], Rivest et al. suggested HE serves as a solution against the incompatibility between user privacy and the need for data delegation and computational requirements [RAD78]. HE performs computations on encrypted data and produces an encrypted result that can be decrypted to the same result as if the computations had been performed on unencrypted data. Several existing public-key cryptosystems, including RSA [RSA78], ElGamal [ElG85], and Goldwasser-Micali [GM19], present homomorphic properties, though none supports the computation of arbitrary functions on ciphertexts. The goal of obtaining a fully homomorphic encryption (FHE) scheme continued until Gentry's breakthrough in 2009 [Gen09]. This scheme dealt with the difficulty of specific mathematical problems concerning ideal lattices and enabled *unrestricted* computations on encrypted data. Through unrestricted computation feature,

*This work was funded by the HARPOCRATES EU research project (No. 101069535).

FHE has been brought as the leading solution to many applications, such as private information retrieval [ACLS18], private set intersection [CLR17, CHLR18, CMdG+21], and privacy-preserving machine learning [HHCP18, LLL+22].

However, the practical adoption of FHE schemes in real-world applications is hindered by high computational, storage, and communication requirements. FHE ciphertexts impose significant storage and transfer overhead, leading to considerable storage and communication costs. Historically, these inefficiencies were overlooked, as prior applications assumed the availability of a hypothetical, resource-unlimited Cloud Service Provider (CSP). However, FHE is ill-suited for resource-constrained environments, such as the Internet of Things (IoT), fog computing, or federated learning in limited-capacity networks, where numerous entities generate data and require privacy-enhancing technologies. Beyond these scenarios, even in the context of unlimited resources, application owners face the financial burden of storage costs, which increase rapidly when storing FHE ciphertexts for further use.

As stated in [NLV11], HE schemes have yet to provide efficient communication and computation on the client side, thus becoming inapplicable for many real-world scenarios. This is when the idea of Hybrid Homomorphic Encryption (HHE) was put forth. HHE integrates one or more HE schemes with a symmetric cipher to create a robust and efficient encryption system [NLV11]. Employing a Symmetric Key Encryption (SKE) scheme allows users to encrypt data with an encryption ratio that ensures ciphertext sizes are nearly equal to plaintext sizes. This approach significantly reduces storage and communication costs while offering low computational overhead compared to HE schemes. The symmetrically encrypted ciphertexts are transmitted to and stored by a CSP. Upon user request, the CSP employs a symmetric decryption circuit, implemented homomorphically as a *transform* function, to convert the stored symmetric ciphertexts into homomorphic ciphertexts. These homomorphic ciphertexts are then processed as inputs for the requested computation, following the standard HE workflow. By integrating SKE with HE, an HHE system enhances computational capabilities while mitigating the storage and communication overhead commonly associated with HE schemes [DGH+23]. HHE has already demonstrated its potential in various domains, including privacy-preserving machine learning [BFM22, CHMS22, FNB+24, NBF+24] and IoT applications [HD24].

Note 1.1: HHE trade-off

HHE significantly reduces bandwidth requirements and computational costs on the client side, as well as storage costs on the server side. However, this efficiency is accompanied by increased computational overhead in the encrypted domain (i.e., server-side computations) due to the SKE to HE ciphertext *transformation*.

1.1 Contribution

Though a variety of HHE schemes have been proposed over the past decade, to the best of our knowledge, no published work attempts to analyze, compare, and evaluate these schemes, and this is precisely what this work aims to do. Our contribution can be summarized as follows:

C1. We lay the basic theoretical foundation allowing researchers to comprehend current HE and HHE schemes.

C2. We provide a detailed description of how the most important HHE schemes–PASTA, HERA, Rubato, and Elisabeth– work and make a thorough comparison.

C3. We analyze the security properties of current HHE schemes and demonstrate that, although most existing schemes satisfy the necessary theoretical security properties, they are still susceptible to attacks.

C4. We analyze various implementation aspects such as processing power, memory allocation, security, and ease of running a scheme. Furthermore, we elaborate on the limitations preventing the existing HHE solutions from being implemented in real-world settings. A real-world setting follows a client-server architecture, where the client can use the symmetric cipher independently, and the server performs HHE computations.

C5. We support open science and reproducible research by making our code available online on GitHub. Specifically, to provide a fair and solid comparison of the examined works, we built our HHE library with Go programming language, using a modular approach, to run superior HHE schemes, namely PASTA, HERA, and Rubato.

1.2 Organization

The rest of this paper is structured as follows. Section 2 provides some necessary background information to understand the core components of HE and HHE ciphers. Section 3 delivers definitions and foundations for HE and four common HE schemes. Section 4 continues by providing our universal definition of HHE and offers a brief history of HE-friendly ciphers. Additionally, it elaborates on the building blocks of HHE schemes, highlighting their enhancements over the years. It then outlines four cutting-edge HHE schemes and their underlying techniques. Section 5 presents the evaluation results of our HHE implementation, considering both client and server perspectives in a real-world setting. Section 6 discusses the semantic security of an HHE scheme based on the universal definition we provided earlier. Furthermore, we take the security analysis of HHE one step further by studying and describing a wide range of attacks that can be applied to HE-friendly ciphers and are not related to semantic security. Section 7 categorizes various attacks on HE-friendly ciphers. Finally, in Section 8, we present the key insights, highlight identified research gaps, and outline proposed directions for future work.

2 Background

2.1 Notation

Throughout the paper, vectors are represented in lowercase bold letters, and matrices in capital letters. Let $[n]$ be the set of integers from 1 to n. $\lfloor \cdot \rceil$ denotes the nearest integer to a real number, while $\langle \cdot, \cdot \rangle$ represents the inner product of two vectors. $[\cdot]_q$ signifies the mod q reduction, and $\|\mathbf{v}\|_p$ denotes the ℓ_p-norm of the vector $\mathbf{v}$. We denote λ as the security parameter of a cryptosystem. For a probability distribution χ, we denote $x \xleftarrow{\chi} S$ if x is sampled from a set S according to the distribution χ, with \$ the uniformly random distribution. Finally, $\mathcal{D}_{\mathcal{S},\sigma}$ stands for the Gaussian distribution in a set $\mathcal{S}$ for width σ. We use $\mathbb{G}$ to denote an additive group, $\mathbb{Z}_p$ to denote the set of integers modulo p, $\mathbb{F}_p$ the field of integers modulo p, and $\mathcal{R}_n$ the quotient ring $\mathbb{Z}_n[X]/(P)$ of polynomials over $\mathbb{Z}_n[X]$ by the ideal generated by an irreducible polynomial P. We use $\mathcal{P}$ to indicate the plaintext space and $\mathcal{C}$ to indicate the ciphertext space. We use the notation $\mathbf{v_1} \odot \mathbf{v_2}$ to represent the element-wise product between two vectors. Finally, we use the terms `pk`, `sk`, and `evk` to denote public, secret, and evaluation keys, respectively.

2.2 Substitution-Permutation Network (SPN)

SPN is a cryptographic construction used to design block ciphers [Fei73]. The SPN consists of non-linear substitutions (S-boxes) and linear bit permutations based on confusion and diffusion [Sha49]. N represents the block size of a basic SPN consisting of r-rounds of $n \times n$ s-boxes [YTH96].

Permutation. We call permutation any bijection of a set in itself. The permutation of a vector $\mathbf{x} = (x_1, \ldots, x_n)$ is a rearrangement of its elements $\left(x_{\sigma(1)}, \ldots, x_{\sigma(n)}\right)$, for $\sigma(i) \in [1, n]$ and $\sigma(i) \neq \sigma(j)$, $i \neq j$.

S-Box. An essential part of designing a symmetric cipher is choosing an efficient S-Box for the non-linear layer. Authors in [DGH+23] analyzed and compared different S-Boxes. Here, we add the missing ones and redefine all with our notation for future use in Section 4.

χ-S-box: This is a non-linear function that takes as input a vector of n elements in $\mathbb{Z}_q$ and outputs a vector of n elements in $\mathbb{Z}_q$. The χ-*S-box* is defined as follows:

$$S_\chi(\mathbf{x})_i = x_i + x_{i+2} + (x_{i+1} \cdot x_{i+2}) = x_i + x_{i+2} \cdot (1 + x_{i+1}) .$$

Cube S-Box: Given a prime p, such that $gcd(p-1, 3) = 1$, the cube S-Box is defined as follows:

$$S_c(\mathbf{x})_i = (x_i)^3 .$$

Feistel-Like S-Box via a Quadratic Function: The Feistel-like S-Box is defined as follows:

$$S_{fq}(\mathbf{x})_i = \begin{cases} x_i & \text{if } i = 0, \\ x_i + (x_{i-1})^2 & \text{otherwise,} \end{cases}$$

Feistel-Like S-Box via the χ-Function: The Feistel-like S-Box is defined as follows:

$$S_{f\chi}(\mathbf{x})_i = \begin{cases} x_i & \text{if } i \leq 1, \\ x_i + (x_{i-1} \cdot x_{i-2}) & \text{otherwise,} \end{cases}$$

LowMC S-Box: For three input bits a, b, and c, the LowMC S-box is defined as follows:

$$S_{mc}(a, b, c) = (a \oplus bc, a \oplus b \oplus ac, a \oplus b \oplus c \oplus ab)$$

2.3 Stream Ciphers

As defined in [QYS+23], most stream ciphers are a generic construction that can be represented through a finite state machine characterized by the key (K), the initial vector (IV), and the internal state (S). An initialization function expands K and the initial value of IV into an initial internal state. Subsequently, the internal state undergoes updates via an update function, and the output function generates an output based on the final state.

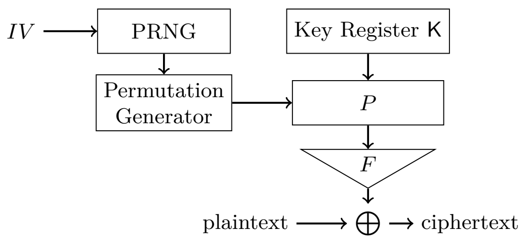

Figure 1: Filter Permutator Construction [MJSC16]

2.4 Filter Permutator

As illustrated in Figure 1, a filter permutator (FP) consists of a constant key register (K), a permutation generator, a pseudo-random number generator (PRNG) initialized with a public IV, a pseudo-random permutation (P), and a non-linear filter function (F). Based on the PRNG output, the permutation generator applies a new bit-permutation to K in each cycle. The filtering function F generates a keystream bit from the permuted key, which is then XORed with the plaintext to create ciphertext.

2.5 Look-Up Table (LUT)

A look-up table L over a set S is an array of N elements in S, indexed by $i \in [N]$. We denote $L[i] \in S$, the value of L stored in position i. In this work, we only focus on Negacyclic LUT (NLUT), defined below. Note that if we can define an LUT on any set, an NLUT can only be defined over a group. In this article, this group is the real torus $\mathbb{T} = \mathbb{R}/\mathbb{Z}$.

Definition 1 (Negacyclic Look-Up Table (NLUT))**.** A negacyclic look-up table over $\mathbb{T}$ is a look-up table of size $2N$ such that $\forall i \in [N]$, $L[N+i] = -L[i]$

In practice, an NLUT is represented by a polynomial $P \in \mathbb{Z}_{q,N}[X]$. Denote $P = a_0 + a_1 X + \ldots + a_{N-1}X^{N-1}$, where $2N$ is the size of the LUT. The value of the i-th entry is given by $(X^{-i} \cdot P) \mod X$. This polynomial representation allows the homomorphic evaluation of an NLUT presented in Section 4.6.

2.6 Learning With Errors (LWE)

LWE was introduced and established by Regev in 2005 [Reg05, Reg09], showcasing its comparable worst-case hardness properties through a quantum reduction. With the diversification of cryptographic protocols, new variants of this problem have emerged, such as RLWE [LPR13] or TLWE [CGGI20]. These variants reduce to the same hardness assumption, which we call GLWE. Due to space constraints, we formally define only the General LWE (GLWE) problem.

Definition 2 (GLWE sample)**.** Let $\mathcal{R}$ be a ring and $\mathcal{R}_N[X] := \mathcal{R}[X]/(X^N + 1)$ for N a power of 2. For a positive integer k, a vector $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_N[X]^k$ and a polynomial $\mu \in \mathcal{R}_N[X]$, we define $\mathtt{GLWE}_{\mathbf{s}}(\mu) = (\mathbf{a}, \mu + \langle \mathbf{s}, \mathbf{a} \rangle + e)$, where $\mathbf{a} \xleftarrow{\$} \mathcal{R}_N[X]^k$ and $e \xleftarrow{\chi} \mathcal{R}_N[X]$ for an error distribution χ. The trivial sample $\mathtt{GLWE}_{\mathbf{s}}(0_{\mathcal{R}}) = (\mathbf{a}, \langle \mathbf{z}, \mathbf{a} \rangle + e)$ is called the $\mathtt{GLWE}$ distribution.

Definition 3 (GLWE problem)**.** Let $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_N[X]^k$ be a secret vector.

- Search GLWE: Given $\mathtt{GLWE}_{\mathbf{s}}(0) = (\mathbf{a}, \langle \mathbf{z}, \mathbf{a} \rangle + e)$ and a distribution χ, find $\mathbf{s}$;
- Decisional GLWE: Given a pair $(\mathbf{a}, b) \in \mathcal{R}_N[X]^{k+1}$ and the distribution χ, decide if b is chosen at random, *i.e.* $b \xleftarrow{\$} \mathcal{R}_N[X]$, or if it follows a $\mathtt{GLWE}$ distribution that is $(\mathbf{a}, b) = \mathtt{GLWE}_{\mathbf{s}}(0_{\mathcal{R}})$.

The classical LWE problem stated by Regev corresponds to the case $N = 1$ and $\mathcal{R} = \mathbb{Z}_q$, for q a power of a prime. In short, an LWE sample is given by $\mathtt{LWE}_{\mathbf{s}}(\mu) = ((a_1, \ldots, a_k), \mu + \langle \mathbf{s}, \mathbf{a} \rangle + e)$, for $\mathbf{a} = (a_1, \ldots, a_k) \in \mathbb{Z}_q^k$, $\mathbf{s} \in \mathbb{Z}^k$, $\mu \in \mathbb{Z}_q$ and $e \xleftarrow{\chi} \mathbb{Z}_q$.

General Gentry-Sahai-Waters (GGSW). A GGSW encryption, named after Gentry *et al.* [GSW13] who first introduced it, is an extension of GLWE. In a nutshell, it is given by a vector of GLWE distributions. Consider an integer plaintext $\mu \in \mathbb{Z}_N[X]^k$ and a vector $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_N[X]^k$. For $d > 0$ its *depth* and a basis $\beta > 0$, a GGSW sample is given by

$\texttt{GGLW}_{\mathbf{s}}(\mu) = \mathbf{Z} + \mu \cdot \mathbf{G}^{(\beta)} \in \mathbb{T}_N[X]^{d(k+1)\times(k+1)}$, where $\mathbf{Z}$ is a matrix of $d(k+1)$ GLWE distributions and $\mathbf{G}^{(\beta)}$ is the gadget matrix in $\mathbb{T}_N[X]^{d(k+1)\times(k+1)}$ given by $G_{i,j}^{(\beta)} = \mathbf{I}_{k+1} \otimes \mathbf{g}$ for $\mathbf{I}_{k+1}$ the identity matrix, $\otimes$ the tensor product and $\mathbf{g}$ the vector of size d given by $g_i = \beta^{-i}, i \in [d]$.

3 Homomorphic Encryption

In this section, we first present the general HE definition and its correctness. Subsequently, we explore the four HE schemes commonly employed across various HHE schemes.

Over the years, researchers have developed three types of HE: Partially Homomorphic Encryption (PHE), Somewhat Homomorphic Encryption (SHE), and Fully Homomorphic Encryption (FHE), each with distinct properties and capabilities. FHE, though computationally intensive and resource-demanding, enables arbitrary computations on ciphertexts, including addition, multiplication, and other complex operations. In contrast, PHE supports only a single operation, either addition or multiplication, making it computationally less demanding but significantly less flexible than FHE. SHE offers an intermediate solution, allowing a limited number of additions and multiplications while being more practical than PHE and less resource-intensive than FHE.

Since the groundbreaking work by Gentry [Gen09], numerous HE schemes have emerged to handle diverse data types. Notably, recommended by Homomorphic Encryption Standard [ACC+21], the Brakerski-Gentry-Vaikuntanathan (BGV) scheme [BGV12] supports modular arithmetic over finite fields, enabling computations on vectors of integers in $\mathbb{Z}_q$ with $q \geq 2$. Similarly, the Brakerski/Fan-Vercauteren (B/FV) scheme [Bra12, FV12] operates over finite fields and facilitates computations on integer vectors. The Cheon-Kim-Kim-Song (CKKS) scheme [CKKS17] allows approximate computations on vectors of real and complex numbers. Additionally, the Torus FHE (TFHE) scheme [CGGI16, CGGI17, CGGI20] employs boolean circuits and decision diagrams for low-precision integers [CJP21]. Each HE scheme is meticulously designed to achieve specific optimization goals. BGV, B/FV, and CKKS prioritize minimizing the multiplicative depth in their decryption circuits, while TFHE focuses on reducing the gate count required for decryption operations.

Def. 3.1: General HE Definition

An $\texttt{HE} := (\texttt{KeyGen}, \texttt{Enc}, \texttt{Eval}, \texttt{Dec})$ scheme is a tuple of four algorithms defined as follows:

1. $\texttt{KeyGen}\left(1^{\lambda}\right)$ takes as input a security parameter λ and outputs the public key $\texttt{pk}$, the secret key $\texttt{sk}$ and the evaluation key $\texttt{evk}$;
2. $\texttt{Enc}\left(\texttt{pk}, \mathbf{m}\right)$ takes as input the public key $\texttt{pk}$ and a message $\mathbf{m}$ and outputs a ciphertext $\mathbf{c_m}$;
3. $\texttt{Eval}\left(\texttt{evk}, f, (c_{m_1}, \ldots, c_{m_n})\right)$ takes as input the evaluation key $\texttt{evk}$, a function f and an n-tuple of ciphertexts $(c_{m_1}, \ldots, c_{m_n})$ and outputs a ciphertext $\mathbf{c}^f$;
4. $\texttt{Dec}\left(\texttt{sk}, \mathbf{c}^f\right)$ takes as input the secret key $\texttt{sk}$ and a ciphertext $\mathbf{c}^f$ and outputs $\mathbf{m}^f$.

Correctness: An HE scheme is correct if:

$$Pr\left[\texttt{Dec}\left(\texttt{sk}, \mathbf{c}^f\right) \neq f(m_1, \ldots, m_n) \;\middle|\; \left[(\texttt{pk}, \texttt{sk}, \texttt{evk}) \leftarrow \texttt{KeyGen}\left(1^{\lambda}\right)\right]\right.$$
$$\left.\wedge \left[\mathbf{c}^f \leftarrow \texttt{Eval}\left(\texttt{evk}, f, (c_{m_1}, \ldots, c_{m_n})\right)\right]\right] = negl(\lambda).$$

3.1 BGV

BGV is an SHE scheme based on RLWE that supports modular arithmetic over finite fields [BGV12]. BGV construction is scale-dependent, relying on a predetermined sequence of moduli $\mathcal{Q} = \{q_0, q_1, \ldots, q_L\}$. Each modulus is the ciphertext scale for a certain level of operation, and each multiplication requires a modulus switch.

Setup. Given a security parameter λ, generate two integers $t, n > 0$ and $q_0 < \ldots < q_L$ a sequence of powers of prime q_ℓ with $\ell \in [L]$. Then set $\mathcal{P} = \mathcal{R}_t = \mathbb{Z}_t[X]/(X^n + 1)$ the plaintext space and $\mathcal{C} = \mathcal{R}_{q_l} \times \mathcal{R}_{q_l}$ the ciphertext space at level ℓ, for $\mathcal{R}_{q_l} = \mathbb{Z}_{q_l}[X]/(X^n + 1)$. Finally, denote $\mathcal{D}_{q_l,\sigma}$ the discrete Gaussian distribution over $\mathcal{R}_{q_l}$ of width σ. The BGV algorithms are illustrated in Definition 3.2.

Def. 3.2: BGV

1. $\mathtt{KeyGen}\left(1^\lambda\right)$ takes as input a security parameter λ. Sample $\mathtt{sk} \xleftarrow{\$} \mathcal{R}$ with coefficients in $\{-1, 0, 1\}$, $a \xleftarrow{\$} \mathcal{R}_{q_L}$, and $e \leftarrow \mathcal{D}_{q_L,\sigma}$; set $\mathtt{pk} = (b, a) \in \mathcal{R}^2_{q_L}$, where $b \leftarrow [-a \cdot \mathtt{sk} + t \cdot e]_{q_L}$, and $\mathtt{evk} = (b', a) \in \mathcal{R}^2_{q_L}$, where $b' \leftarrow \left[-(a \cdot \mathtt{sk} + e) + \mathtt{sk}^2\right]_{q_l}$; outputs $(\mathtt{sk}, \mathtt{pk}, \mathtt{evk})$;

2. $\mathtt{Enc}(\mathtt{pk}, m)$ takes as input the public key $\mathtt{pk}$ and a plaintext $m \in \mathcal{P}$. Sample $u \xleftarrow{\$} \mathcal{R}$ with coefficients in $\{-1, 0, 1\}$, and $e_1, e_2 \leftarrow \mathcal{D}_{q_\ell,\sigma}$. Set $c_1 \leftarrow [b \cdot u + t \cdot e_1 + m]_{q_\ell}$ and $c_2 \leftarrow [a \cdot u + t \cdot e_2]_{q_\ell}$; outputs $c = (c_1, c_2)$;

3. $\mathtt{EvalAdd}\left(c^{(1)}, c^{(2)}\right)$ takes as input two ciphertexts $c^{(1)}, c^{(2)}$ and computes $c_1^{\mathtt{add}} \leftarrow \left[c_1^{(1)} + c_1^{(2)}\right]_{q_\ell}$ and $c_2^{\mathtt{add}} \leftarrow \left[c_2^{(1)} + c_2^{(2)}\right]_{q_\ell}$; outputs $c^{\mathtt{add}} = (c_1^{\mathtt{add}}, c_2^{\mathtt{add}})$;

4. $\mathtt{EvalMult}\left(c^{(1)}, c^{(2)}\right)$ takes as input two ciphertexts $c^{(1)}, c^{(2)}$ and computes $\overline{c} = (\overline{c_1}, \overline{c_2}, \overline{c_3}) \leftarrow \left(\left[c_1^{(1)} \cdot c_1^{(2)}\right]_{q_\ell}, \left[c_1^{(1)} \cdot c_2^{(2)} + c_2^{(1)} \cdot c_1^{(2)}\right]_{q_\ell}, \left[c_2^{(1)} \cdot c_2^{(2)}\right]_{q_\ell}\right)$. Set $c^{\mathtt{mult}} \leftarrow \mathtt{Relinearize}\left(\mathtt{evk}, \overline{c}\right)$; outputs $c^{\mathtt{mult}}$;

5. $\mathtt{Relinearize}(\mathtt{evk}, \overline{c})$ takes as input the evaluation key $\mathtt{evk} = (b', a)$ and $\overline{c} = (\overline{c_1}, \overline{c_2}, \overline{c_3})$, compute $c_1 \leftarrow [\overline{c_1} + b' \cdot \overline{c_3}]_{q_l}$ and $c_2 \leftarrow [\overline{c_2} + a \cdot \overline{c_3}]_{q_l}$; outputs $c = (c_1, c_2)$;

6. $\mathtt{ModSwitch}(\mathtt{evk}, c)$ takes as input the evaluation key $\mathtt{evk}$, a ciphertext c encrypted modulo q_ℓ; computes $c' \leftarrow \left\lfloor \frac{q_\ell}{q_{\ell-1}} \cdot c \right\rceil$; outputs c';

7. $\mathtt{Dec}(\mathtt{sk}, c)$ takes as input the secret key $\mathtt{sk}$ and the ciphertext c. Compute $m' \leftarrow [c_1 + c_2 \cdot \mathtt{sk}]_{q_\ell}$.

For simplicity and due to space constraints, we will not redefine functions for other HE schemes sharing the same principles as BGV.

3.2 B/FV

B/FV is another SHE scheme that supports modular arithmetic over finite fields [Bra12, FV12]. B/FV is scale-independent, meaning the ciphertext modulus remains constant during homomorphic evaluation. The plaintext and the ciphertext spaces are defined over two polynomial rings denoted by $\mathcal{P} = \mathcal{R}_t$, and $\mathcal{C} = \mathcal{R}_q \times \mathcal{R}_q$. Unlike the BGV scheme, in B/FV, the plaintext is placed on the most significant bits of the data structure. This is achieved by utilizing a scale factor, denoted as Δ, and adjusting the message's scale before encryption and after decryption. The B/FV is outlined in Definition 3.3.

Def. 3.3: B/FV

1. $\texttt{ScaleUp}\left(m, \Delta = \left\lfloor \frac{q}{t} \right\rfloor\right)$ takes as input the message m and scale factor Δ and outputs $m' = \Delta \cdot m$;
2. $\texttt{ScaleDown}\left(m', \Delta^{-1} = \left\lfloor \frac{t}{q} \right\rfloor\right)$ takes the scaled-up message m' and reverse scale factor Δ^{-1}, and outputs $m = \Delta^{-1} \cdot m'$.
3. $\texttt{KeyGen}\left(1^\lambda\right)$ outputs $(\texttt{sk}, \texttt{pk}, \texttt{evk})$;
4. $\texttt{Enc}\left(\texttt{pk}, m'\right)$ outputs c;
5. $\texttt{EvalAdd}\left(c^{(1)}, c^{(2)}\right)$ outputs $c^{\texttt{add}}$;
6. $\texttt{EvalMult}\left(c^{(1)}, c^{(2)}\right)$ outputs $c^{\texttt{mult}}$;
7. $\texttt{Relinearize}\left(\texttt{evk}, \bar{c}\right)$ outputs c;
8. $\texttt{Dec}\left(\texttt{sk}, c\right)$ outputs m', all are similar to BGV.

3.3 CKKS

CKKS is an SHE scheme permitting approximate computation on vectors of real and complex numbers [DM15]. CKKS operates as a scale-dependent HE scheme, where the ciphertext modulus adapts throughout homomorphic evaluation, akin to the BGV scheme. The plaintext and ciphertext spaces are defined in the same way as $\mathcal{P} = \mathcal{R}$ and $\mathcal{C} = \mathcal{R}_q^2$.

Encoding. We consider a message as a vector of complex numbers $\mathbf{z} \in \mathbb{C}^{n/2}$ and provide an encoding to convert $\mathbf{z}$ into a suitable plaintext. This method relies on a scaling factor Δ and the canonical embedding $\pi : \mathbb{R}^n \to \mathbb{C}^{n/2}$. This construction is detailed in [CKKS17].

Def. 3.4: CKKS

1. $\texttt{Encode}\left(\mathbf{z}, \Delta\right)$ takes as input a vector $\mathbf{z} \in \mathbb{C}^n$ and the scale factor Δ. Maps $\mathbf{z}$ into element $w \in \mathcal{R}$, where $w \leftarrow \left\lfloor \Delta \cdot \pi^{-1}(\mathbf{z}) \right\rceil$; outputs w;
2. $\texttt{Decode}\left(w, \Delta\right)$ takes a plaintext ring element $w \in \mathcal{R}$, and computes $\mathbf{z} \leftarrow \pi\left(\Delta^{-1} \cdot w\right)$; outputs $\mathbf{z}$;
3. $\texttt{Rescale}\left(c_{q_\ell}, \Delta\right)$ scale the input ciphertext c down by Δ such that $c'_{q_{\ell-1}} = \Delta^{-1} \cdot \left[(c_1, c_2)\right]_{q_\ell}$, similar to BGV $\texttt{ModSwitch}$, outputs $c'_{q_{\ell-1}}$;
4. $\texttt{KeyGen}\left(1^\lambda\right)$ outputs $(\texttt{sk}, \texttt{pk}, \texttt{evk})$;
5. $\texttt{Enc}\left(\texttt{pk}, m\right)$ outputs c;
6. $\texttt{EvalAdd}\left(c^{(1)}, c^{(2)}\right)$ outputs $c^{\texttt{add}}$;
7. $\texttt{EvalMult}\left(c^{(1)}, c^{(2)}\right)$ outputs $c^{\texttt{mult}}$;
8. $\texttt{Relinearize}\left(\texttt{evk}, \bar{c}\right)$ outputs c;
9. $\texttt{Dec}\left(\texttt{sk}, c\right)$ outputs m', all similar to BGV.

3.4 TFHE

TFHE is an FHE scheme founded on bootstrapping. Unlike the *leveled* HE schemes, TFHE is *fully* homomorphic, enabling support for any logical circuit and, consequently, any function. TFHE is particular because it uses two types of encryption: the classical LWE ciphertext and the so-called General-GSW ciphertext. The latter permits efficient homomorphic multiplications.

Bootstrapping. To define a proper cryptosystem, one has to guarantee that any ciphertext can be decrypted correctly. As homomorphic operations tend to increase noise, it is necessary to provide a method to keep the noise of a ciphertext under a given bound. This can be achieved with a technique known as bootstrapping, which consists in *refreshing* the encryption of a message into a new ciphertext, hence resetting the noise to an acceptable level. We refer the reader to the article of A. Al Bawadi and Y. Polyakov [ABP23] for more information on bootstrapping.

Let $\mathbb{B} = \{-1, 0, 1\}$, and $\mathbb{T}$ be the real Torus, *i.e.* the set of real numbers modulo one. For $q, N > 0$ two powers of 2, we denote $\mathbb{T}_N[X] = \mathbb{T}_N[X]/(X^N + 1)$, $\mathbb{B}_N[X] = \mathbb{B}[X]/(X^N + 1)$ and $\mathbb{Z}_{q,N}[X] = \mathbb{Z}_q[X]/(X^N + 1)$. Let $\beta > 0$ and $d \in \mathbb{Z}$ be two integers, called the base and the depth, respectively.

Def. 3.5: TFHE

1. $\texttt{KeyGen}\left(1^\lambda\right)$ takes as input a security parameter λ. Sample $k = k(\lambda)$ polynomials $\texttt{sk} := (s_1, \ldots, s_k) \xleftarrow{\$} (\mathbb{B}_{q,N}[X])^k$. Set $\texttt{pk} := (\mathbf{a}, b) \leftarrow \texttt{GLWE}_{\texttt{sk}}(\mathbf{0})$, outputs the keys $(\texttt{pk}, \texttt{sk})$.
2. $\texttt{Enc}(\texttt{pk}, m)$ takes as input the public key $\texttt{pk}$ and a message $m \in \mathbb{Z}_{q,N}[X]$. Compute $\mathbf{c} \leftarrow (\mathbf{a}, b + m + e) = \texttt{GLWE}_{\texttt{sk}}(m) \in (\mathbb{Z}_{q,N}[X])^{k+1}$, for $e \xleftarrow{\chi} \mathbb{Z}_{q,N}[X]$ and outputs $\mathbf{c}$;
3. $\texttt{EvalMult}(\mathbf{G}, \mathbf{c})$ takes as input a GGSW ciphertext $\mathbf{G}$ and a GLWE ciphertext $\mathbf{c}$. First compute $(\mathbf{c})_\beta \leftarrow ((a_1)_\beta, \ldots, (a_k)_\beta, (b)_\beta) \in \mathbb{Z}_{q,N}[X]^{d(k+1)}$ where $(\cdot)_\beta$ is the decomposition in basis β. Compute the GLWE ciphertext $\mathbf{c}^{\texttt{mult}} \leftarrow (\mathbf{c})_\beta \cdot \mathbf{G}$; outputs $\mathbf{c}^{\texttt{mult}}$;
4. $\texttt{KeySwitch}(\texttt{sk}', \mathbf{c})$ takes as input a new secret key $\texttt{sk}'$ and a GLWE ciphertext $\mathbf{c} = (\mathbf{a}, b)$. Define the key-switching key $\mathbf{K} \in \mathbb{Z}_{q,N}[X]^{d \cdot k \times (k+1)}$ as the first $d \cdot k$ rows of $\texttt{GGSW}_{\texttt{sk}'}(1)$. Compute the GLWE ciphertext $\mathbf{c}' \leftarrow (\mathbf{0}, b) - \texttt{EvalMult}(\mathbf{K}, \mathbf{a})$ encrypted under $\texttt{sk}'$; outputs $\mathbf{c}'$;
5. $\texttt{CMux}(\mathbf{B}, \mathbf{c_0}, \mathbf{c_1})$ takes as input two GLWE ciphertexts $\mathbf{c_0}, \mathbf{c_1}$ of plaintexts m_0, m_1 and a GGSW ciphertext $\mathbf{B}$ of a bit b. Compute the GLWE ciphertext $\mathbf{c_3} \leftarrow \texttt{EvalMult}(\mathbf{B}, \mathbf{c_1} - \mathbf{c_0}) + \mathbf{c_0}$, which is an encryption of m_b with fresh noise; outputs $\mathbf{c_3}$;
6. $\texttt{Dec}(\texttt{sk}, \mathbf{c})$ takes as input the secret key $\texttt{sk}$ and a ciphertext $\mathbf{c}$. Compute $b - \sum_{i=1}^{n} a_i \cdot s_i = m + e$. Then, round to the nearest integer to output the message m.

Mark that a GLWE ciphertext mentioned in Definition 3.5 is an LWE ciphertext if $N = 1$ and an RLWE ciphertext if $k = 1$.

Classical methods to achieve bootstrapping are usually too slow and, thus, impractical. TFHE introduces a novel idea that consists of running the bootstrapping together with the evaluation of a look-up table. Given as input $\mathbf{G}$, a GGSW encryption of an NLUT L, and $\mathbf{c}$ the LWE encryption of a message m, one can compute directly a fresh ciphertext $\mathbf{c}'$ of $L[m]$. This method requires a heavy usage of the $\texttt{CMux}$ algorithm that can be optimized using *programmable bootstrapping.* We refer the reader to the articles of Chillotti *et al.* [CGGI20] and Cosseron *et al.* [CHMS22]

4 Hybrid Homomorphic Encryption

In this section, we introduce a novel definition for HHE schemes, while in Section 6, we use it to define HHE semantic security. Additionally, we conduct a literature analysis on the diverse techniques and design approaches employed in creating HHE schemes. Our study is divided into two parts: a brief history of HE-friendly symmetric ciphers and an examination of state-of-the-art HE-friendly ciphers and their characteristics. Within the context of HE, high computational costs and significant ciphertext expansion are two major challenges real-world applications face.

To tackle the mentioned issues, researchers in [NLV11] proposed a hybrid approach, where the plaintext m is symmetrically encrypted using a randomly chosen key $\mathtt{K}$ by a client. The resulting ciphertext c_m is much smaller than a homomorphic ciphertext, with an encryption ratio of $|c_m|/|m| \approx 1$. The client then sends c_m along with homomorphically encrypted $\mathtt{K}$ to a remote location, such as a cloud service provider. Here, c_m is *transformed* into a homomorphic ciphertext c^{evl} by evaluating the SKE decryption circuit.

With this in mind, we provide a formal definition of HHE that takes up the construction provided by Dobraunig *et al.* [DGH+23]. At this point, it is important to highlight that the definition of their proposed encryption algorithm was misleading, rendering impossible a universal definition and proper security analysis. Therefore, we extend the definition of HHE with an *encapsulation* algorithm ($\mathtt{Encap}$). This algorithm specifically manages the generation of the symmetric key and its encryption into a homomorphic ciphertext.

Let $\mathtt{HE} := (\mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Dec}, \mathtt{Eval})$ be a homomorphic encryption scheme and $\mathtt{SKE} := (\mathtt{Gen}, \mathtt{Enc}, \mathtt{Dec})$ be a symmetric cipher. A hybrid homomorphic encryption scheme is a tuple of the six algorithms $\mathtt{HHE} := (\mathtt{KeyGen}, \mathtt{Encap}, \mathtt{Enc}, \mathtt{Decomp}, \mathtt{Eval}, \mathtt{Dec})$ as follows:

Def. 4.1: Universal HHE Definition

1. $\mathtt{KeyGen}\left(1^\lambda\right)$ takes as input a security parameter λ and generates the homomorphic keys $(\mathtt{pk}, \mathtt{sk}, \mathtt{evk}) \leftarrow \mathtt{HE.KeyGen}\left(1^\lambda\right)$, outputs $(\mathtt{pk}, \mathtt{sk}, \mathtt{evk})$;
2. $\mathtt{Encap}(\mathtt{pk}, 1^\mu)$ takes as input the public key $\mathtt{pk}$, a security parameter μ and computes the symmetric key $\mathtt{K} \leftarrow \mathtt{SKE.Gen}(1^\mu)$ and its homomorphic encryption $c_{\mathtt{K}} \leftarrow \mathtt{HE.Enc}(\mathtt{pk}, \mathtt{K})$, outputs $(\mathtt{K}, c_{\mathtt{K}})$;
3. $\mathtt{Enc}(\mathtt{K}, \mathbf{m})$ takes as input, the symmetric key $\mathtt{K}$ and a message $\mathbf{m}$ and computes the ciphertext $\mathbf{c_m} \leftarrow \mathtt{SKE.Enc}(\mathtt{K}, \mathbf{m})$, outputs $\mathbf{c_m}$;
4. $\mathtt{Decomp}(\mathtt{evk}, c_{\mathtt{K}}, \mathbf{c_m})$ takes as input the evaluation key $\mathtt{evk}$, the homomorphically encrypted symmetric key $c_{\mathtt{K}}$ and a symmetric ciphertext $\mathbf{c_m}$, computes $\mathbf{c_m^{evl}} \leftarrow \mathtt{HE.Eval}\Big(\mathtt{evk}, \mathtt{SKE.Dec}, (c_{\mathtt{K}}, \mathbf{c_m})\Big)$, outputs $\mathbf{c_m^{evl}}$;
5. $\mathtt{Eval}\left(\mathtt{evk}, f, \left(c^{evl}_{m_1}, \ldots, c^{evl}_{m_n}\right)\right)$ takes as input the evaluation key $\mathtt{evk}$, a function f defined in the ciphertext space of $\mathtt{HE}$ and a n-tuple of homomorphic ciphertexts $c^{evl}_{m_i}$ and computes $\mathbf{c}^f \leftarrow \mathtt{HE.Eval}\left(\mathtt{evk}, f, \left(c^{evl}_{m_1}, \ldots, c^{evl}_{m_n}\right)\right)$, outputs $\mathbf{c}^f$;
6. $\mathtt{Dec}\left(\mathtt{sk}, \mathbf{c}^f\right)$ takes as input the homomorphic secret key $\mathtt{sk}$ and a ciphertext $\mathbf{c}^f$ and computes $\mathbf{m}^f \leftarrow \mathtt{HE.Dec}\left(\mathtt{sk}, \mathbf{c}^f\right)$, if the message $\mathbf{m}^f$ is valid, outputs $\mathbf{m}^f$; otherwise outputs $\perp$.

Correctness: Additionally, an $\mathtt{HHE}$ scheme is correct if:

$$Pr\left[\mathtt{Dec}\left(\mathtt{sk},\mathbf{c}^{f}\right)\neq f\left(\mathbf{m}\right)\mid\left[\left(\mathtt{pk},\mathtt{sk},\mathtt{evk}\right)\leftarrow\mathtt{KeyGen}\left(1^{\lambda}\right)\right]\wedge\left[\mathbf{c}^{f}\leftarrow\mathtt{Eval}\left(\mathtt{evk},f,\mathbf{c}_{\mathbf{m}}^{\mathbf{evl}}\right)\right]\right.$$

$$\left.\wedge\left[\mathbf{c}_{\mathbf{m}}^{\mathbf{evl}}\leftarrow\mathtt{Decomp}\left(\mathtt{evk},c_{\mathtt{K}},\mathbf{c}_{\mathbf{m}}\right)\right]\wedge\left[\left(\mathtt{K},c_{\mathtt{K}}\right)\leftarrow\mathtt{Encap}\left(\mathtt{pk},1^{\mu}\right)\right]\right]=negl\left(\lambda,\mu\right).$$

The correctness of $\mathtt{HHE}$ is achieved if both $\mathtt{HE}$ and $\mathtt{SKE}$ are correct.

4.1 Standard Symmetric Ciphers

Initially, homomorphic evaluation of standard symmetric ciphers such as **AES** [DR05, JV02] has received considerable attention [GHS12, BHKR13, CCK+13, CLT14]. Yet, the early implementations of homomorphic AES faced significant performance drawbacks. For example, the work in [GHS12] by Gentry *et al.*, utilizing the BGV scheme, achieved a running time of 5 minutes in Byte mode and 40 minutes in SIMD mode per block. Additionally, researchers experimented with other ciphers, such as **PRINCE** [BCG+12], which yielded an evaluation time of 3.3 seconds per block [DSES14]. The performance drawbacks of these schemes are due to the extensive multiplicative depth in their circuit. Pointing it out, authors in [ARS+15] indicated the need for new symmetric *HE-friendly Ciphers* with a lower multiplication complexity and smaller multiplicative depth while maintaining the same level of security. Early implementation of their proposed HE-friendly cipher **LowMC** had an evaluation time of 0.36 seconds per block, outperforming its counterparts by orders of magnitude. Eventually, it divided research in the HHE field into two branches, both with a focus on real-world applications: *(a)* Improving the performance of traditional ciphers such as AES to be more practical, and *(b)* Designing new HE-friendly ciphers and employing them in combination with different HE schemes for various applications. **In this paper, we primarily focus on the HE-friendly ciphers**. However, we refer readers interested in the first branch to [ADE+23], an efficient HHE scheme to perform AES over CKKS, where the authors provided a comprehensive comparison with the previous AES-based HHE schemes. Likewise, works in [TCBS23, WWL+23] attempted to enhance the computation costs of running AES using the TFHE scheme.

4.2 HE-friendly Symmetric Ciphers

As depicted in Figure 2, many HHE schemes have been proposed in recent years, most of which are based on one of two main design approaches for HE-friendly ciphers: *(1)* SPN-based ciphers (mainly utilizing s-boxes and matrix multiplication) and *(2)* Register-based stream ciphers (employing filter permutation functions). We briefly introduce these two design approaches and their historical improvements by studying the first generation of HHE schemes. We then provide more detail for their successors.

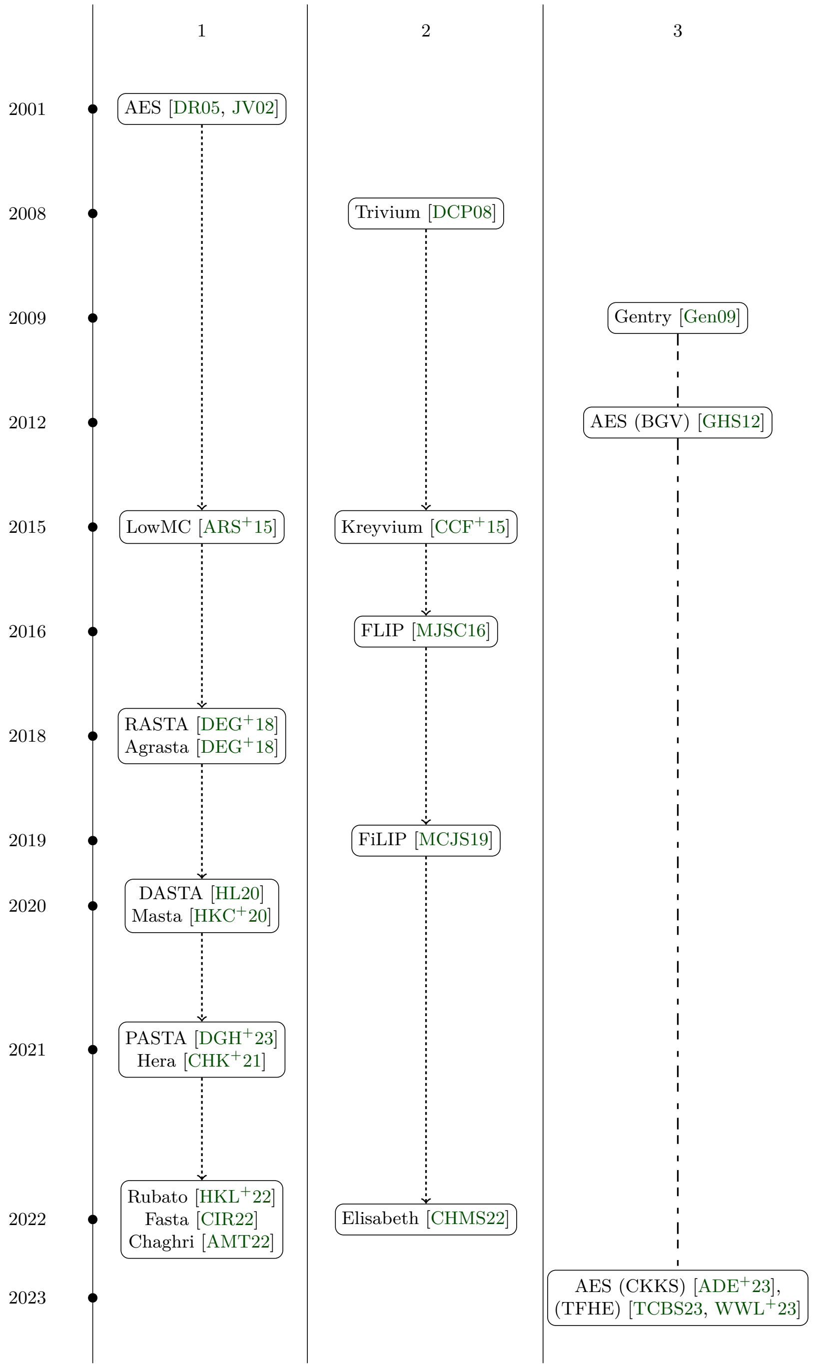

Figure 2: Timeline of HHE Evolutions **1: SPN-based ciphers, 2: Register-based ciphers, 3: Homomorphic AES**

SPN-based approach. In 2015, **LowMC** [ARS+15], the first HE-friendly cipher, was introduced. LowMC is designed to minimize the number of nonlinear operations by using efficient s-boxes while depending on a robust linear layer to ensure its security. Subsequently, the authors in [DEG+18] expanded on the concept of incorporating a robust linear layer, further developing the idea by proposing **Rasta** and **Agrasta**. Rasta uses a publicly chosen and fixed substitution layer. The affine layers are formed using a public *nonce* and a *counter*, ensuring that no affine layer is likely ever to be reused under a single key, which makes a large part of the computation nonce-dependent but key-independent. Agrasta is the aggressive variant whose key and block sizes are equal to the security level proposed to explore Rasta's limits. Let N, i, r, S_χ, and x be the nonce, block counter, round counter, s-box, and input vector, respectively. Let $M_{j,N,i}$, $c_{j,N,i}$ be the $n \times n$ binary matrix and round constant generated by an extendable output function (XOF) (see detail in [Dwo15]). Rasta's affine layer $A_{r,N,i}$ is as follows:

$$A_{r,N,i} = M_{j,N,i} \cdot x \oplus c_{j,N,i}$$

Given $\mathtt{K}$ and (N, i), Rasta's keystream is $\mathtt{ks}_{N,i} \leftarrow \mathtt{K} \oplus P_{N,i}(\mathtt{K})$, where

$$P_{N,i} = A_{r,N,i} \circ S_\chi \circ A_{r-1,N,i} \circ \ldots \circ S_\chi \circ A_{1,N,i} \circ S_\chi \circ A_{0,N,i}.$$

Authors in [HL20] found that using XOF to generate random matrices makes the pre-computation phase slower because of the costly restriction of checking matrix invertibility. Therefore, they designed **Dasta** – a variant of Rasta that avoids the use of XOF. Instead of randomly generated linear layers, which are random invertible binary matrices, the authors consider linear layers to be split into two parts: (1) a variable bit permutation and (2) a fixed linear transformation. Another variant of Rasta called **Masta** [HKC+20] employed modular arithmetic to support HE schemes over a non-binary plaintext space. Its advantage over Rasta lies in the reduced computational cost on the client side, which is achieved by defining affine layers with finite field multiplication, resulting in improved performance. **FASTA** [CIR22] as a variant of Rasta designed for efficient homomorphic *packed* evaluation over BGV schemes. Its linear layer, a rotation-based transformation, is combined with five parallel calls of a specific Rasta instance. Each of these instances operates with the 329-bit key and contributes to generating a portion of the keystream. **Chaghri** [AMT22], a recently proposed SPN-based cipher, follows the *Marvellous* design strategy [AABS+20]. It operates in rounds, with each round comprising three layers: S-box, linear, and subkey injection. The subkeys for injection are derived from the master key using a key schedule algorithm. In the S-box layer, a power map x^a is applied to each state element, followed by an invertible affine transformation. Regrettably, Chaghri's author only compared their work with **AES**, achieving a running time of 54.32 seconds per block compared to 97.84 seconds per block for AES. Chaghri fails to surpass other HHE schemes in performance. Consequently, we have left it out of the state-of-the-art schemes.

Register-based approach. In 2015, the authors of **Kreyvium** [CCF+15] proposed a completely different approach, relying on tailor-made stream ciphers. Kreyvium is a variant of the Trivium [DCP08] stream cipher, designed to deliver 128-bit security while preserving its performance attributes. Trivium is a stream cipher that has garnered recommendation within the eSTREAM portfolio of stream ciphers [ECR12]. The authors introduced a new HHE structure (Figure 3) consisting of two phases: the **offline** and **online** phases. The offline phase is independent of the plaintext and can be completed in advance. In contrast, the online phase is executed upon receiving the symmetric ciphertext, which depends on the plaintext input.

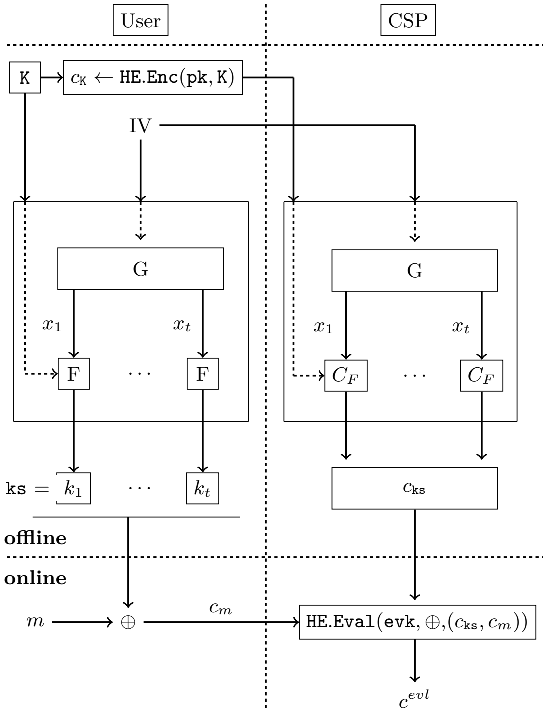

Figure 3: Kreyvium HHE framework [CCF+15]

Kreyvium's plaintext space has been considered as $\{0, 1\}$. The stream cipher is a combination of an expansion function G, which maps ℓ_{IV}-bit strings to strings of arbitrary size, and a fixed-size parametrized function F with input size ℓ_x, parameter size $\ell_{\mathtt{K}}$ and output size N. The expansion function G is a CTR mode counter defined as $G\left(IV, t \cdot \ell_x\right) = \left(IV, IV \boxplus 1, \ldots, IV \boxplus (t-1)\right)$ where $a \boxplus b = (a+b) \mod 2^{\ell_x}$. Additionally, F is designed to generate the keystream using a synchronization function that takes the IV and $\mathtt{K}$ and outputs an n-bit initial state, a transition function that computes the next state, and a filtering function that takes the internal state and computes the keystream based on that.

The authors of **FLIP** [MJSC16] proposed an alternative method with a similar objective. FLIP stream cipher incorporates a filter permutator using a forward PRNG [BY03] based on AES-128, the Knuth shuffle [D+97] bit permutation generator, and a filter function. This design achieves consistent noise reduction when integrated with HE schemes. FLIP's authors propose a boolean filtering function optimized for FHE schemes like GSW [GSW13] and FHEW [ASP14]. The authors introduced a novel framework, named Homomorphic Encryption-Symmetric Encryption (HE-SE), comprising five steps, as illustrated in Figure 4. FLIP has shown certain security weaknesses [DLR16]. In response to these vulnerabilities, the authors introduced **FiLIP** [MCJS19], an enhanced iteration of FLIP, adopting the Improved Filter Permutator (IFP) paradigm. The main advantage of IFPs is that they can apply to any filtering function and register size, providing a general framework for determining the security of IFP instances. FiLIP's authors coined the term *Transciphering* adopted widely among other HHE schemes. The homomorphic evaluation of the `SKE` cipher's decryption circuit is known as *transciphering*, and it is the most resource-intensive part of HHE frameworks. It is noteworthy that *transciphering* is often denoted by terms such as *decompression* or *decomp* in the context of HHE schemes.

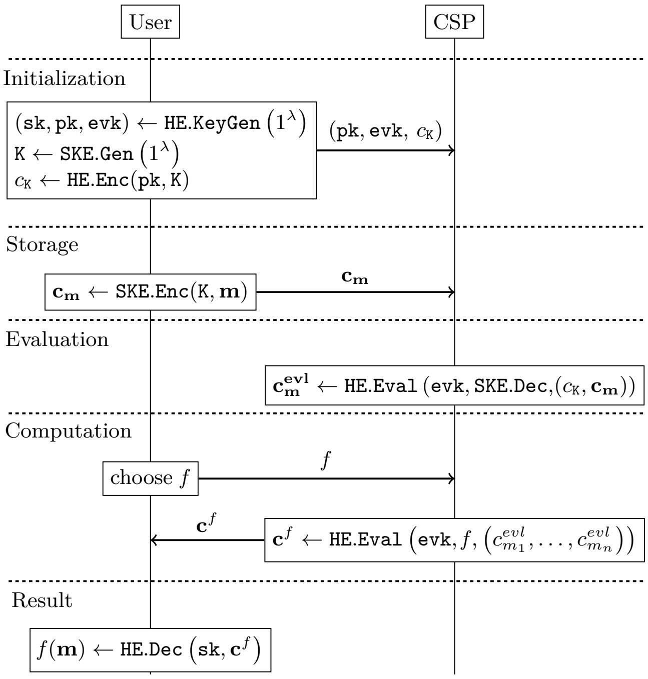

Figure 4: General HHE framework [MJSC16]

Ultimately, recent HHE design advancements have suggested schemes such as **PASTA**, **HERA**, and **Rubato**, leveraging robust linear layers and S-boxes. In addition, **Elisabeth** follows the FP design paradigm. The subsequent sections delve into the specifics of these diverse HHE schemes.

4.3 Pasta

Pasta [DGH+23] is a new stream cipher optimized for integer use cases over $\mathbb{F}_p$. Pasta's authors provide an extensive comparison of different existing symmetric ciphers in the context of HHE spanning several libraries. Their results show that Pasta achieves a better balance between ciphertext expansion and computational efficiency compared to existing symmetric ciphers. This better balance is accomplished by combining efficient techniques for the S-boxes and linear layers specifically tailored for integer arithmetic over $\mathbb{F}_p$. The design of PASTA is based on the Rasta design strategy, namely splitting the cipher into two parallel branches to optimize the linear layer. Taking $\texttt{K}$ as an input, Pasta's permutation is as follows:

$$P_{N,i} = A_{r,N,i} \circ S_c \circ A_{r-1,N,i} \circ S_{fq} \circ A_{r-2,N,i} \circ \ldots \circ A_{1,N,i} \circ S_{fq} \circ A_{0,N,i}$$

It utilizes two types of S-boxes, as previously defined in Section 2. Further, similar to Rasta, Pasta's affine layer was defined as follows:

$$A_{j,N,i} = \begin{bmatrix} 2 \cdot I & I \\ I & 2 \cdot I \end{bmatrix} \begin{bmatrix} M_{j,L,N,i}(\mathbf{x}_L) + \mathbf{c}_{j,L,N,i} \\ M_{j,R,N,i}(\mathbf{x}_R) + \mathbf{c}_{j,R,N,i} \end{bmatrix}$$

where $I, M \in \mathbb{F}_p^{t \times t}$, and $\mathbf{c} \in \mathbb{F}_p^t$ represent the identity matrix, invertible matrix, and constant for each round, respectively. It is worth noting that FASTA was introduced after Pasta; however, the authors abstained from a direct comparison between FASTA and Pasta. Instead, they asserted that FASTA's design signifies an improvement over the Rasta and Dasta schemes.

4.4 HERA

HERA [CHK+21] is an HE-friendly stream cipher with a randomized key schedule. HERA's author proposed a new transciphering framework called RtF (Real-to-Finite-field) for efficient computation over encrypted data of real numbers using HE. The framework combines the CKKS and BFV homomorphic encryption schemes and uses HERA stream cipher with modular arithmetic in between. Following the design paradigm of Kreyvium, RtF also incorporates two distinct phases for offline and online computation. HERA encrypts a real message vector $\mathbf{m} \in \mathbb{R}^n$ on the client side and converts the ciphertexts into the corresponding CKKS ciphertexts on the server side. The main idea behind the HERA cipher is to use a simple randomized key schedule to generate a set of polynomials over $\mathbb{Z}_t$ in unknowns $\{k_0, \ldots, k_{15}\}$, where $k_i \in \mathbb{Z}_t$ denotes the i-th component of the secret key $\mathtt{K} \in \mathbb{Z}_t^{16}$. Taking $\mathtt{K}$ as an input, HERA's permutation is as follows:

$$P_N = Fin_{r,N} \circ RF_{r-1,N} \circ \ldots \circ RF_{1,N} \circ KSD_{0,N},$$

where *Fin*, *RF*, and *KSD* denote the final round function, round function, and key scheduler, respectively. For each round $1 < i < r-1$, the round function is defined as

$$RF_{i,N} = KSD_{i,N} \circ S_c \circ MixRows \circ MixColumns$$

Additionally, the last round function operates on the final round r as

$$Fin_{r,N} = KSD_{r,N} \circ MixRows \circ MixColumns \circ S_c \circ MixRows \circ MixColumns$$

Finally, the key scheduler component is a product between a uniformly random value $\mathbf{rc} \in (\mathbb{Z}_t^{16})^{r+1}$ obtained from an XOF function fed by a nonce N, and $\mathtt{K}$, denoted as $KSD_{i,N}(\mathbf{x}) = \mathbf{x} + \mathtt{K} \odot \mathbf{rc}_i$. Compared to other HE-friendly ciphers, such as FLIP and Rasta, which use randomized linear layers, HERA requires fewer random bits, significantly improving its efficiency on both the client and server sides.

4.5 Rubato

Rubato [HKL+22] is a family of noisy ciphers for approximate homomorphic encryption based on HERA design. Rubato introduces noise to a symmetric cipher of low algebraic degree, significantly reducing multiplicative complexity without compromising security. Rubato operates in the same transciphering framework as RtF. It takes a symmetric key and a nonce as input and returns a keystream. The keystream is generated by applying linear and nonlinear transformations to the input key and nonce. The noise is introduced during the encryption process, resulting in a noisy cipher that is not suitable for the transciphering of exact data. Same as HERA, Rubato uses a randomized key schedule to generate a set of polynomials over $\mathbb{Z}_q$ in unknowns $\{k_0, \ldots, k_n\}$, where $k_i \in \mathbb{Z}_q$ denotes the i-th component of the secret key $\mathtt{K} \in \mathbb{Z}_q^n$. Taking $\mathtt{K}$ as an input, Rubato's permutation is as follows:

$$P_N = NF \circ Fin_{r,N} \circ RF_{r-1,N} \circ \ldots \circ RF_{1,N} \circ KSD_{0,N}$$

where *NF*, *Fin*, *RF*, and *KSD* denote the adding noise, final round, round functions, and key scheduler, respectively. For each round $1 < i < r-1$, the round function is defined as

$$RF_{i,N} = KSD_{i,N} \circ S_{fq} \circ MixRows \circ MixColumns$$

Additionally, the last round function operates on the final round r as

$$Fin_{r,N} = TR_{n,l} \circ KSD_{r,N} \circ MixRows \circ MixColumns \circ S_{fq} \circ MixRows \circ MixColumns,$$

where $TR_{n,l} : \mathbb{Z}_q^n \to \mathbb{Z}_q^l$ is the truncation function. Same as HERA, the key scheduler denoted as $KSD_{i,N}(\mathbf{x}) = \mathbf{x} + \mathtt{K} \odot \mathbf{rc}_i$, where $\mathbf{rc} \in (\mathbb{Z}_q^n)^{r+1}$. Finally, Gaussian noise is added to the output of the final round $(x_1, \ldots, x_l)$ as $NF(x) = (x_1 + e_1, \ldots, x_l + e_l)$, where $(e_1, \ldots, e_l)$ are l elements sampled from a one-dimensional discrete Gaussian distribution. For a given message vector $\mathbf{m} \in \left(\mathbb{R}^l\right)^b$, Rubato encryption is defined as $c = \lfloor \Delta \cdot \mathbf{m} \rceil + \mathbf{ks} \bmod q$, where $\mathbf{ks} \in \left(\mathbb{Z}_q^l\right)^b$ is the keystream of b-block generated by the Rubato cipher, and $\Delta \in \mathbb{R}$ is the scaling factor. Compared to Pasta and HERA, Rubato exhibits lower multiplicative depth and demands fewer random bits for linear layers.

4.6 Elisabeth

Elisabeth [CHMS22] is a stream cipher optimized for the TFHE scheme. It provides a variety of server-side operations for homomorphic evaluation, especially for neural network inference. It uses a Group Filter Permutator (GFP) paradigm. The filters in Elisabeth's design use the fewest levels possible of NLUT to parallelize computation efficiently. Taking $\mathtt{K} \in \mathbb{Z}_q^n$ as input, with q a power of two, Elisabeth's i-th keystream is $\mathtt{ks}_i = F\left(P_i\left(S_i(\mathtt{K})\right) + w_i\right)$, where $S_i : \mathbb{Z}_q^n \to \mathbb{Z}_q^k$ is a subset extraction function that picks at random k elements in $\mathbb{Z}_q^n$, P_i is a random permutation and $w_i \xleftarrow{\$} \mathbb{Z}_q^k$ is a random mask. $F : \mathbb{Z}_q^k \to \mathbb{Z}_q$ is the filter function, defined as:

$$F(x_1, \ldots, x_k) = \bigoplus_{i=1}^{k/t} f(x_1, \ldots, x_t)$$

with $\bigoplus$ the direct sum over $\mathbb{Z}_q$, t a divisor of k, and $f : \mathbb{Z}_q^t \mapsto \mathbb{Z}_q$ a sequence of additions and NLUT evaluations. The choice of t and the definition of f are key parameters towards the optimization of Elisabeth stream cipher. We refer the reader to the complete analysis provided by Cosseron *et al.* [CHMS22] for more detail on the NLUT choices and the construction of f.

5 Evaluation

We commence by examining the specifications of various HHE ciphers to provide a comprehensive overview of their distinct properties. Our analysis summarizes the implementation of each HHE scheme based on the authors' claims and their support for open science. Subsequently, we assess the performance of three cutting-edge HHE schemes: PASTA, HERA, and Rubato, in a real-world setting, employing our implementation. The source codes for both the client and server sides are composed in Golang version 1.21.5, leveraging the Lattigo library [lat23] version 5.0.2. This library presently supports B/FV, BGV, and CKKS schemes. Our experiments were conducted in a single-threaded environment, with the client side operating on a laptop equipped with an Intel Core i5-9300H CPU @ 2.40GHz and 16GB memory and the server side on a PC powered by an Intel Core i7-8700 CPU @ 3.20GHz with 64GB memory.

5.1 Implementations of HHE Ciphers.

As depicted in Table 1, various HHE ciphers have been proposed, each with different properties. Most of these ciphers are designed to operate with one or more HE schemes, depending on the plaintext space and data type. To incorporate these HHE ciphers into an HHE framework, it is necessary to implement the decryption circuit using a library that provides an Application Programming Interface (API) to support the requisite HE scheme. There are numerous libraries available for implementing HE schemes. We direct the reader to [GMT23], where the authors thoroughly analyze these diverse libraries to ascertain their respective strengths and weaknesses.

Table 1: HHE Ciphers Properties

Cipher	T[1]	Supported HE	Security (bits)	Tool	Field
LowMC	B	B/FV, BGV, TFHE	80, 128, 256	S-box	$\mathbb{F}_2$
Kreyvium	S	B/FV, BGV, TFHE	128	FP	$\mathbb{F}_2$
FLIP	S	B/FV, BGV, TFHE	80, 128	FP	$\mathbb{F}_2$
R & A [2]	S	B/FV, BGV, TFHE	80, 128, 256	S-box	$\mathbb{F}_2$
FiLIP	S	B/FV, BGV, TFHE	80, 128	FP	$\mathbb{F}_2$
Dasta	S	B/FV, BGV, TFHE	80, 128, 256	S-box	$\mathbb{F}_2$
Masta	S	B/FV, BGV	80, 128	S-box	$\mathbb{F}_p$
PASTA	S	B/FV, BGV	80, 128	S-box	$\mathbb{F}_p$
FASTA	S	B/FV, BGV	128	S-box	$\mathbb{F}_2$
HERA	S	CKKS	80, 128, 256	S-box	$\mathbb{F}_p$
Rubato	S	CKKS	80, 128	S-box	$\mathbb{F}_p$
Elisabeth	S	TFHE	128	FP	$\mathbb{F}_{16}$, $\mathbb{F}_2$
Chaghri	B	BGV	128	S-box	$\mathbb{F}_2$

[1] **T** denotes cipher type, S: Stream cipher and B: Block cipher
[2] Rasta and Agrasta

We have examined the available implementations for various HHE schemes to comprehend the proposed HHE framework and ensure the fairness of their experiments. Our findings, which are presented in Table 2, illustrate the availability of various cipher implementations that support open science, and it also compares these ciphers with their counterparts. As Table 2 illustrates, recent schemes such as PASTA, HERA, and Elisabeth are accompanied by open-source implementations, enabling comparisons with numerous other HHE schemes.
The authors of PASTA provide a comprehensive framework named *hybrid-HE- framework* [Hyb21], which includes implementations for comparing eight ciphers across three different libraries. The authors of HERA implemented their *RtF-Transciphering* framework [RtF21] using the Lattigo library, providing implementations only for HERA and Rubato. However, they claim to compute comparison metrics with other works directly from Dasta [HL20], which is *not* implemented in the same programming language, leading to an *unfair comparison.* The authors of Elisabeth used the Concrete Library to implement their scheme in Rust [Eli22], implementing only the FiLIP cipher and using the results from PASTA's benchmarking framework to compare their work with others.

Table 2: HHE Ciphers with Open-Source Implementation

Cipher	Comparison	Library	Language	Open-source
C1: LowMC	AES, PRINCE	HElib	C/C++	✓
C2: Kreyvium	C1, Trivium	HElib	C/C++	✗
C3: FLIP	C1, C2	HElib	C/C++	✗
C4: Rasta, Agrasta	C1, Trivium, C2, C3	HElib	C/C++	✓
C5: FiLIP	C1, C3, C4	HElib	C/C++	✗
C6: Dasta	C4	HElib	C/C++	✗
C7: Masta	C4	HElib	C/C++	✗
C8: PASTA	C1, C2, C4, C5, C6, C7, C9	HElib, SEAL, TFHE	C/C++	✓
C9: HERA	C1, C3, C4, C6, C7	Lattigo	Golang	✓
C10: Fasta	C4	HElib, TFHE, PALISADE	C/C++	✓
C11: Rubato	C7, C8, C9	Lattigo	Golang	✓
C12: Elisabeth	C1, C2, C5, C6, C7, C8, C9	Concrete	Rust	✓
C13: Chaghri	AES	HElib	C/C++	✗

✓ denotes that the implementation is open-source and available for benchmarking.
✗ denotes denotes that the implementation is not publicly available.

Our study reveals that among all these implementations, PASTA's framework covers implementations for most of the HHE schemes in the same programming language. However, their implementation has two major flaws. First, it does not separate client and server execution as in a real-world application setting. Second, it does not use large input vectors to fully analyze the performance of the HHE schemes. Elisabeth's implementation encountered a similar issue. Likewise, the author of PASTA did not implement Rubato, which is claimed to be more efficient than HERA for client-side computations.

5.2 Our HHE Benchmarking Approach

To the best of our knowledge, none of the existing HHE implementations are intended for use in real-world scenarios, with *independent* client and server implementations. Furthermore, as shown in Figure 5, past works' comparisons have always been conducted on *powerful* devices with no resource constraints. To ensure a fair and realistic comparison, our implementation includes two primary components, client and server, that can be executed *independently*. We implemented PASTA, HERA, and Rubato ciphers in Lattigo. However, as Lattigo does not yet fully support TFHE, we could *not* implement Elisabeth [CHMS22] and include it in our comparison. Yet, according to the comparison provided in [CHMS22], PASTA and HERA outperform Elisabeth regarding running time per bit for 128-bit security; therefore, the same results are expected.

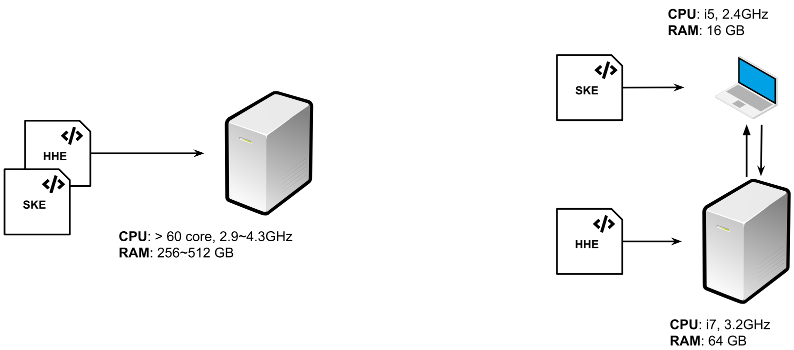

Figure 5: **Two different settings with specification for HHE implementation**: (left) non-real-world settings, (right) our real-world setting

Golang Benchmarking Tool. Golang incorporates built-in tools for crafting comprehensive benchmarks, offering statistics for each process execution. These statistics encompass the following metrics:

M1. *Average Latency* represents the average execution time per operation.

M2. *Memory allocation* indicates the total number of bytes allocated per operation on the heap.

M3. *Number of (memory) allocations* denotes the number of memory allocations required for each operation.

Memory allocation operations require CPU resources to locate the proper memory chunks. As a result, a rise in the number of memory blocks created for memory allocations corresponds to a higher CPU resource consumption. To optimize code execution performance, efforts should be made during the scheme's implementation to reduce excessive memory allocations [PŁ22]. Thus, measuring such metrics is critical for developing new memory-efficient HHE schemes.

We utilized the standard Golang benchmarking tool to evaluate the mentioned metrics for HHE schemes. A lower value for each metric is preferred to achieve an efficient scheme. To ensure a fair comparison, we provided parameters in Table 3 that yield the same security level (128-bit). For simplicity, each set of parameters is named and will be used to present results. Furthermore, we opted to generate random plaintext vectors of size N, where N denotes the maximum number of coefficients, also referred to as the maximum number of slots for plaintext and ciphertext, corresponding to the HE parameters.

Table 3: Benchmarking Parameters

Param	#Rounds	#Key (words)	#Blocks (words)	$\log_2$ P	$\log_2$ N
PASTA					
P3-1614	3	256	128	16	14
P3-3215	3	256	128	32	15
P3-6015	3	256	128	60	15
P4-1614	4	64	32	16	14
P4-3215	4	64	32	32	15
P4-6016	4	64	32	60	16
HERA					
H5-2816	5	16	16	28	16
H5-2516	5	16	16	25	16
Rubato					
R5-2616	5	16	12	26	16
R3-2516	3	36	32	25	16
R2-2516	2	64	60	25	16

In our implementation, the execution flow of HHE schemes is divided into five functions: `KeyGen`, `Encap`, `Enc`, `Decomp`, and *Relinearization* and *Rotation* `(R&R) KeyGen` (also known as halfboot keys in HERA and Rubato schemes).
The first four functions are explained in Definition 4.1. In principle, the `R&R KeyGen` is considered part of `KeyGen`; however, in practice, it operates on the *server-side* to produce the requisite keys for `Decomp`, due to its high resource consumption. The client uses `KeyGen`, `Encap`, and `Enc` to generate HE keys, then creates a master symmetric key `K` and encrypts it with the respected HE scheme. She then encrypts her data using `K`. Upon receiving encrypted data and homomorphically encrypted $c_{\mathtt{K}}$, the server uses the `R&R KeyGen` and `Decomp` functions to **transcipher** symmetrically encrypted data into homomorphic data. This approach allows the server to perform HE operations on the data.

Table 4: Results for Symmetric Ciphers

Param	Time (ms/op)	Memory Allocs (MB/op)	#Allocs (allocs/op)
P3-1614	35.89223	17.64820862	859835
P3-3215	42.30898	21.32488155	860274
P3-6015	49.06042	27.65648174	860290
P4-1614	2.910659	1.417246819	69212
P4-3215	3.405685	1.698161125	69222
P4-6016	3.964417	2.198654175	69257
H5-2816	0.075601	0.01574707	858
H5-2516	0.084632	0.01574707	858
R5-2616	0.143319	0.049766541	1229
R3-2516	0.084013	0.026374817	691
R2-2516	0.052809	0.017601013	483

Client-Side: The results of the client-side experiment, including *keystream generation* and *encrypting* a vector of 128 plaintexts, are presented in Table 4. The results indicate that Rubato outperforms HERA, PASTA-4, and PASTA-3 by an average factor of 1.2×, 42×, and 500×, respectively, regarding running time per operation. Furthermore, Rubato surpasses PASTA-4 and PASTA-3 by an average factor of 710× and 56×, respectively, regarding memory allocation per operation. However, it should be noted that Rubato consumes more memory per operation (20×) than HERA while having a better number of allocations per operation for R3-2516 and R2-2516.

<u>*Server-Side:*</u> The results of server-side experiments for PASTA, HERA, and Rubato are summarized in Table 5, aligning with the metrics used on the client side. The results reveal that for `R&R KeyGen`, PASTA-3 exceeds PASTA-4, HERA, and Rubato by an average factor of 3.1×, 5.2×, and 5.4× in regards to running time. Likewise, PASTA-3 outperforms PASTA-4, Rubato, and HERA by an average of 3×, 7.8×, and 7.9×, respectively, regarding memory allocation. However, Rubato surpasses HERA, PASTA-3, and PASTA-4 by an average factor of 1.1×, 15.7×, and 29.4×, respectively, in terms of running time. HERA outperforms Rubato, PASTA-3, and PASTA-4 by an average of 1.5×, 29.9×, and 35.6× in memory allocation, respectively. Experiments on the average execution time and memory consumption for performing `Decomp` indicate that Rubato and HERA schemes surpass PASTA by a significant order of magnitude. The advantage is ascribed to their low multiplicative depth for the decryption circuit and the data encoding techniques, resulting in using all coefficients in the ciphertext polynomial rings. Utilizing CKKS encoding techniques and transferring data between coefficients and slots for decoding enhances the efficiency of evaluating the symmetric circuit – widely regarded as the most computationally expensive component of HHE schemes – for HERA and Rubato.

Table 5: Benchmark results for HHE schemes in full-coefficient mode

Param	Benchmark	Time (s/op)	Memory Allocs (MB/op)	#Allocs (allocs/op)
P3-1614	KeyGen	0.0397791	51.08751678	21873
	R&R KeyGen	7.287907	1808.771645	71191
	Encap	0.0265839	4.968841553	243
	Enc	3.7912274	2258.974701	110063048
	Decomp	780.66732	384619.065	91051940
P3-3215	KeyGen	0.0785246	101.573555	21877
	R&R KeyGen	27.1918257	6608.258629	99038
	Encap	0.0560374	9.921966553	243
	Enc	8.9038207	5453.794571	220208763
	Decomp	3269.26386	1532179.02	181687981
P3-6015	KeyGen	0.0793637	101.5717239	21872
	R&R KeyGen	27.0965226	6608.259872	99042
	Encap	0.0557442	9.921966553	243
	Enc	10.4351284	7079.774445	220202696
	Decomp	3278.80850	1533410.287	181062228
P4-1614	KeyGen	0.041437	51.08769989	21875
	R&R KeyGen	26.4933591	6206.03894	152984
	Encap	0.0251192	4.968841553	243
	Enc	1.2451438	725.6864471	35442042
	Decomp	1747.36410	550370.2523	55599409
P4-3215	KeyGen	0.0791121	101.5734787	21876
	R&R KeyGen	101.6135308	24378.86888	264367
	Encap	0.0532069	9.921966553	243
	Enc	2.8978117	1743.103241	70890711
	Decomp	7371.90987	2195520.681	110678564
H5-2816	KeyGen	1.1824637	1570.83329	375771
	R&R KeyGen	106.4032398	40065.93994	140442
	Encap	9.2174537	3818.448524	8531
	Enc	0.56478	125.0019531	6881377
	Decomp	179.68644	40091.24128	6509545
H5-2516	KeyGen	1.0762753	1427.113785	390495
	R&R KeyGen	110.6379293	39143.16187	136406
	Encap	8.6461869	3610.431923	8115
	Enc	0.57781	125.0019531	6881377
	Decomp	165.19297	36903.18815	6502104
R5-2616	KeyGen	1.0813324	1427.113632	390477
	R&R KeyGen	111.3192927	39143.16061	136400
	Encap	9.2465584	3610.431946	8116
	Enc	0.81869	324.5014648	7929929
	Decomp	91.76728	25918.47347	6452755
R3-2516	KeyGen	1.0676216	1427.111946	390487
	R&R KeyGen	110.9901145	39143.16535	136418
	Encap	20.9380585	8085.594765	18161
	Enc	1.24966	415.5039139	10682562
	Decomp	148.61062	56542.57570	9692901
R2-2516	KeyGen	1.0687663	1427.111885	390482
	R&R KeyGen	111.11005	39143.15746	136383
	Encap	36.6681119	14341.22204	32208
	Enc	1.66919	512.5073242	13631849
	Decomp	224.21408	98637.28681	12964543

Overall Findings. Our comparison concentrated on memory allocation and execution time metrics for both client and server. Below is a summary of our key findings:

F1. Utilizing the **B/FV** scheme, PASTA variations perform faster and need less memory than HERA and Rubato variations for both `KeyGen` and `Encap`.

F2. In `Enc`, HERA marginally outperforms Rubato, while both surpass PASTA regarding execution time and memory utilization.

F3. The `R&R KeyGen` function appears to be more memory-intensive and time-consuming in HERA and Rubato than in PASTA.

F4. Rubato and HERA's `Decomp` function surpasses PASTA in execution time and memory consumption by a significant order of magnitude, respectively. The advantage is ascribed to their low multiplicative depth for the decryption circuit and data encoding techniques, mainly when the **CKKS** is employed as the underlying HE scheme.

F5. We discovered an out-of-memory problem with **P4-6016**, so, our analysis excludes the server-side results for P4-6016. These findings underscore a certain level of impracticality. More precisely, despite being on the server side and assuming unlimited resources, it is essential to acknowledge that these resources are paid for. Therefore, running these experiments in the cloud would incur significant costs, rendering it financially unsustainable for many users.

Discussion on the choice of HE library. When we started our study, selecting an appropriate programming language for our implementations was a key decision. Given the available libraries, we identified two main options: OpenFHE (C++) [ABBB+22] and Lattigo (Go) [lat23]. Eventually, we chose Lattigo due to three main reasons:

D1. Compared to C++, Go allows developers to design and develop applications rapidly and cost-effectively using modern software development methods. Furthermore, Go offers a more flexible maintenance process – a characteristic that is of paramount importance for industries wishing to build HHE services.

D2. Go supports multi-platform deployment and execution due to its building capabilities. Developers can readily deploy Go and Lattigo-based applications using Docker.

D3. Two primary schemes of our study – HERA and Rubato – were already implemented in Lattigo (not in a client-server setting). Hence, we already have a good starting point for our library. Therefore, we decided to implement PASTA, HERA, and Rubato using our modular approach and migrate to the necessary dependencies. Even though Lattigo is a promising library for HE, it does not have built-in support for the TFHE scheme. However, it is currently under continuous development, and TFHE support is already planned. This will allow us to add the missing implementation for the Elizabeth scheme.

Complexity. Initially, we intended to augment our experiments with a comprehensive comparison of the complexity of the examined schemes. We believed this would provide an additional, insightful metric for evaluating the efficiency of various HHE schemes. However, upon commencing the complexity analysis, we determined that a practical assessment is the only pertinent method to evaluate scheme efficiency (due to space constraints, further elaboration on this point is not feasible here). This observation may also explain why such an analysis is often *absent* in many HHE papers.

Open Science and Reproducible Research. To support open science and reproducible research and provide other researchers with the opportunity to use, test, and hopefully extend our implementation, the source code used for our evaluations has been made available online[1].

[1] https://github.com/hosseinabdinf/HHELand

6 Security

This section provides a detailed and formal outline of the semantic security required for an HHE scheme to be fulfilled. We then examine diverse attack strategies employed to compromise HE-friendly symmetric ciphers. Notably, our analysis underscores that despite meeting established security criteria, these ciphers remain susceptible to various attacks, posing a potential threat to the security of the corresponding HHE scheme.

6.1 Security definitions

This subsection is a background for the definition of HHE security given in Definition 7. We try to capture the concept of perfect secrecy introduced by C. Shannon in 1946, stating that the ciphertext gives the adversary no information about the underlying message. This notion is well-defined for traditional PKE schemes, but there is still no clear consensus on its definition for HE. In particular, a recent article from B. Li and D. Micciancio [LM21] introduced a novel notion of semantic security, namely IND-CPA^D, designed for *approximate* HE schemes, as CKKS. This new security definition is more robust and comprehensive, as it addresses several flaws of traditional IND-CPA in the case of approximate HE. However, J. H. Cheon *et al.* [CCP+24] discuss it as too strong for practical use in cryptography in the general case. For this reason, we chose to rely on the traditional definition of IND-CPA security and extend it to a general definition of security for HHE. Nonetheless, for the sake of comparison, the definition of IND-CPA^D is provided in Definition 5.

Definition 4 (Semantic Security for HE)**.** Let $\mathtt{HE} = (\mathtt{KeyGen}, \mathtt{Enc}, \mathtt{Eval}, \mathtt{Dec})$ be an HE scheme for message space $\mathcal{M}$ and functionality space $\mathcal{F} : \mathcal{M}^n \to \mathcal{M}$. We say that $\mathtt{HE}$ is semantically secure if for all PPT adversary $\mathcal{A}$, it holds that the advantage of $\mathcal{A}$ given by $\mathbf{Adv}^{\text{ind-cpa}}_{\mathtt{HE},\mathcal{A}}(\lambda) = \left| Pr\left[\mathbf{Exp}^{\text{ind-cpa}}_{\mathtt{HE},\mathcal{A}}(\lambda) \to \text{true}\right] - \frac{1}{2}\right|$ is negligible in λ, where the experiment is defined as in Figure 6.

$\underline{\mathbf{Exp}^{\text{ind-cpa}}_{\mathtt{HE},\mathcal{A}}(\lambda)}$:	$\underline{\mathcal{O}\textsc{Encrypt}(\mathtt{pk}, m_0, m_1)}$:	$\underline{\mathcal{O}\textsc{Eval}(\mathtt{evk}, f, (c_1, \ldots, c_n))}$:
$b \xleftarrow{\$} \{0,1\}$; $L \leftarrow \emptyset$ $(\mathtt{sk}, \mathtt{pk}, \mathtt{evk}) \xleftarrow{\$} \mathtt{KeyGen}(1^\lambda)$ $b' \leftarrow \mathcal{A}(1^\lambda, \mathtt{pk}, \mathtt{evk})$ **return** $(b = b')$	If $m_0, m_1 \notin \mathcal{M}$: **return** $\perp$ $c \xleftarrow{\$} \mathtt{Enc}(\mathtt{pk}, m_b)$ $L \leftarrow L \cup \{c\}$ **return** c	If $f \notin \mathcal{F}$: **return** $\perp$ For $i \in [1, n]$ If $c_i \notin L$: **return** $\perp$ $c \leftarrow \mathtt{Eval}\,(\mathtt{evk}, f, (c_1, \ldots, c_n))$ $L \leftarrow L \cup \{c\}$ **return** c

Figure 6: Security indistinguishability game for HE. $\mathbf{Exp}^{\text{ind-cpa}}_{\mathtt{HE},\mathcal{A}}$ is the ind-cpa experiment for an HE scheme $\mathtt{HE}$ and PPT adversary $\mathcal{A}$. The two additional algorithms are the oracles accessible to $\mathcal{A}$.

This security game involves a challenger, playing the role of a user and an adversary $\mathcal{A}$ represented as a PPT algorithm. $\mathcal{A}$ is provided access to two oracles he can call a finite number of times, according to its capabilities: $\mathcal{O}$Encrypt, which, on the input of two plaintexts chosen by the adversary, outputs the encryption of one of them depending on the bit b randomly chosen at the beginning of the game, and $\mathcal{O}$Eval, which runs the evaluation algorithm for a valid function f and a tuple of ciphertext known by the adversary. This game maintains a record of the ciphertexts known by the adversary through a list L. Hence, every call of an oracle expands the list L with a new ciphertext. The adversary's goal is to guess the bit b, and we consider that he wins if he can find b with a probability significantly different than a random guess, that is $1/2$.

IND-CPA^D Security. As mentioned in Section 6.1, recent works state that IND-CPA is insufficient to capture the semantic security of HE. On the initiative of Li and Micciancio [LM21], the notion of IND-CPA^D was first defined in 2020 and has ever since raised many discussions and disagreements in the community. The name IND-CPA^D stands for Indistinguishability under Chosen Plaintext Attack *with a Decryption oracle*, as the only but crucial difference with traditional IND-CPA is the introduction of a particular decryption oracle.

Throughout this game, the list L does not only keep records of a ciphertext c, but stores every query as a tuple (m_0, m_1, c_β), where m_0, m_1 are the plaintexts sent by $\mathcal{A}$, and c_β the ciphertext output by the encryption oracle $\mathcal{O}$Encrypt, depending on the bit β. Concerning the evaluation oracle $\mathcal{O}$Eval for a function f and ciphertexts $(m_0^1, m_1^1, c_\beta^1), \ldots, (m_0^n, m_1^n, c_\beta^n)$, the plaintexts m_0, m_1 are the output of the function f on the corresponding component, that is $m_0 = f(m_0^1, \ldots, m_0^n)$ and $m_1 = f(m_1^1, \ldots, m_1^n)$. Therefore, the list L exhaustively keeps all the queries, which permits properly defining the decryption oracle. Given a ciphertext (m_0, m_1, c_β) sent by $\mathcal{A}$, $\mathcal{O}$Decrypt outputs the decryption of c_β. This query is made under the obvious assumption that the ciphertext does not depend on the bit β, as it would give a straightforward advantage to $\mathcal{A}$. This assumption is formalized by verifying that $m_0 = m_1$.

Definition 5 (IND-CPA^D Security)**.** Let $\texttt{HE} = (\texttt{KeyGen}, \texttt{Enc}, \texttt{Eval}, \texttt{Dec})$ be an HE scheme for message space $\mathcal{M}$ and functionality space $\mathcal{F} : \mathcal{M}^n \to \mathcal{M}$. We say that $\texttt{HE}$ is semantically secure if for all PPT adversary $\mathcal{A}$, it holds that the advantage of $\mathcal{A}$ given by $\mathbf{Adv}^{\text{ind-cpa}^D}_{\texttt{HE},\mathcal{A}}(\lambda) = \left| Pr\left[\mathbf{Exp}^{\text{ind-cpa}^D}_{\texttt{HE},\mathcal{A}}(\lambda) \to \text{true}\right] - \frac{1}{2}\right|$ is negligible in λ, where the experiment is defined as follows:

$\underline{\mathbf{Exp}^{\text{ind-cpa}^D}_{\texttt{HE},\mathcal{A}}(\lambda)\text{:}}$

$b \xleftarrow{\$} \{0,1\}$; $L \leftarrow \emptyset$
$(\texttt{sk}, \texttt{pk}, \texttt{evk}) \xleftarrow{\$} \texttt{KeyGen}(1^\lambda)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}\text{racles}}(1^\lambda, \texttt{pk}, \texttt{evk})$
return $(b = b')$

$\mathcal{O}$racles

$\underline{\mathcal{O}\textsc{Encrypt}(\texttt{pk}, m_0, m_1) :}$

If $m_0, m_1 \notin \mathcal{M}$: **return** $\perp$
$c \xleftarrow{\$} \texttt{Enc}(\texttt{pk}, m_\beta)$
$L \leftarrow L \cup \{(m_0, m_1, c)\}$
return c

$\underline{\mathcal{O}\textsc{Decrypt}((m_0, m_1, c)) :}$

If $(m_0, m_1, c) \notin L$ or $m_0 \neq m_1$:
  return $\perp$
$m \leftarrow \texttt{Dec}(\texttt{sk}, c)$
$L \leftarrow L \cup \{c\}$
return m

$\underline{\mathcal{O}\textsc{Eval}\left(\texttt{evk}, f, (m_0^i, m_1^i, c^i)_{i\in[n]}\right) :}$

If $f \notin \mathcal{F}$: **return** $\perp$
For $i \in [1, n]$:
  If $(m_0^i, m_1^i, c^i) \notin L$: **return** $\perp$
$m_0 \leftarrow f(m_0^1, \ldots, m_0^n)$
$m_1 \leftarrow f(m_1^1, \ldots, m_1^n)$
$c \leftarrow \texttt{Eval}\left(\texttt{evk}, f, (c^1, \ldots, c^n)\right)$
$L \leftarrow L \cup \{(m_0, m_1, c)\}$
return c

Figure 7: Oracles in the IND-CPA^D game for an HE scheme $\texttt{HE}$ and PPT adversary $\mathcal{A}$.

Definition 6 (IND-CPA security for SKE)**.** Let $\texttt{SKE} = (\texttt{Gen}, \texttt{Enc}, \texttt{Dec})$ be an SKE scheme for message space $\mathcal{M}$ and key space $\mathcal{K}$. We say that $\texttt{SKE}$ is semantically secure if for all PPT adversary $\mathcal{A}$, it holds that the advantage $\mathbf{Adv}^{\text{ind-cpa}}_{\texttt{SKE},\mathcal{A}}(\mu) = \left| Pr\left[\mathbf{Exp}^{\text{ind-cpa}}_{\texttt{SKE},\mathcal{A}}(\mu) \to \text{true}\right] - \frac{1}{2}\right|$ of $\mathcal{A}$ is negligible in μ, where the experiment is defined in Figure 9.

$\underline{\mathbf{Exp}^{\text{ind-cpa}}_{\texttt{SKE},\mathcal{A}}(\mu)\text{:}}$

$b \xleftarrow{\$} \{0,1\}$; $\texttt{K} \xleftarrow{\$} \texttt{Gen}(1^{\mu})$
$b' \leftarrow \mathcal{A}(1^{\mu})$
return $(b = b')$

$\underline{\mathcal{O}\textsc{Encrypt}(m_0, m_1)\ \text{:}}$

If $m_0, m_1 \notin \mathcal{M}$ or $|m_0| \neq |m_1|$:
  return $\perp$
$c \xleftarrow{\$} \texttt{Enc}(\texttt{K}, m_b)$
$L \leftarrow L \cup \{c\}$
return c

Figure 8: Security indistinguishability game for SKE. $\mathbf{Exp}^{\text{ind-cpa}}_{\texttt{SKE},\mathcal{A}}$ is the ind-cpa experiment for an SKE scheme SKE and PPT adversary $\mathcal{A}$. $\mathcal{O}$ENCRYPT is an encryption oracle accessible to $\mathcal{A}$.

The security game for SKE is similar to the previous one, except that the information given to the adversary $\mathcal{A}$ is minimal. In a secret-key setup, $\mathcal{A}$ does not know any key and has access to only a single encryption oracle, $\mathcal{O}$ENCRYPT, that, on the input of two messages m_0, m_1 chosen by $\mathcal{A}$, outputs the encryption of m_b under the secret key.

6.2 Semantic Security extended to HHE

This section provides the first security definition for HHE, building on the HE and SKE definitions of the previous section. We also formally demonstrate the semantic security of HHE with respect to our definition.

Definition 7 (IND-CPA Security for HHE)**.** Let $\texttt{HHE} = (\texttt{KeyGen}, \texttt{Encap}, \texttt{Enc}, \texttt{Decomp}, \texttt{Eval}, \texttt{Dec})$ be an HHE scheme for message space $\mathcal{M}$ and functionality space $\mathcal{F}$. We say that HHE is IND-CPA secure if for all PPT adversary $\mathcal{A}$, it holds that the advantage of $\mathcal{A}$:

$$\mathbf{Adv}^{\text{ind-cpa}}_{\texttt{HHE},\mathcal{A}}(\lambda, \mu) = \left| Pr\left[\mathbf{Exp}^{\text{ind-cpa}}_{\texttt{HHE},\mathcal{A}}(\lambda, \mu) \to \text{true}\right] - \frac{1}{2} \right|$$

is negligible in λ and μ, where the experiment is defined in Figure 9.

$\underline{\mathbf{Exp}^{\text{ind-cpa}}_{\texttt{HHE},\mathcal{A}}(\lambda, \mu)\text{:}}$

$b \xleftarrow{\$} \{0,1\}$; $L \leftarrow \emptyset$; $S \leftarrow \emptyset$
$(\texttt{sk}, \texttt{pk}, \texttt{evk}) \xleftarrow{\$} \texttt{KeyGen}(1^{\lambda})$
$(\texttt{K}, c_{\texttt{K}}) \xleftarrow{\$} \texttt{Encap}(\texttt{pk}, 1^{\mu})$
$b' \xleftarrow{\$} \mathcal{A}(1^{\lambda}, \texttt{pk}, \texttt{evk}, c_{\texttt{K}})$
return $(b = b')$

$\underline{\mathcal{O}\textsc{Encrypt}(m_0, m_1)\ \text{:}}$

If $m_0, m_1 \notin \mathcal{M}$:
  return $\perp$
$c \xleftarrow{\$} \texttt{Enc}(\texttt{K}, m_b)$
$S \leftarrow S \cup \{c\}$
return c

$\underline{\mathcal{O}\textsc{Decomp}(\texttt{evk}, c_{\texttt{K}}, c_m)\ \text{:}}$

If $c_m \notin S$: **return** $\perp$
$c \leftarrow \texttt{Decomp}(\texttt{evk}, c_{\texttt{K}}, c_m)$
$L \leftarrow L \cup \{c\}$
return c

$\underline{\mathcal{O}\textsc{Eval}(\texttt{evk}, f, (c_1, \ldots, c_n))\ \text{:}}$

If $f \notin \mathcal{F}$: **return** $\perp$
For $i \in [1, n]$
  If $c_i \notin L$: **return** $\perp$
$c \leftarrow \texttt{Eval}\left(\texttt{evk}, f, (c_1, \ldots, c_n)\right)$
$L \leftarrow L \cup \{c\}$
return c

Figure 9: Security indistinguishability game for HHE. $\mathbf{Exp}^{\text{ind-cpa}}_{\texttt{HHE},\mathcal{A}}$ is the ind-cpa experiment for an HHE scheme HHE and PPT adversary $\mathcal{A}$. The three additional algorithms are the oracles accessible to $\mathcal{A}$.

Finally, we provide the following Theorem 1 that ensures the theoretical security of HHE, given that HE and SKE are secure.

Theorem 1. *Let* HHE *be an HHE scheme built from an HE scheme* HE *and an SKE scheme* SKE*. If* HE *is IND-CPA secure and* SKE *is IND-CPA secure, then* HHE *is IND-CPA secure.*

Proof. We prove Theorem 1 through a sequence of games, starting with the genuine HHE game and ending with a game where the advantage of $\mathcal{A}$ is negligible. We denote ε_i the advantage of $\mathcal{A}$ for game i and $\varepsilon(\lambda)$ (*resp.* $\varepsilon(\mu)$) its advantage for the HE (*resp.* SKE) game.
Game 0: This is the initial genuine HHE game defined in Figure 9. The challenger initializes the game by picking at random a bit $b \xleftarrow{\$} \{0, 1\}$ and generating the keys. On the one hand, she samples the homomorphic keys $(\mathtt{pk}, \mathtt{sk}, \mathtt{evk}) \xleftarrow{\$} \mathtt{KeyGen}(1^\lambda)$, and on the other the encapsulation $(\mathtt{K}, c_\mathtt{K}) \xleftarrow{\$} \mathtt{Encap}(\mathtt{pk}, 1^\mu)$. She submits $\mathtt{sk}, \mathtt{evk}, c_\mathtt{K}$ to a PPT adversary $\mathcal{A}$, who has access to the oracles $\mathcal{O}$Encrypt, $\mathcal{O}$Eval, $\mathcal{O}$Decomp.Eventually, $\mathcal{A}$ outputs a guess b' on the value of b with an advantage ϵ_0.
Game 1: This game is similar to the previous one, except that the oracle $\mathcal{O}$Eval is replaced by an oracle $\mathcal{O}\text{Eval}^{\mathbf{G1}}$ defined as follows:

$$\begin{array}{l}
\underline{\mathcal{O}\textsc{Eval}^{\mathbf{G1}}(\mathtt{evk}, f, (c_1, \ldots, c_n)):} \\
\quad \text{If } f \notin \mathcal{F} : \textbf{return } \perp \\
\quad \text{For } i \in [1, n]\text{: If } c_i \notin L : \textbf{return } \perp \\
\quad c \xleftarrow{\$} \mathcal{C};\ L \leftarrow L \cup \{c\} \\
\quad \textbf{return } c
\end{array}$$

In short, when $\mathcal{A}$ calls the $\mathcal{O}$Eval oracle, the challenger runs the $\mathcal{O}\text{Eval}^{\mathbf{G1}}$ oracle instead, which returns a random element of the ciphertext space $\mathcal{C}$. We prove thereunder that a PPT adversary cannot distinguish this game from the previous one (with a non-negligible advantage).
Claim 1. $|\epsilon_0 - \epsilon_1| \leq \epsilon(\lambda)$, where $\epsilon(\lambda)$ is the advantage of an efficient adversary that breaks the HE game.

To prove this claim, we reduce **Game 1** to the HE-IND-CPA game defined in Figure 6. We introduce a key $\mathtt{pk}^{\mathbf{G1}} := (\mathtt{pk}, \mathtt{evk}, c_K)$ and an oracle $\mathcal{O}\text{Encrypt}^{\mathbf{G1}}$ as follows:

$$\begin{array}{l}
\underline{\mathcal{O}\textsc{Encrypt}^{\mathbf{G1}}(\mathtt{pk}^{\mathbf{G1}}, m_0, m_1):} \\
\quad \text{If } m_0, m_1 \notin \mathcal{M} : \textbf{return } \perp \\
\quad c_m \xleftarrow{\$} \mathcal{O}\textsc{Encrypt}(m_0, m_1) \\
\quad c \xleftarrow{\$} \mathcal{O}\textsc{Decomp}(\mathtt{evk}, c_\mathtt{K}, c_m) \\
\quad \textbf{return } c
\end{array}$$

Note that the oracle $\mathcal{O}\text{Encrypt}^{\mathbf{G1}}$ is well-defined, as it calls oracle accessible to $\mathcal{A}$ and $\mathtt{pk}^{\mathbf{G1}}$ only relies on public information. One notices that $\mathcal{O}\text{Encrypt}'$ corresponds to the classical encryption oracle $\mathcal{O}$Encrypt defined in Figure 6. It follows that $\mathcal{A}$ has access to the oracles of the HE-IND-CPA game, which ends the reduction. Therefore, if $\mathcal{A}$ can break **Game 1**, it has a non-negligible advantage (greater than $\epsilon(\lambda)$) on the HE-IND-CPA game, which concludes.
Game 2: This game proceeds as the previous one, except that we replace the $\mathcal{O}$Decomp oracle with a new oracle $\mathcal{O}\text{Decomp}^{\mathbf{G2}}$, defined as follows:

$$\begin{array}{l}
\underline{\mathcal{O}\textsc{Decomp}^{\mathbf{G2}}(\mathtt{evk}, c_\mathtt{K}, c_m):} \\
\quad \text{If } c_m \notin S : \textbf{return } \perp \\
\quad c \xleftarrow{\$} \mathcal{C};\ L \leftarrow L \cup \{c\} \\
\quad \textbf{return } c
\end{array}$$

In this game, when $\mathcal{A}$ calls the $\mathcal{O}$Decomp oracle, the challenger runs the $\mathcal{O}\text{Decomp}^{\mathbf{G2}}$ oracle instead. In short, the challenger does not run $\mathtt{Decomp}(\mathtt{evk}, c_\mathtt{K}, c_m)$ but instead outputs a random element of the ciphertext space $\mathcal{C}$. We prove that a PPT adversary cannot distinguish this game from the previous one (with a non-negligible advantage).

Claim 2. $|\epsilon_1 - \epsilon_0| \leq \epsilon(\lambda)$, where $\epsilon(\lambda)$ is the advantage of an efficient adversary that breaks the `HE` game.

We prove this claim by constructing an adversary $\mathcal{B}$ that can break the `HE` security game, given an adversary $\mathcal{A}$ that can successfully distinguish the output of the oracle $\mathcal{O}\textsc{Decomp}^{\mathbf{G2}}$ from the original oracle $\mathcal{O}\textsc{Decomp}$; that is, given c_K and c_m, A can distinguish $c \xleftarrow{\$} \mathtt{Eval}(\mathtt{evk}, \mathtt{SKE.Dec}, (c_{\mathtt{K}}, c_m))$ from a random element $c \xleftarrow{\$} \mathcal{C}$ with a non-negligible advantage. During the `HE` game, $\mathcal{B}$ calls the $\mathcal{O}\textsc{Encrypt}$ oracle for the pair of messages $(0, m) \in \mathcal{M}^2$. The challenger outputs a ciphertext c, the encryption of either 0 or m. $\mathcal{B}$ forwards $(\mathtt{evk}, c, m)$ to $\mathcal{A}$ which calls the oracle $\mathcal{O}\textsc{Decomp}(\mathtt{evk}, c, m)$. If $c = \mathtt{Enc}(\mathtt{pk}, 0)$, then c is the encapsulation of the trivial symmetric key $c = 0$ and $m = \mathtt{SKE.Enc}(0, m)$. Therefore, $\mathcal{O}\textsc{Decomp}(\mathtt{evk}, c, m)$ outputs $\mathtt{Enc}(\mathtt{pk}, m)$.

Game 3: The challenger does not replace any oracle in this game. Instead, it operates at the initialization level and replaces c_K with a random element in the ciphertext space. More formally, it replaces the experiment $\mathbf{Exp}^{\text{ind-cpa}}_{\mathtt{HHE},\mathcal{A}}$ with $\mathbf{Exp}^{\mathbf{G3}}_{\mathcal{A}}$ defined as follows:

$$
\begin{array}{l}
\underline{\mathbf{Exp}^{\mathbf{G3}}_{\mathcal{A}}(\lambda, \mu):} \\
\quad b \xleftarrow{\$} \{0,1\};\ L \leftarrow \emptyset;\ S \leftarrow \emptyset \\
\quad (\mathtt{sk}, \mathtt{pk}, \mathtt{evk}) \xleftarrow{\$} \mathtt{KeyGen}(1^\lambda) \\
\quad \mathtt{K} \xleftarrow{\$} \mathtt{Gen}(\mathtt{pk})\ ;\ c \xleftarrow{\$} \mathcal{C} \\
\quad b' \xleftarrow{\$} \mathcal{A}^{\mathcal{O}\text{racles}}(1^\lambda, \mathtt{pk}, \mathtt{evk}, c) \\
\quad \mathbf{return}\ (b = b')
\end{array}
$$

This game aims to make homomorphic parameters unusable to reduce the standard SKE-IND-CPA game defined in Definition 6.

Claim 3. $|\epsilon_3 - \epsilon_2| \leq \epsilon(\lambda) + \epsilon(\mu)$, where $\epsilon(\lambda)$ is the advantage of an efficient adversary that breaks the `HE` game and $\epsilon(\mu)$ is the advantage of an efficient adversary that breaks the `SKE` game.

The key argument to prove this claim is that the homomorphic ciphertext $c_{\mathtt{K}}$ is not used in any oracle anymore, as $\mathcal{O}\textsc{Eval}$ and $\mathcal{O}\textsc{Decomp}$ have both been replaced by random oracles in the previous games. Hence, distinguishing the output $c_{\mathtt{K}}$ of the algorithm `Encap` from a random element $c \xleftarrow{\$} \mathcal{C}$ is reduced to an IND-CPA game in a PKE setup. The advantage of $\mathcal{A}$ to distinguish c from $c_{\mathtt{K}}$ is less than $\epsilon(\lambda)$. Now, one notices that $\mathbf{Exp}^{\mathbf{G3}}_{\mathcal{A}}$ is the SKE experiment $\mathbf{Exp}^{ind-cpa}_{\mathtt{SKE},\mathcal{A}}(\mu)$ defined in Definition 6. By hypothesis, $\mathcal{A}$ has a negligible advantage $\epsilon(\mu)$ to win this game. Finally, its advantage to distinguish **Game 2** and **Game 3** is less than $\epsilon(\lambda) + \epsilon(\mu)$.

Conclusion. The overall advantage $\epsilon_{\mathtt{HHE}}$ of $\mathcal{A}$ in the `HHE` game defined in Definition 7 is $\epsilon_{\mathtt{HHE}} = 3\,\epsilon(\lambda) + \epsilon(\mu)$. As a finite sum of negligible elements, $\epsilon_{\mathtt{HHE}}$ is negligible, which concludes. □

6.3 Attacks on HE-friendly Symmetric Ciphers

To deliver a solid understanding of applicable attacks against HE-friendly ciphers, we started by exploring and categorizing different attacks. We presented three primary attack categories applicable to these ciphers: algebraic-based, differential-based, and linear-based attacks, along with LWE-based attacks within the context of HHE frameworks in Section 7. Based on our study, this part summarizes our security evaluation of recent attacks on HE-friendly symmetric ciphers. Moreover, we present a concise summary of the security evaluation in Table 6, covering each scheme's claims and recent attacks on state-of-the-art HHE schemes.

Attacks on HE-friendly ciphers. Initially resilient to differential and linear cryptanalysis, LowMC encountered challenges, exposing vulnerabilities to algebraic attacks and linearization techniques [DLMW15, GKRS20, BBVD20, BBVY21, LSW+22, QYS+23].

In [DLR16], the authors demonstrated that an adversary could break the FLIP cipher using a guess-and-determine strategy based on a fixed internal state. In their study [LSMI21], authors successfully executed trivial linearization attacks on Rasta and Dasta through algebraic cryptanalysis. Furthermore, a recent technique known as coefficient grouping [LAW+23] evaluated the algebraic degree of Chaghri, resulting in the breaking of its full 8-round with low complexity. The 4-round instance of Elisabeth fell victim to a known-IV linearization attack [GHBJR23], leading to key recovery. Subsequently, Elisabeth's authors proposed a patch [HMS23] to address the security weaknesses of their scheme.

A new attack strategy, SASTA [ASR24], utilizes Differential Fault Analysis (DFA) to break PASTA, achieving full key recovery. SASTA extends to other HHE schemes, such as RASTA, MASTA, and HERA, resulting in a unique key recovery. Similarly, authors in [JLHG24] established a DFA attack against HERA. Authors in [WT24] provide more details of the DFA attack on MASTA, PASTA, and Elisabeth.

Recently, Meaux et al. [MR24] proposed a novel technique for conducting DFA attacks on the FLIP and FiLIP. This technique enables successful key recovery for both of these schemes. Notably, the new approach applies to any filtering function, provided that only a limited number of keystream bits are involved. Nevertheless, Rubato remains secure against SASTA due to adding random noise from a Gaussian distribution to the keystream.

However, in a recent study [GMAH+23], authors showed that it is possible to overcome the noise using a brute force attack. They then recovered the positions of keystream bits without introducing additional noise. As a result, the Rubato cipher became vulnerable to full-key recovery through a linearization attack (Section 7 provides detailed definitions of each attack).

Table 6: Security evaluation of HHE symmetric ciphers against common attacks

Attacks/Scheme	LowMC	Kreyvium	FLIP	Rasta	FiLIP	Dasta	Masta	Pasta	FASTA	HERA	Rubato	Elisabeth	Chaghri
Algebraic Attacks	✗	✓	✗	✗	✗	✗	✓	✓	✓	✓	✗	✗	✗
Trivial Linearization	✗	∗	∗	✗	∗	✗	✓	✓	✓	✓	✗	✗	✗
Number of Monomials	∗	∗	∗	✓	∗	∗	✓	∗	✓	∗	∗	✗	∗
Gröbner basis attack	✗	∗	∗	✓	✓	∗	✓	✓	✓	✓	✓	∗	✓
GCD attack	∗	∗	∗	∗	∗	∗	∗	∗	✓	✓	✓	∗	∗
Differential Cryptanalysis	✓	✓	∗	✓	∗	∗	✓	✓	✓	✓	✓	∗	∗
Higher-order Differential Attacks	✓	∗	∗	✓	∗	∗	✓	✓	∗	∗	∗	∗	✓
Truncated Cryptanalysis	✓	∗	∗	∗	∗	✓	✓	∗	✓	✓	✓	∗	✓
Cube Attack	✓	✓	✓	✓	∗	∗	✓	∗	✓	✓	✓	∗	∗
Invariant Subspace Trail	✗	∗	∗	∗	∗	∗	∗	∗	∗	✓	✓	∗	✓
Linear Cryptanalysis	✓	∗	∗	✓	∗	✓	✓	∗	∗	✓	✓	∗	✓
Interpolation Attack	✗	∗	∗	∗	∗	∗	✓	∗	✓	✓	✓	∗	✓
Boomerang Attacks	✓	∗	∗	∗	∗	∗	✓	∗	∗	∗	∗	∗	∗
Time-Memory Trade-Off (TMDTO)	∗	✓	∗	∗	∗	∗	∗	∗	∗	∗	∗	∗	∗
Correlation attacks	∗	∗	✓	∗	✓	∗	∗	∗	∗	∗	∗	✓	∗
Guess and Determine Attacks	∗	∗	✗	✓	✓	✗	∗	∗	✓	∗	∗	✓	∗
Augmented Function attacks	∗	∗	✓	∗	∗	∗	∗	∗	∗	∗	∗	∗	∗
BKW-like Attack	∗	∗	✓	∗	∗	∗	∗	∗	∗	∗	✓	∗	∗
Differential Fault Analysis (DFA)	∗	∗	✗	✗	✗	∗	✗	✗	∗	✗	✓	✗	∗

✓ denotes that the scheme resists the attack.
✗ denotes the scheme's vulnerability to the attack.
∗ denotes that the authors did not claim security against the specified attack.

Note 6.1: In Conclusion

This analysis reveals that the vulnerabilities of HHE schemes mainly stem from the vulnerabilities of the underlying symmetric cipher. Depending on the cipher, it can lead to a complete security breach or a simple weakening of the security. In any case, the core principle of HHE is not threatened. As noted, the DFA attack [ASR24, WT24, MR24] is the only attack that utilizes the structure of HHE to recover the key and hence endangers the core principle of this technique. Fortunately, this attack can only be conducted under certain conditions, and **noisy ciphers**, such as Rubato, remain secure.

7 Attacks Categorization

This section explores three primary attack categories applicable to these ciphers: algebraic-based, differential-based, and linear-based attacks. Additionally, we discuss LWE-based attacks within the context of HHE frameworks.

7.1 Algebraic-based Attacks

Algebraic attacks [CM03, Cou03a] represent a class of cryptographic attacks that utilize an algebraic system of equations to extract the key stream. In such attacks, armed with (plaintext/ciphertext) pairs, an attacker formulates key-stream outputs as multivariate polynomials over the secret key elements. The key can be recovered by solving this system of equations. Techniques for solving these algebraic systems span from simple linearization to sophisticated methods employing Gröbner bases. Our understanding of algebraic attacks on stream ciphers has been enhanced by recent proposals like the extreme algebraic attack [MW24], which shows significant applicability to ciphers such as FLIP and FiLIP.

In the case of **Trivial Linearization**, the technique involves replacing all monomials with new variables, thereby transforming a system of polynomial equations into a linear form. The effectiveness of algebraic attacks can be influenced by the **Number of Monomials**, particularly when the cipher has a limited number of them. An attacker could resort to key guessing to decrease the number of monomials, thereby enhancing the likelihood of linearization and facilitating the solution of the system.

The **Gröbner Basis Attack** [Fau99, Fau02, SS21] is a more advanced technique that solves a polynomial system by computing a Gröbner basis. Once the basis is computed, variables can be systematically eliminated by altering the order of monomials.

Another strategy is the **GCD Attack** [HKL+22], which calculates the greatest common divisor (GCD) of univariate polynomials. This attack is typically used in ciphers that operate over a large field where the representation is a polynomial in a single variable. This attack can be extended for multivariate polynomial equations by guessing all key variables except one.

The **Guess and Determine Attack** [DLR16] commences by *guessing* specific bits of the internal state or the key and then uses information from keystream bits to *determine* the unknown bits. Assisted by algebraic attacks, guess-and-determine attacks are often feasible. In FP-based HHE schemes, such as FLIP, an adversary might employ a guess-and-determine strategy due to the use of a fixed internal state. Moreover, in schemes where the internal state remains constant, like FLIP, and the register is unaltered, guessing a single bit at any moment can provide information about another bit at a different time. Additionally, the FP in these schemes is characterized by a limited number of high-degree monomials.

The **Cube Attack** [DS09] is utilized to tackle the complex problem of solving multivariate systems of nonlinear equations over a finite field. The fundamental principle behind the cube attack lies in the observation that polynomial equations generated by many symmetric-key cryptosystems are not arbitrary and unrelated. Instead, they often originate from a single *master polynomial*, with *tweakable variables* that the attacker can set to any desired value during a chosen plaintext attack. Given a symmetric-key cipher with $n + m$ input bits of secret and public variables, the goal is to determine the algebraic normal form of the output over $\mathbb{F}_2$, denoted by P. This normal form represents a sum of monomial products. The cube attack consists of two distinct phases: preprocessing and online. During preprocessing, the attacker can analyze the cipher by running it with various keys and plaintexts. Subsequently, in the online phase, the n secret values are set to unknown, allowing the attacker to assign values to the m public variables as desired and evaluate P on the combined input.

The **Integral Attack** [KW02] is used to predict the values in the integrals after a certain number of encryption rounds. For any multiset S comprising elements in $\mathbb{F}_2^n$, the integral over S is precisely defined as the sum of all its elements, denoted as $\bigoplus_{e \in S} e$. Integral attacks exploit the specific value obtained by integrating a function F over a carefully selected input set $\mathcal{X}$. This is mathematically expressed as the integral of $F(\mathcal{X}) : \bigoplus_{x \in \mathcal{X}} F(x)$. Notably, when $\mathcal{X}$ forms a linear or affine subspace, this integral aligns with the value of a higher-order differential of the function.

7.2 Differential-based attacks

Differential Cryptanalysis [BS91] is a type of attack that uses chosen plaintexts and evaluates the probability of differentials, denoted by a pair (α, β). Here, α is the difference between a pair of distinct inputs m_1 and m_2, while β is a potential difference for the resulting outputs $c_1 = f(m_1)$ and $c_2 = f(m_2)$. The primary goal is to identify input pairs (m_1, m_2) with the same difference α that, upon encryption, yield output pairs (c_1, c_2) with the same difference β at an unusually high probability. A well-constructed differential can be used to mount distinguishing attacks and key recovery attacks on the cipher. Typically, the differential needs to cover all but one or a few rounds of the cipher to achieve this goal.

Truncated Cryptanalysis [Knu95, KB96] is a generalization of Differential Cryptanalysis that focuses on differentials predicting only parts of an n-bit value, allowing the other bits to take any possible value. This is referred to as a *truncated differential*. More formally, if (α, β) represents an r-round differential, and α' is a subsequence of α while β' is a subsequence of β, then the pair (α', β') is termed an r-round truncated differential.

Boomerang Attack [Wag99] is a variant of differential cryptanalysis designed specifically for ciphers where identifying high-probability differentials is challenging. The Boomerang attack constructs a distinguisher with two short differentials (α_1, β_1) and (α_2, β_2). Initially, it dissects the cipher $E : \mathbb{F}_2^n \times \mathbb{F}_2^k \to \mathbb{F}_2^n$ into two sub-ciphers denoted as $E = E_1 \circ E_2$. For an r-round block cipher, E_1 contains the first r_1 rounds, while E_2 handles the remaining $r_2 = r - r_1$ rounds. Combining these two differential characteristics makes the Boomerang attack effective against ciphers that might resist conventional differential cryptanalysis. This method has successfully broken ciphers previously considered secure against traditional differential cryptanalysis techniques.

Invariant Subspace (Trail) Attacks [LAAZ11, LMR15, GRR16] exploit a structural property inherent in block ciphers. Specifically, they leverage the property that a partition of the plaintext space into a set and its complement is preserved under the application of the block cipher. If an invariant subspace (V) exists for both the round function (F) and the key schedule function (f), an invariant subspace trail attack can be effectively deployed to establish a rapid distinguisher and facilitate key recovery. By deriving round keys from a master key K as $(k_0, \dots, k_n) = f(\mathsf{K})$ and considering a coset $V \oplus a = \{v \oplus a \mid \forall v \in V\}$, where V is a subspace of a vector space W and a is an element of W, such that $F(V \oplus a) = V \oplus a'$; if the master key K resides in $V \oplus (a \oplus a')$, then it logically follows that $F(V \oplus a') \oplus \mathsf{K} = V \oplus a$, allowing the derivation of an iterative invariant subspace. A *subspace trail* of length r is then essentially a set of $r+1$ subspaces $(V_1, \dots, V_{r+1})$ that satisfy $F(V_i \oplus a_i) \oplus \mathsf{K} \subseteq (V_{i+1} \oplus a_{i+1})$.

Higher-order Differential Cryptanalysis [Lai94] employs higher-order derivatives to extend Differential Cryptanalysis for deriving the secret key when more than two inputs are provided. As mentioned in [Lai94], “if a (nontrivial) *i-th* derivative of $(r-1)$ round function takes on a value with a high probability, then it is possible to derive the key for the last round from the known 2^i outputs and from the value of the anticipated derivative.”

Interpolation Attack [JK97] involves determining the polynomial representation of a state bit. By combining knowledge about the restrictions of this polynomial with a sufficient number of evaluations of the polynomial function, the attacker reconstructs the polynomial representation using (plaintext/ciphertext) pairs through Lagrange interpolation. With the algebraic representation of the system as the function $f(x, k)$, linking the key-independent integral to the ciphertext x and the last round key k, interpolation attacks express f as a function of known ciphertext bits with unknown coefficients. This results in an equation of degree 1 in the unknown coefficients for any values of the ciphertext, recoverable by solving a linear system. The interpolation attack is commonly employed to exploit cryptographic algorithm vulnerabilities by scrutinizing the behavior of the polynomial functions used for generating cryptographic keys.

Differential Fault Analysis (DFA) [TMA11] is a physical attack where the attacker gains access to public information, such as nonce, IV, inputs, and outputs of the device running the cipher for a limited time. The attacker injects a fault into the cipher's input to obtain a different result for the final state and then employs differential analysis to uncover the key. In [ASR24], the author introduces a new DFA model, SASTA, tailored for HHE schemes. SASTA initially targets PASTA and subsequently achieves successful key recovery in other schemes like RASTA, MASTA, and HERA. Similarly, authors in [MR24] targeted the FLIP and FiLIP schemes with a DFA attack.

7.3 Linear-based attacks

Linear Cryptanalysis [Mat93, BSV07] is a commonly used method for analyzing the security of a cipher. The cryptanalyst seeks to identify affine approximations of the cipher that hold with substantial accuracy. This process involves uncovering linear characteristics, which are sequences of linear approximations applied to consecutive rounds of the cipher. These linear characteristics significantly impact **S-boxes**, playing a crucial role in the approximations. Similar to differential cryptanalysis, linear cryptanalysis can be employed to initiate distinguishing and key recovery attacks.

Correlation Attacks [Sie84] primarily apply to stream ciphers for extracting information on secret key bits. These attacks, specifically key-recovery attacks, can be executed when a straightforward dependency between the keystream sequence $ks = (ks_0, ks_1, \ldots, ks_n)$ and the state s or key $\mathtt{K}$ is identified. Typically, correlation attacks focus on state recovery, utilizing a single keystream sequence. In **Fast Correlation Attacks** [MS88], the approach involves attempting to discover a low-weight parity check polynomial of the system's linear part, followed by applying an iterative decoding procedure. Additionally, a category of correlation attacks targets filter generators [EJ04], whose objective is to invert the nonlinear function and recover the initial state.

Time-Memory Trade-Off (TMDTO) attacks [HS05, DCLP05] constitute a generic approach for the inversion of one-way functions, applicable to both stream and block ciphers. In the context of stream ciphers, vulnerability to TMDTO arises when the length of the IV is shorter than that of the key. Significantly, this vulnerability remains regardless of the size of the internal state. Additionally, chosen plaintext TMDTO presents a threat to block ciphers across various modes of operation.

Higher-Order Correlation Attacks [Cou03b] primarily target stream ciphers and employ linear approximations of the output function to mount an attack on the cipher. The filtering function is approximated with a degree-d polynomial, and the corresponding algebraic system is solved using Gröbner basis algorithms. The attack's efficiency depends on the function's closeness to a degree-d polynomial. It can be integrated with guess-and-determine attacks, but its complexity consistently exceeds that of fast algebraic or correlation attacks.

Augmented Function attacks [FM07] involve considering x as an n-bit internal state for a stream cipher, with an update function U and output function f producing a single bit of keystream in a single iteration. The augmented function $S_m : \mathbb{F}^n \to \mathbb{F}^m$ is then defined as $S_m(x) = (f(x), f(U(x)), \ldots, f(U^{m-1}(x)))$. The update function may exhibit linearity, resembling a filter generator, or non-linearity. The output y corresponds to an m-bit block of the known keystream. This attack aims to recover the initial state x through algebraic or correlation approaches, utilizing conditional equations $F_y(x) = 0$ of degree d for the output y of the augmented function S_m. This approach emphasizes multiple outputs of the function rather than a singular one to identify coefficients that enable the exploitation of a relationship between the key and the outputs.

7.4 LWE-based attacks

In addition to symmetric cryptanalysis, as mentioned in [HKL+22], LWE cryptanalysis can also be applied to HHE frameworks. The naive approach for solving LWE hard problems is *exhaustive search*. The **meet-in-the-middle (MITM)** [BG14] approach, a variant of the TMDTO attack, can assist exhaustive search. Another method is the **primal attack** [ZZW22], which reduces the LWE problem to the unique-SVP through embedding and then employs lattice reduction techniques such as BKZ [SE94, CN11] to find the shortest vector. Additionally, a **dual attack** [LP11, PS24] can be utilized to distinguish between the uniform distribution and the modular discrete Gaussian over $\mathbb{Z}_q$.

The **BKW-like Attack** [BKW03] is a lattice version of Gaussian elimination parametrized by a and b. Assuming an LWE distribution $L_{\mathbf{s},\chi}$, where $\chi = D_{\alpha q}$ is parameterized by dimension N and modulus q, the BKW attack first reduces A to a block diagonal matrix and then employs it to solve a lattice problem. The **Arora-Ge attack** proposed in [AG11] is an algebraic algorithm designed to solve the search-LWE problem. It leverages the idea that, given LWE samples $\{(\mathbf{a}_i, b_i)\}_i$, the errors fall into some interval $[-t\alpha q, t\alpha q]$ for a sufficiently large t, ensuring that the equations $\prod_{e=-t\alpha q}^{t\alpha q}(b_i - \langle \mathbf{a}_i, \mathbf{s}\rangle - e) = 0$ hold.

The authors in [APS15] investigated the computational hardness of solving the LWE problem, focusing on the cost of attacking LWE instances with specific parameter sets. Their work provides concrete guidance and a widely used tool [Ac23] for selecting LWE parameters that guarantee robust security. This is particularly crucial for designing cryptographic schemes based on LWE, including HE and HHE.

8 Discussion

In this paper, we detailed the most recent HHE schemes and evaluated their claims for security and performance through a systematized study. We provided a universal definition for HHE, and following that, we extended the IND-CPA security definition for HHE. Moreover, we analyzed all the potential attacks in the literature for HE-friendly ciphers and HE schemes, resulting in a categorization of these attacks for HHE. Furthermore, we implemented the pioneer HHE schemes to measure their performance in a real-world setting, and we open-sourced our implementation. The field of HHE is constantly evolving due to the continuous advancements in both HE-friendly ciphers and HE schemes, as well as the wide range of applications that HHE supports. Therefore, since our main motivation was to establish the HHE foundation for future research in the field, we are presenting some key takeaways and insights:

T1. Unlike standard symmetric ciphers such as AES, which have undergone extensive practical maintenance and security analysis with well-defined parameter sets, identifying their vulnerabilities across various applications, HE-friendly ciphers are still in the early stages of development and require significant progress to achieve enterprise-level adoption. While existing efforts [ACC+21, BCC+24] aim to standardize HE parameter sets for different security levels, a key missing component for HHE schemes is the establishment of a standardized set of parameters that aligns with existing HE parameter sets.

T2. The HHE schemes have been developed to reduce the computation and communication overheads for clients with limited resources. However, in a 2-party model, the result of transciphering, which is still a homomorphic cipher, will expand due to further homomorphic evaluation. Eventually, the client will need to download and decrypt this ciphertext. There are techniques for HE, such as ciphertext compression [MDK23], which decrease communication costs. This could be an interesting research direction for HHE schemes as well.

T3. By using HHE, a massive part of computation can be offloaded to the server side, allowing users with low-power devices to benefit from HE-based privacy-preserving computation for any application. This approach helps application owners attract more users and create a more scalable system that accommodates low-powered devices. However, this places a higher demand on the server side. An interesting question that remains to be answered is "What is the energy consumption of HHE schemes, and how does it compare to the energy consumption of HE?"

T4. Many encoding and packing techniques have been utilized in HE schemes. One of the drawbacks of PASTA was the exact problem due to the underlying HE scheme. In [BCK+23], the author suggested using ring-packing techniques to create a more efficient framework. In their approach, the client encrypts data into small-degree LWE ciphers, which are then packed into an RLWE cipher on the server side. Again, analyzing this new approach and adjusting it with a HE-friendly cipher is a likely research direction.

T5. Finally, as mentioned earlier, using standard symmetric ciphers such as AES has been the primary aim of HHE schemes. However, due to the high multiplicative depth, it was impractical, leading to new research attempts to design HE-friendly ciphers until recent advancements in the field. Since design-wise, AES is more complex than HE-friendly ciphers with low multiplicative depth, discovering the possibility and experimental results for applying the same techniques [ADE+23, TCBS23, WWL+23] to state-of-the-art HE-friendly ciphers can also be a potential research direction.

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.

[ABBB+22] Ahmad Al Badawi, Jack Bates, Flavio Bergamaschi, David Bruce Cousins, Saroja Erabelli, Nicholas Genise, Shai Halevi, Hamish Hunt, Andrey Kim, Yongwoo Lee, et al. Openfhe: Open-source fully homomorphic encryption library. In *Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, pages 53–63, 2022.

[ABP23] Ahmad Al Badawi and Yuriy Polyakov. Demystifying bootstrapping in fully homomorphic encryption. *Cryptology ePrint Archive*, 2023.

[Ac23] Martin Albrecht and contributors. lattice-estimator: Estimation of lattice attack complexities. https://github.com/malb/lattice-estimator, 2023. Accessed: 2025-01-05.

[ACC+21] Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, et al. Homomorphic encryption standard. *Protecting privacy through homomorphic encryption*, pages 31–62, 2021.

[ACLS18] Sebastian Angel, Hao Chen, Kim Laine, and Srinath Setty. Pir with compressed queries and amortized query processing. In *2018 IEEE symposium on security and privacy (SP)*, pages 962–979. IEEE, 2018.

[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*, 2023.

[AG11] Sanjeev Arora and Rong Ge. New algorithms for learning in presence of errors. In *International Colloquium on Automata, Languages, and Programming*, pages 403–415. Springer, 2011.

[AMT22] Tomer Ashur, Mohammad Mahzoun, and Dilara Toprakhisar. Chaghri-a fhe-friendly block cipher. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 139–150, 2022.

[APS15] Martin R Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

[ARS+15] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for mpc and fhe. In *Advances in Cryptology – EUROCRYPT 2015*, pages 430–454, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[ASP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I 34*, pages 297–314, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[ASR24] Aikata Aikata, Dhiman Saha, and Sujoy Sinha Roy. Sasta: Ambushing hybrid homomorphic encryption schemes with a single fault. *Cryptology ePrint Archive*, 2024.

[BBVD20] Subhadeep Banik, Khashayar Barooti, Serge Vaudenay, and Fatma Betül Durak. Cryptanalysis of lowmc instances using single plaintext/ciphertext pair. *IACR Transactions on Symmetric Cryptology*, 2020(4):130–146, 2020.

[BBVY21] Subhadeep Banik, Khashayar Barooti, Serge Vaudenay, and Hailun Yan. New attacks on lowmc instances with a single plaintext/ciphertext pair. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021*, pages 303–331, Cham, 2021. Springer International Publishing.

[BCC+24] Jean-Philippe Bossuat, Rosario Cammarota, Ilaria Chillotti, Benjamin R Curtis, Wei Dai, Huijing Gong, Erin Hales, Duhyeong Kim, Bryan Kumara, Changmin Lee, et al. Security guidelines for implementing homomorphic encryption. *Cryptology ePrint Archive*, 2024.

[BCG+12] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, et al. Prince–a low-latency block cipher for pervasive computing applications. In *Advances in Cryptology–ASIACRYPT 2012: 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings 18*, pages 208–225. Springer, 2012.

[BCK+23] Youngjin Bae, Jung Hee Cheon, Jaehyung Kim, Jai Hyun Park, and Damien Stehlé. Hermes: efficient ring packing using mlwe ciphertexts and application to transciphering. In *Annual International Cryptology Conference*, pages 37–69. Springer, 2023.

[BFM22] Alexandros Bakas, Eugene Frimpong, and Antonis Michalas. Symmetrical disguise: Realizing homomorphic encryption services from symmetric primitives. In *International Conference on Security and Privacy in Communication Systems*, pages 353–370. Springer, 2022.

[BG14] Shi Bai and Steven D Galbraith. Lattice decoding attacks on binary lwe. In *Information Security and Privacy: 19th Australasian Conference, ACISP 2014, Wollongong, NSW, Australia, July 7-9, 2014. Proceedings 19*, pages 322–337. Springer, 2014.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, page 309–325, New York, NY, USA, 2012. Association for Computing Machinery.

[BHKR13] Mihir Bellare, Viet Tung Hoang, Sriram Keelveedhi, and Phillip Rogaway. Efficient garbling from a fixed-key blockcipher. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, SP '13, page 478–492, USA, 2013. IEEE Computer Society.

[BKW03] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM (JACM)*, 50(4):506–519, 2003.

[Bra12] Zvika Brakerski. Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP. In *Advances in Cryptology – CRYPTO 2012*, pages 868–886, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

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

[BSV07] Thomas Baigneres, Jacques Stern, and Serge Vaudenay. Linear cryptanalysis of non binary ciphers: (with an application to safer). In *Selected Areas in Cryptography: 14th International Workshop, SAC 2007, Ottawa, Canada, August 16-17, 2007, Revised Selected Papers 14*, pages 184–211. Springer, 2007.

[BY03] Mihir Bellare and Bennet Yee. Forward-security in private-key cryptography. In Marc Joye, editor, *Topics in Cryptology — CT-RSA 2003*, pages 1–18, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[CCF+15] 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. Cryptology ePrint Archive, Paper 2015/113, 2015.

[CCK+13] Jung Hee Cheon, Jean-Sébastien Coron, Jinsu Kim, Moon Sung Lee, Tancrède Lepoint, Mehdi Tibouchi, and Aaram Yun. Batch fully homomorphic encryption over the integers. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, pages 315–335, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[CCP+24] Jung Hee Cheon, Hyeongmin Choe, Alain Passelègue, Damien Stehlé, and Elias Suvanto. Attacks against the INDCPA-d security of exact FHE schemes. Cryptology ePrint Archive, Paper 2024/127, 2024.

[CGGI16] 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: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I 22*, pages 3–33. Springer, 2016.

[CGGI17] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Faster packed homomorphic operations and efficient circuit bootstrapping for tfhe. In *International Conference on the Theory and Application of Cryptology and Information Security*, 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.

[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 Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021*, pages 640–669, Cham, 2021. Springer International Publishing.

[CHLR18] Hao Chen, Zhicong Huang, Kim Laine, and Peter Rindal. Labeled psi from fully homomorphic encryption with malicious security. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1223–1237, 2018.

[CHMS22] Orel Cosseron, Clément Hoffmann, Pierrick Méaux, and François-Xavier Standaert. Towards case-optimized hybrid homomorphic encryption. In *Advances in Cryptology – ASIACRYPT 2022*, pages 32–67, Cham, 2022. Springer Nature Switzerland.

[CIR22] Carlos Cid, John Petter Indrøy, and Håvard Raddum. Fasta – a stream cipher for fast fhe evaluation. In *Topics in Cryptology – CT-RSA 2022*, pages 451–483, Cham, 2022. Springer International Publishing.

[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*, pages 1–19, Cham, 2021. Springer International Publishing.

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23*, pages 409–437. Springer, 2017.

[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 Hugo Krawczyk, editor, *Public-Key Cryptography – PKC 2014*, pages 311–328, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[CM03] Nicolas T Courtois and Willi Meier. Algebraic attacks on stream ciphers with linear feedback. In *Advances in Cryptology—EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings 22*, pages 345–359. Springer, 2003.

[CMdG$^+$21] Kelong Cong, Radames Cruz Moreno, Mariana Botelho da Gama, Wei Dai, Ilia Iliashenko, Kim Laine, and Michael Rosenberg. Labeled psi from homomorphic encryption with reduced computation and communication. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1135–1150, 2021.

[CN11] Yuanmi Chen and Phong Q Nguyen. Bkz 2.0: Better lattice security estimates. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 1–20. Springer, 2011.

[Cou03a] Nicolas T Courtois. Fast algebraic attacks on stream ciphers with linear feedback. In *Advances in Cryptology-CRYPTO 2003: 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003. Proceedings 23*, pages 176–194. Springer, 2003.

[Cou03b] Nicolas T Courtois. Higher order correlation attacks, xl algorithm and cryptanalysis of toyocrypt. In *Information Security and Cryptology—ICISC 2002: 5th International Conference Seoul, Korea, November 28–29, 2002 Revised Papers 5*, pages 182–199. Springer, 2003.

[D$^+$97] E Donald et al. The art of computer programming, volume 2: Seminumerical algorithms.-3rd, 1997.

[DCLP05] Christophe De Canniere, Joseph Lano, and Bart Preneel. Comments on the rediscovery of time memory data tradeoffs. *URL: https://www. ecrypt. eu. org/stream/papersdir/040. pdf (Last accessed: 25.11. 2020)*, 2005.

[DCP08] Christophe De Canniere and Bart Preneel. *Trivium.* Springer, Berlin, Heidelberg, 2008.

[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 *Advances in Cryptology – CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part I*, page 662–692, Berlin, Heidelberg, 2018. Springer-Verlag.

[DGH$^+$23] Christoph Dobraunig, Lorenzo Grassi, Lukas Helminger, Christian Rechberger, Markus Schofnegger, and Roman Walch. Pasta: A case for hybrid homomorphic encryption. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 30–73, 2023.

[DLMW15] Itai Dinur, Yunwen Liu, Willi Meier, and Qingju Wang. Optimized interpolation attacks on lowmc. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015*, pages 535–560, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[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.

[DM15] Léo Ducas and Daniele Micciancio. Fhew: Bootstrapping homomorphic encryption in less than a second. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, pages 617–640, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[DR05] Joan Daemen and Vincent Rijmen. *Rijndael/AES*, pages 520–524. Springer US, Boston, MA, 2005.

[DS09] Itai Dinur and Adi Shamir. Cube attacks on tweakable black box polynomials. In *Advances in Cryptology-EUROCRYPT 2009: 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings 28*, pages 278–299. Springer, 2009.

[DSES14] Yarkın Doröz, Aria Shahverdi, Thomas Eisenbarth, and Berk Sunar. Toward practical homomorphic evaluation of block ciphers using prince. In *Financial Cryptography and Data Security: FC 2014 Workshops, BITCOIN and WAHC 2014, Christ Church, Barbados, March 7, 2014, Revised Selected Papers 18*, pages 208–220. Springer, 2014.

[Dwo15] Morris J Dworkin. Sha-3 standard: Permutation-based hash and extendable-output functions. 2015.

[ECR12] ECRYPT. The eSTREAM project. http://www.ecrypt.eu.org/stream/, 2012. Accessed: 2023-12-21.

[EJ04] Håkan Englund and Thomas Johansson. A new simple technique to attack filter generators and related ciphers. In *International Workshop on Selected Areas in Cryptography*, pages 39–53. Springer, 2004.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology*, pages 10–18, Berlin, Heidelberg, 1985. Springer Berlin Heidelberg.

[Eli22] Elisabeth implementation. Online: https://github.com/princess-elisabeth/Elisabeth, November 2022.

[Fau99] Jean-Charles Faugere. A new efficient algorithm for computing gröbner bases (f4). *Journal of pure and applied algebra*, 139(1-3):61–88, 1999.

[Fau02] Jean Charles Faugere. A new efficient algorithm for computing gröbner bases without reduction to zero (f5). In *Proceedings of the 2002 international symposium on Symbolic and algebraic computation*, pages 75–83, 2002.

[Fei73] Horst Feistel. Cryptography and computer privacy. *Scientific american*, 228(5):15–23, 1973.

[FM07] Simon Fischer and Willi Meier. Algebraic immunity of s-boxes and augmented functions. In *Fast Software Encryption: 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, March 26-28, 2007, Revised Selected Papers 14*, pages 366–381. Springer, 2007.

[FNB+24] Eugene Frimpong, Khoa Nguyen, Mindaugas Budzys, Tanveer Khan, and Antonis Michalas. Guardml: Efficient privacy-preserving machine learning services through hybrid homomorphic encryption. In *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*, pages 953–962, 2024.

[FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptol. ePrint Arch.*, 2012:144, 2012.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, page 169–178, New York, NY, USA, 2009. Association for Computing Machinery.

[GHBJR23] Henri Gilbert, Rachelle Heim Boissier, Jérémy Jean, and Jean-René Reinhard. Cryptanalysis of elisabeth-4. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 256–284. Springer, 2023.

[GHS12] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the aes circuit. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, pages 850–867, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[GKRS20] Lorenzo Grassi, Daniel Kales, Chistian Rechberger, and Markus Schofnegger. Survey of key-recovery attacks on lowmc in a single plaintext/ciphertext scenario, 2020.

[GM19] Shafi Goldwasser and Silvio Micali. *Probabilistic Encryption & How to Play Mental Poker Keeping Secret All Partial Information*, page 173–201. Association for Computing Machinery, New York, NY, USA, 2019.

[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.

[GMT23] Charles Gouert, Dimitris Mouris, and Nektarios Georgios Tsoutsos. Sok: New insights into fully homomorphic encryption libraries via standardized benchmarks. *Proceedings on Privacy Enhancing Technologies*, 3:154–172, 2023.

[GRR16] Lorenzo Grassi, Christian Rechberger, and Sondre Rønjom. Subspace trail cryptanalysis and its applications to aes. *Cryptology ePrint Archive*, 2016.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, pages 75–92, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[HD24] Yuanshun Huang and Guihua Duan. A privacy-preserving decision tree evaluation scheme for multiple wearable devices. In *2024 27th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 2547–2552. IEEE, 2024.

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

[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 Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022*, pages 581–610, Cham, 2022. Springer International Publishing.

[HL20] Phil Hebborn and Gregor Leander. Dasta – alternative linear layer for rasta. *IACR Transactions on Symmetric Cryptology*, 2020(3):46–86, 09 2020.

[HMS23] Clément Hoffmann, Pierrick Méaux, and François-Xavier Standaert. The patching landscape of elisabeth-4 and the mixed filter permutator paradigm. *Cryptology ePrint Archive*, 2023.

[HS05] Jin Hong and Palash Sarkar. New applications of time memory data tradeoffs. In *Advances in Cryptology-ASIACRYPT 2005: 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4-8, 2005. Proceedings 11*, pages 353–372. Springer, 2005.

[Hyb21] Framework for hybrid homomorphic encryption. Online: https://github.com/IAIK/hybrid-HE-framework, November 2021.

[JK97] Thomas Jakobsen and Lars R Knudsen. The interpolation attack on block ciphers. In *Fast Software Encryption: 4th International Workshop, FSE'97 Haifa, Israel, January 20–22 1997 Proceedings 4*, pages 28–40. Springer, 1997.

[JLHG24] Lin Jiao, Yongqiang Li, Yonglin Hao, and Xinxin Gong. Differential fault attacks on privacy protocols friendly symmetric-key primitives: Rain and hera. *IET Information Security*, 2024(1):7457517, 2024.

[JV02] Daemen Joan and Rijmen Vincent. The design of rijndael: Aes-the advanced encryption standard. *Information Security and Cryptography*, 17:31–50, 2002.

[KB96] Lars R Knudsen and Thomas A Berson. Truncated differentials of safer. In *Fast Software Encryption: Third International Workshop Cambridge, UK, February 21–23 1996 Proceedings 3*, pages 15–26. Springer, 1996.

[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.

[KW02] Lars Knudsen and David Wagner. Integral cryptanalysis. In *Fast Software Encryption: 9th International Workshop, FSE 2002 Leuven, Belgium, February 4–6, 2002 Revised Papers 9*, pages 112–127. Springer, 2002.

[LAAZ11] Gregor Leander, Mohamed Ahmed Abdelraheem, Hoda AlKhzaimi, and Erik Zenner. A cryptanalysis of printcipher: the invariant subspace attack. In *Advances in Cryptology–CRYPTO 2011: 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings 31*, pages 206–221. Springer, 2011.

[Lai94] Xuejia Lai. Higher order derivatives and differential cryptanalysis. *Communications and Cryptography: Two Sides of One Tapestry*, pages 227–233, 1994.

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

[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.

[LLL+22] 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*, pages 12403–12422. PMLR, 2022.

[LM21] Baiyu Li and Daniele Micciancio. On the security of homomorphic encryption on approximate numbers. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021*, pages 648–677, Cham, 2021. Springer International Publishing.

[LMR15] Gregor Leander, Brice Minaud, and Sondre Rønjom. A generic approach to invariant subspace attacks: Cryptanalysis of robin, iscream and zorro. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 254–283. Springer, 2015.

[LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for lwe-based encryption. In *Topics in Cryptology–CT-RSA 2011: The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*, pages 319–339. Springer, 2011.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On Ideal Lattices and Learning with Errors over Rings. *Journal of the ACM*, 60(6):1–35, November 2013.

[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.

[LSW+22] Fukang Liu, Santanu Sarkar, Gaoli Wang, Willi Meier, and Takanori Isobe. Algebraic meet-in-the-middle attack on lowmc. In Shweta Agrawal and Dongdai Lin, editors, *Advances in Cryptology – ASIACRYPT 2022*, pages 225–255, Cham, 2022. Springer Nature Switzerland.

[Mat93] Mitsuru Matsui. Linear cryptanalysis method for des cipher. In *Workshop on the Theory and Application of Cryptographic Techniques*, pages 386–397. Springer, 1993.

[MCJS19] Pierrick Méaux, Claude Carlet, Anthony Journault, and François-Xavier Standaert. Improved filter permutators for efficient fhe: Better instances and implementations. In *Progress in Cryptology – INDOCRYPT 2019: 20th International Conference on Cryptology in India, Hyderabad, India, December 15–18, 2019, Proceedings*, page 68–91, Berlin, Heidelberg, 2019. Springer-Verlag.

[MDK23] Rasoul Akhavan Mahdavi, Abdulrahman Diaa, and Florian Kerschbaum. He is all you need: Compressing fhe ciphertexts using additive he. *arXiv preprint arXiv:2303.09043*, 2023.

[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*, pages 311–343, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[MR24] Pierrick Méaux and Dibyendu Roy. Theoretical differential fault attacks on flip and filip. *Cryptography and Communications*, pages 1–24, 2024.

[MS88] Willi Meier and Othmar Staffelbach. Fast correlation attacks on stream ciphers. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 301–314. Springer, 1988.

[MW24] Pierrick Méaux and Qingju Wang. Extreme algebraic attacks. *Cryptology ePrint Archive*, 2024.

[NBF+24] Khoa Nguyen, Mindaugas Budzys, Eugene Frimpong, Tanveer Khan, and Antonis Michalas. A pervasive, efficient and private future: Realizing privacy-preserving machine learning through hybrid homomorphic encryption. In *2024 IEEE Conference on Dependable, Autonomic and Secure Computing (DASC)*, pages 47–56. IEEE, 2024.

[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*, CCSW '11, page 113–124, New York, NY, USA, 2011. Association for Computing Machinery.

[PS24] Amaury Pouly and Yixin Shen. Provable dual attacks on learning with errors. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024*, pages 256–285, Cham, 2024. Springer Nature Switzerland.

[Pł22] Bartłomiej Płotka. *Efficient go: Data-driven performance optimizations.* O'Reilly Media, Inc., 2022.

[QYS+23] Wenxiao Qiao, Hailun Yan, Siwei Sun, Lei Hu, and Jiwu Jing. New cryptanalysis of lowmc with algebraic techniques. *Designs, Codes and Cryptography*, 91(5):2057–2075, 2023.

[RAD78] R L Rivest, L Adleman, and M L Dertouzos. On data banks and privacy homomorphisms, 1978.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 84–93, 2005.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM*, 56(6):1–40, September 2009.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, feb 1978.

[RtF21] Rtf transciphering framework. Online: https://github.com/KAIST-CryptLab/RtF-Transciphering, November 2021. EPFL-LDS, Tune Insight SA.

[SE94] Claus-Peter Schnorr and Martin Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. *Mathematical programming*, 66:181–199, 1994.

[Sha49] Claude E Shannon. Communication theory of secrecy systems. *The Bell system technical journal*, 28(4):656–715, 1949.

[Sie84] Thomas Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications (corresp.). *IEEE Transactions on Information theory*, 30(5):776–780, 1984.

[SS21] Jan Ferdinand Sauer and Alan Szepieniec. Sok: Gröbner basis algorithms for arithmetization oriented ciphers. *Cryptology ePrint Archive*, 2021.

[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.

[TMA11] Michael Tunstall, Debdeep Mukhopadhyay, and Subidh Ali. Differential fault analysis of the advanced encryption standard using a single fault. In *Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication: 5th IFIP WG 11.2 International Workshop, WISTP 2011, Heraklion, Crete, Greece, June 1-3, 2011. Proceedings 5*, pages 224–233. Springer, 2011.

[Wag99] David Wagner. The boomerang attack. In *International Workshop on Fast Software Encryption*, pages 156–170. Springer, 1999.

[WT24] Weizhe Wang and Deng Tang. Differential fault attack on he-friendly stream ciphers: Masta, pasta and elisabeth. *Cryptology ePrint Archive*, 2024.

[WWL+23] Benqiang Wei, Ruida Wang, Zhihao Li, Qinju Liu, and Xianhui Lu. Fregata: Faster homomorphic evaluation of aes via tfhe. In *International Conference on Information Security*, pages 392–412. Springer, 2023.

[YTH96] AM Youssef, Stafford E Tavares, and HM Heys. A new class of substitution-permutation networks, 1996.

[ZZW22] Xue Zhang, Zhongxiang Zheng, and Xiaoyun Wang. A detailed analysis of primal attack and its variants. *Science China Information Sciences*, 65(3):132301, 2022.