Rubato: Noisy Ciphers for Approximate Homomorphic Encryption (Full Version)

Jincheol Ha[1], Seongkwang Kim[2]*, Byeonghak Lee[1], Jooyoung Lee[1]**, and Mincheol Son[1]

[1] KAIST, Daejeon, Korea,
{smilecjf,lbh0307,hicalf,encrypted.def}@kaist.ac.kr
[2] Samsung SDS, Seoul, Korea,
seongkwang.kim23@gmail.com

Abstract. A transciphering framework converts a symmetric ciphertext into a homomorphic ciphertext on the server-side, reducing computational and communication overload on the client-side. In Asiacrypt 2021, Cho et al. proposed the RtF framework that supports approximate computation.

In this paper, we propose a family of *noisy* ciphers, dubbed Rubato, with a novel design strategy of introducing noise to a symmetric cipher of a low algebraic degree. With this strategy, the multiplicative complexity of the cipher is significantly reduced, compared to existing HE-friendly ciphers, without degrading the overall security. More precisely, given a moderate block size (16 to 64), Rubato enjoys a low multiplicative depth (2 to 5) and a small number of multiplications per encrypted word (2.1 to 6.25) at the cost of slightly larger ciphertext expansion (1.26 to 1.31). The security of Rubato is supported by comprehensive analysis including symmetric and LWE cryptanalysis. Compared to HERA within the RtF framework, client-side and server-side throughput is improved by 22.9% and 32.2%, respectively, at the cost of only 1.6% larger ciphertext expansion.

Keywords: homomorphic encryption, transciphering framework, stream cipher, HE-friendly cipher

1 Introduction

Real-world data typically contain some errors from their true values since they are represented by real numbers rather than bits or integers. Even in the case that input data are represented by exact numbers without approximation, one might have to approximate intermediate values during data processing for efficiency. Therefore, it would be practically relevant to support approximate computation

[*] This work was done while S. Kim was a PhD student at KAIST.

[**] This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No.2021R1F1A1047146).

over encrypted data. The CKKS encryption scheme [24] provides the desirable feature using an efficient encoder for real numbers. Due to this feature, CKKS achieves good performance in various applications, for example, to securely evaluate machine learning algorithms on a real dataset [17,56].

Unfortunately, current HE schemes including CKKS commonly suffer from heavy computational and memory overload. The encryption/decryption speed is relatively slow compared to conventional encryption schemes, and it implies that HE is inadequate for bulk encryption. Also, ciphertext expansion seems to be an intrinsic problem of homomorphic encryption due to the noise used in the encryption algorithm. Although the ciphertext expansion has been significantly reduced down to the order of hundreds in terms of the ratio of a ciphertext size to its plaintext size since the invention of the batching technique [38], it does not seem to be acceptable from a practical viewpoint. Furthermore, this ratio becomes even worse when it comes to encryption of a short message; encryption of a single bit might result in a ciphertext of a few megabytes.

1.1 Transciphering and HE-friendly Ciphers

TRANSCIPHERING FRAMEWORK. To address the issue of computational overload and the ciphertext expansion, a hybrid framework, also called a *transciphering framework*, has been proposed for exact computation [55]. It basically converts a symmetric ciphertext $\mathbf{c} = \mathsf{E}_{\mathbf{k}}(\mathbf{m})$ to a homomorphic ciphertext $\mathsf{Enc}^{\mathsf{HE}}(\mathbf{m})$ by homomorphically evaluating the cipher. For approximate computation, Cho et al. [25] proposed a new transciphering framework, dubbed the RtF framework (see Figure 1). We give a brief description of the RtF framework in the following.

For a given message vector $\mathbf{m} \in \mathbb{R}^n$, a client encrypts an encoded message $\lfloor \Delta \cdot \mathbf{m} \rceil \in \mathbb{Z}_q^n$ using a symmetric cipher E over $\mathbb{Z}_q$ with a secret key $\mathbf{k} \in \mathbb{Z}_q^n$ and a nonce nc; this secret key is encrypted using the FV encryption algorithm $\mathsf{Enc}^{\mathsf{FV}}$. The resulting ciphertexts $\mathbf{c} = \mathsf{E}_{\mathbf{k}}(\mathbf{m})$, the FV-encrypted symmetric key $\mathsf{Enc}^{\mathsf{FV}}(\mathbf{k})$, and the nonce nc are stored in the server. When the server wants to compute $\mathsf{Enc}^{\mathsf{CKKS}}(\mathbf{m})$ (for computation over encrypted data), the server homomorphically evaluates the server-side conversion of the RtF framework, securely obtaining $\mathsf{Enc}^{\mathsf{CKKS}}(\mathbf{m})$.

Given a symmetric cipher with low multiplicative depth and complexity, a transciphering framework provides the following advantages on the client-side.

- A client does not need to encrypt all its data using an HE algorithm (except the symmetric key). All the data can be encrypted using only a symmetric cipher, significantly saving computational resources in terms of time and memory.
- Symmetric encryption does not result in ciphertext expansion, so the communication overload between the client and the server will be significantly low compared to using any homomorphic encryption scheme alone.

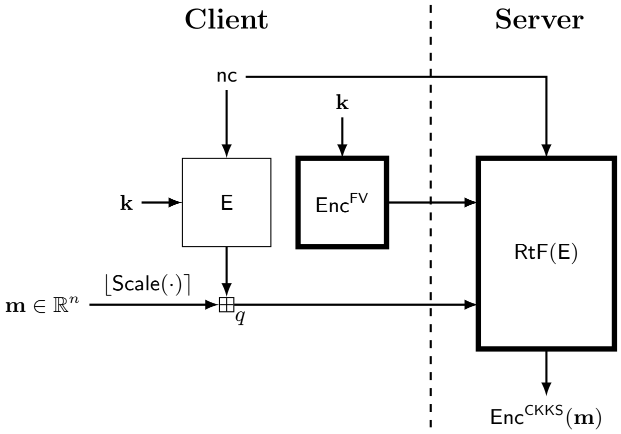


Fig. 1: A simplified diagram of the RtF transciphering framework. Homomorphic operations are performed in the boxes with thick lines.

All these merits come at the cost of computational overload on the server-side. That said, this trade-off would be worth considering in practice since servers are typically more powerful than clients.

Although a transciphering framework can be considered at any place where HE is used, it is not a panacea for every privacy problem since it takes more time than HE-only until $\mathsf{Enc}^{\mathsf{HE}}(\mathbf{m})$ is finally obtained. We suggest two appropriate scenarios for transciphering frameworks in Appendix A.

HE-friendly Ciphers. Symmetric ciphers are built on top of linear and nonlinear layers, and in a conventional environment, there has been no need to take different design principles for the two types of layers with respect to their implementation cost. However, when a symmetric cipher is combined with BGV/FV-style HE schemes [18,34] in a transciphering framework, homomorphic addition becomes way cheaper than homomorphic multiplication in terms of computation time and noise growth. With this observation, the efficiency of an HE-friendly cipher is evaluated by its multiplicative complexity and depth. In an arithmetic circuit, its multiplicative complexity is represented by the number of multiplications (ANDs in the binary case). Multiplicative depth is the depth of the tree that represents the arithmetic circuit, closely related to the noise growth in the HE-ciphertexts. These two metrics have brought a new direction in the design of symmetric ciphers: to use simple nonlinear layers at the cost of highly randomized linear layers as adopted in the design of FLIP [53] and Rasta [27].

1.2 Our Contribution

Designing a symmetric cipher can be seen as a trade-off between security and efficiency. A designer should identify important cost metrics of the targeted platform (e.g., x86, ARM, and HE), and focus on optimizing them within a given security level. When it comes to HE-friendly ciphers, one of the most important cost metrics is the time for evaluating the cipher while homomorphically encrypted, typically translated to multiplicative depth and complexity in the literature. To optimize such metrics, quadratic S-boxes [6], random linear layers [53], and nonlinear layers with high-degree inverses [27] have been used.

In this regard, the LWE encryption has promising properties as an HE-friendly cipher since it is based on a linear combination of key material, while noise prevents algebraic attacks. However, straightforward application of the LWE encryption has a disadvantage on the client-side; in the LWE encryption, $(\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + e)$ is sampled from an LWE distribution, where $\mathbf{a}$ should be freshly generated by a pseudorandom function for every encryption. It makes the LWE encryption too costly on the client-side compared to conventional symmetric encryption.

In this work, we propose a new HE-friendly cipher, dubbed Rubato, as a cost-effective trade-off between the LWE encryption and conventional symmetric encryption in a transciphering framework for approximate homomorphic encryption. In particular, when the RtF transciphering framework [25] is used, we can add noise only with a partial loss of precision. For a *low-degree* keyed function $\mathsf{E}_\mathbf{k} : \mathbb{Z}_q^k \to \mathbb{Z}_q^\ell$, each sample is of the form

$$(\mathbf{a}, \mathsf{E}_\mathbf{k}(\mathbf{a}) + \mathbf{e})$$

where $\mathbf{e} \in \mathbb{Z}_q^\ell$ is sampled from a discrete Gaussian distribution, and $\mathbf{a}$ is generated by an extendable output function (XOF) with a nonce. We remark that such a noisy cipher is not suitable for transciphering of exact data since the server might lose some information on the original message after transciphering. In Table 1, we compare Rubato to existing HE-friendly ciphers operating on modular domains assuming 128-bit security and the same modulus q.

Cipher	Masta	HERA	Pasta	LWE	Rubato
#(Key words)	16	16	64	1024	64
Multiplicative depth	7	10	5	0	2
#(Multiplications)	7	10	9.81	0	2.1
Random bits	400	150	250	25600	80
Source	[42]	[25]	[30]	[59]	**This work**

Table 1: Comparison of HE-friendly ciphers operating on modular domains, where the modulus q is set to 25 bits. "#(Key words)" is the number of key words in $\mathbb{Z}_q$ and "#(Multiplications)" (resp. "Random bits") is the number of multiplications (resp. random bits) required to generate a single component of a ciphertext.

Since Rubato is a combination of a conventional symmetric cipher and the LWE encryption, we analyze its security in two ways: symmetric cryptanalysis and LWE cryptanalysis. We apply the symmetric cryptanalysis by guessing all the noise, while LWE cryptanalysis is considered by linearizing monomials to new variables. From extensive analysis, we recommend a set of parameters for various applications.

Our implementation of Rubato combined with the RtF transciphering framework can be found in a public repository both for the client side[3] and the server side[4]. When Rubato and HERA are compared in the RtF framework, client-side and server-side throughput is improved by 22.9% and 32.2%, respectively, at the cost of only 1.6% larger ciphertext expansion.

1.3 Related Work

Since the transciphering framework has been introduced [55], early works have been focused on the homomorphic evaluation of popular symmetric ciphers (e.g., AES [38], SIMON [48], and PRINCE [31]). Such ciphers have been designed without any consideration of their arithmetic complexity, so the performance of their homomorphic evaluation was not satisfactory.

LowMC [6], being the first HE-friendly cipher, aims to minimize the depth and the number of AND gates, but its low multiplicative depth makes it vulnerable to algebraic attacks [26,28,58]. Due to these attacks, its parameters have been updated, and the resulting cipher is now called LowMCv3. Canteaut et al. [19] claimed that stream ciphers would be advantageous in terms of online complexity compared to block ciphers, and proposed a new stream cipher Kreyvium. However, its practical relevance is limited since the multiplicative depth (with respect to the secret key) keeps growing as keystreams are generated.

The FLIP stream cipher [53] is based on a novel design strategy that its permutation layer is randomly generated for every encryption without increasing the algebraic degree in its secret key. Furthermore, it has been reported that FiLIP [52], a generalized instantiation of FLIP, can be efficiently evaluated with the TFHE scheme [44]. Rasta [27] is a stream cipher aiming at higher throughput at the cost of high latency using random linear layers, which are generated by an extendable output function. Dasta [43], a variant of Rasta using affine layers with lower entropy, boosts up the client-side computation. Masta [42], another variant of Rasta operating on a modular domain, improves upon Rasta in terms of the throughput on both the client and server side. Dobraunig et al. [30] formally defined hybrid homomorphic encryption and proposed another variant Pasta of Rasta operating on a modular domain, improving performance upon Masta.

Cho et al. [25] proposed a transciphering framework for approximate homomorphic encryption, called RtF, which is composed of a stream cipher over modular domain and conversion from FV to CKKS. The stream cipher HERA was proposed in the same paper as a building block of the RtF framework. The HERA cipher is based on a new design strategy – the key schedule is randomized while linear layers are fixed – which is claimed to be efficient on both sides.

[3] https://github.com/KAIST-CryptLab/Rubato

[4] https://github.com/KAIST-CryptLab/RtF-Transciphering

In order to reduce the ciphertext expansion when encrypting short messages, Chen et al. [21] proposed an efficient LWEs-to-RLWE conversion method which enables transciphering to the HE-ciphertexts (including CKKS): small messages can be encrypted by LWE-based symmetric encryption with a smaller ciphertext size (compared to RLWE-based encryption), and a collection of LWE ciphertexts can be repacked to an RLWE ciphertext to perform a homomorphic evaluation. Lu et al. [50] proposed a faster LWEs-to-RLWE conversion algorithm in a hybrid construction of FHEW [32] and CKKS, dubbed PEGASUS, where the conversion is possible for a larger number of slots.

2 Preliminaries

2.1 Notations

Throughout the paper, bold lowercase letters (resp. bold uppercase letters) denote vectors (resp. matrices). For a real number r, $\lfloor r \rceil$ denotes the nearest integer to r, rounding upwards in case of a tie. For an integer q, we identify $\mathbb{Z}_q$ with $\mathbb{Z} \cap (-q/2, q/2]$, and for any real number z, $[z]_q$ denotes the mod q reduction of z into $(-q/2, q/2]$. The notation $\lfloor \cdot \rceil$ and $[\cdot]_q$ are extended to vectors (resp. polynomials) to denote their component-wise (resp. coefficient-wise) reduction. For a complex vector $\mathbf{x}$, its ℓ_p-norm is denoted by $\|\mathbf{x}\|_p$. When we say ℓ_p-norm of a polynomial, it means that the ℓ_p-norm of the coefficient vector of the polynomial. For a measurable subset $S \subset \mathbb{R}^d$, $\mathrm{vol}(S)$ is the volume of S.

Usual dot products of vectors are denoted by $\langle \cdot, \cdot \rangle$. We denote the multiplicative group of $\mathbb{Z}_q$ by $\mathbb{Z}_q^\times$. The set of strings of arbitrary length over a set S is denoted by S^*. For two vectors (strings) $\mathbf{a}$ and $\mathbf{b}$, their concatenation is denoted by $\mathbf{a}\|\mathbf{b}$. For a set S, we will write $a \leftarrow S$ to denote that a is chosen from S uniformly at random. For a probability distribution $\mathcal{D}$, $a \leftarrow \mathcal{D}$ denotes that a is sampled according to the distribution $\mathcal{D}$. Unless stated otherwise, all logarithms are to the base 2.

2.2 Lattice Background

Let $\mathbf{B} \in \mathbb{R}^{m \times n}$ be a full rank matrix. The lattice $L(\mathbf{B})$ generated by $\mathbf{B}$ is defined by $L(\mathbf{B}) = \{\mathbf{B} \cdot \mathbf{x} : \mathbf{x} \in \mathbb{Z}^n\}$. The matrix $\mathbf{B}$ is called a basis of $L(\mathbf{B})$. The i-th successive minimum $\lambda_i(L)$ of a lattice L is the smallest value t such that at least i linearly independent lattice vectors of length $\leq t$ exist in L. The shortest vector problem (SVP) is finding a shortest non-zero vector of L from a given basis. The γ-unique shortest vector problem (γ-uSVP) is finding the shortest non-zero vector of L provided that $\lambda_2(L) > \gamma\lambda_1(L)$. The γ-bounded distance decoding (BDD_γ) problem is finding a lattice point in L closest to a target vector $\mathbf{t}$ provided that $\mathrm{dist}(\mathbf{t}, L) := \min_{\mathbf{x} \in L} \mathrm{dist}(\mathbf{t}, \mathbf{x}) \leq \gamma \cdot \lambda_1(L)$.

Hermite Factor. Given an n-dimensional lattice L with a basis $\mathbf{B}$, a *root-Hermite factor* δ of the basis $\mathbf{B}$ is defined by $\delta^{n-1} = \|\mathbf{b}_1\|/(\det(L(\mathbf{B})))^{1/n}$ where

$\mathbf{b}_1$ is the shortest vector of the basis and $\det(L(\mathbf{B})) = \sqrt{\mathbf{B}^\intercal \mathbf{B}}$. If δ is smaller, then the basis includes a shorter vector in the lattice.

Gaussian Heuristic. Gaussian heuristic (GH) is a heuristic on how many lattice points are contained in a *nice* object. Given a measurable set $S \subset \mathbb{R}^n$ and a full-rank lattice $L \subset \mathbb{R}^n$, the number of lattice points of L in S is approximated by $\#(S \cap L) = \mathrm{vol}(S)/\det(L)$. If S is an n-dimension ball of radius R, then the equation becomes $\#(S \cap L) = v_n R^n / \det(L)$ where v_n is the volume of the unit n-ball. With this heuristic, the norm of the shortest vector of L can be approximated by

$$GH(L) = \left(v_n^{-1} \det(L)\right)^{\frac{1}{n}}.$$

For random lattices, GH is precise within the error at most 5% [37].

Geometric Series Assumption. Schnorr claimed that the Gram-Schmidt orthogonalized norm of a BKZ-reduced basis behaves as a geometric series, which is called geometric series assumption (GSA) [61]. For a BKZ-reduced basis $\mathbf{B} = [\mathbf{b}_1, \cdots, \mathbf{b}_n]$ and its orthogonalization $\mathbf{B}^* = [\mathbf{b}_1^*, \cdots, \mathbf{b}_n^*]$, it satisfies that $\|\mathbf{b}_1^*\| / \|\mathbf{b}_i^*\| = r^{i-1}$ for all $1 \le i \le n$ where r is a constant.

2.3 Learning with Errors

Let n and q be positive integers. Let χ be a probability distribution over $\mathbb{Z}$. For an unknown vector $\mathbf{s} \in \mathbb{Z}_q^n$, the LWE (learning with errors) distribution $L_{\mathbf{s},\chi}$ over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ is obtained by sampling a vector $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ and an error $e \leftarrow \chi$, and outputting

$$(\mathbf{a}, b = [\langle \mathbf{a}, \mathbf{s} \rangle + e]_q) \in \mathbb{Z}_q^n \times \mathbb{Z}_q.$$

The search-LWE problem is to find $\mathbf{s} \in \mathbb{Z}_q^n$ when independent samples $(\mathbf{a}_i, b_i)$ are obtained according to $L_{\mathbf{s},\chi}$. The decision-LWE problem is to distinguish the distribution $L_{\mathbf{s},\chi}$ from the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

For a positive real $\alpha > 0$, the discrete Gaussian distribution $D_{\alpha q}$ is a probability distribution on $\mathbb{Z}$ defined by

$$\Pr\left[y \leftarrow D_{\alpha q} : y = x\right] \propto \exp\left(-\pi x^2/(\alpha q)^2\right)$$

for each $x \in \mathbb{Z}$. The discrete Gaussian distribution is a popular candidate of the distribution χ.

2.4 RtF Transciphering Framework

We briefly introduce the RtF framework [25], which enables the transciphering of approximate data. The RtF framework works as follows. On the client-side, a real message vector $\mathbf{m} \in \mathbb{R}^n$ is scaled up and rounded off into $\mathbb{Z}_q$. Then, the client encrypts the scaled message $\widetilde{\mathbf{m}} \in \mathbb{Z}_q^n$ using a stream cipher E over $\mathbb{Z}_q$. This "E-ciphertext" will be sent to the server with a nonce nc and an FV-encrypted secret key $\mathcal{K}$ of E.

On the server-side, it first evaluates the stream cipher E homomorphically from nonces $\{\mathsf{nc}_i\}_i$ and the FV-encrypted key $\mathcal{K}$. Then the server performs the linear transformation $\mathsf{SlotToCoeff}^{\mathsf{FV}}$, obtaining the resulting FV-ciphertext $\mathcal{Z}$ that contains the keystreams of E in its coefficients. This process is called the offline phase since evaluating $\mathcal{Z}$ is possible only with nonces and $\mathcal{K}$.

After receiving E-ciphertexts $\{\mathbf{c}_i = \mathsf{E}_{\mathbf{k}}(\widetilde{\mathbf{m}}_i)\}_i$, the server starts its online phase. Computing an FV-ciphertext $\mathcal{C}$ having the E-ciphertexts on its coefficients and subtracting $\mathcal{Z}$ from $\mathcal{C}$, the server obtains the FV-ciphertext $\mathcal{X}$ of $\{\widetilde{\mathbf{m}}_i\}_i$ in its coefficients. Finally, the server CKKS-bootstraps $\mathcal{X}$ to translate it into the corresponding CKKS-ciphertext of $\{\mathbf{m}_i\}_i$ in its slots. Since the messages $\{\mathbf{m}_i\}_i$ should be moved from the coefficients to the slots, the last step of the bootstrapping, $\mathsf{SlotToCoeff}^{\mathsf{CKKS}}$, can be omitted. As a result, the server will be able to approximately evaluate any circuit on the CKKS-ciphertexts. The detailed description of the RtF framework can be found in Appendix B.

3 Rubato: A Family of Noisy Ciphers

3.1 Specification

The Rubato cipher is designed to be flexible in block size so that it offers a more suitable choice of parameters for various applications. The block size n is the square of a positive integer v, which defines the size of matrices in linear layers. The stream cipher Rubato for λ-bit security takes as input a symmetric key $\mathbf{k} \in \mathbb{Z}_q^n$, a nonce $\mathsf{nc} \in \{0,1\}^\lambda$, and returns a keystream $\mathbf{k}_{\mathsf{nc}} \in \mathbb{Z}_q^\ell$ for some $\ell < n$, where the nonce is fed to the underlying extendable output function (XOF) that outputs an element in $(\mathbb{Z}_q^n)^*$. In a nutshell, Rubato is defined as follows.

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

where the i-th round function $\mathsf{RF}[\mathbf{k},\mathsf{nc},i]$ is defined as

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

and the final round function Fin is defined as

$$\mathsf{Fin}[\mathbf{k},\mathsf{nc},r] = \mathsf{Tr}_{n,\ell} \circ \mathsf{ARK}[\mathbf{k},\mathsf{nc},r] \circ \mathsf{MixRows} \circ \mathsf{MixColumns} \circ \mathsf{Feistel} \circ \mathsf{MixRows} \circ \mathsf{MixColumns}$$

for $i = 1, 2, \ldots, r-1$ (see Figure 2).

Key Schedule. The round key schedule can be simply seen as a component-wise product between random values and the master key $\mathbf{k}$, where the uniformly random values in $\mathbb{Z}_q^\times$ are obtained from a certain extendable output function XOF with an input nc. Given a sequence of the outputs from XOF, say $\mathbf{rc} = (\mathbf{rc}_0, \ldots, \mathbf{rc}_r) \in (\mathbb{Z}_q^n)^{r+1}$, ARK is defined as follows.

$$\mathsf{ARK}[\mathbf{k},\mathsf{nc},i](\mathbf{x}) = \mathbf{x} + \mathbf{k} \bullet \mathbf{rc}_i$$

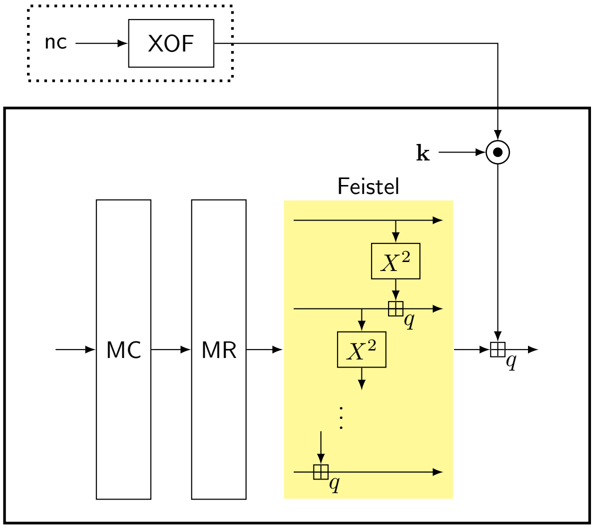


Fig. 2: The round function of Rubato. Operations in the box with dotted (resp. thick) lines are public (resp. secret). "MC" and "MR" represent MixColumns and MixRows, respectively.

for $i = 0, \dots, r$, and $\mathbf{x} \in \mathbb{Z}_q^n$, where $\bullet$ (resp. $+$) denotes component-wise multiplication (resp. addition) modulo q. The extendable output function XOF might be instantiated with a sponge-type hash function SHAKE [33].

Linear Layers. Each linear layer is the composition of MixColumns and MixRows. Similarly to HERA, MixColumns (resp. MixRows) multiplies a certain $v \times v$ MDS matrix $\mathbf{M}_v$ to each column (resp. row) of the state as in Figure 4a and Figure 4b, where the state of Rubato is also viewed as a $v \times v$-matrix over $\mathbb{Z}_q$ (see Figure 3). The MDS matrix $\mathbf{M}_v$ for $v = 4, 6, 8$ is defined as follows.

$$\begin{aligned}
\mathbf{y}_4 &= [2, 3, 1, 1] \\
\mathbf{y}_6 &= [4, 2, 4, 3, 1, 1] \\
\mathbf{y}_8 &= [5, 3, 4, 3, 6, 2, 1, 1] \\
\mathbf{M}_v &= \begin{bmatrix} \mathbf{y}_v \\ \mathrm{ROT}^1(\mathbf{y}_v) \\ \vdots \\ \mathrm{ROT}^{v-1}(\mathbf{y}_v) \end{bmatrix}
\end{aligned}$$

where $\mathrm{ROT}^i(\mathbf{y})$ is the rotation to the right of $\mathbf{y}$ by i components. Therefore, $\mathbf{M}_v$ is a circulant matrix derived from $\mathbf{y}_v$.

Nonlinear Layers. The nonlinear map Feistel is a Feistel network in a row, which was proposed in [30]. For $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{Z}_q^n$, we have

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

$$\begin{array}{|c|c|c|c|}
\hline
x_{1,1} & x_{1,2} & \cdots & x_{1,v} \\
\hline
x_{2,1} & x_{2,2} & \cdots & x_{2,v} \\
\hline
\vdots & \vdots & \ddots & \vdots \\
\hline
x_{v,1} & x_{v,2} & \cdots & x_{v,v} \\
\hline
\end{array}$$

Fig. 3: State of Rubato. Each square stands for the component in $\mathbb{Z}_q$.

$$\begin{bmatrix} y_{1,c} \\ y_{2,c} \\ \vdots \\ y_{v,c} \end{bmatrix} = \mathbf{M}_v \cdot \begin{bmatrix} x_{1,c} \\ x_{2,c} \\ \vdots \\ x_{v,c} \end{bmatrix}$$

(a) MixColumns

$$\begin{bmatrix} y_{c,1} \\ y_{c,2} \\ \vdots \\ y_{c,v} \end{bmatrix} = \mathbf{M}_v \cdot \begin{bmatrix} x_{c,1} \\ x_{c,2} \\ \vdots \\ x_{c,v} \end{bmatrix}$$

(b) MixRows

Fig. 4: Definition of MixColumns and MixRows. For $c \in \{1, 2, \ldots, v\}$, x_{ij} and y_{ij} are defined as in Figure 3.

It is naturally bijective and of degree 2.

Truncation. The truncation function $\mathsf{Tr}_{n,\ell} : \mathbb{Z}_q^n \to \mathbb{Z}_q^\ell$ is just a truncation of the last $n - \ell$ words. For $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{Z}_q^n$, we have

$$\mathsf{Tr}_{n,\ell}(\mathbf{x}) = (x_1, \ldots, x_\ell).$$

Although we know that the truncation function makes some part of the last ARK and MixRows meaningless, we write it in this way for brevity. We recommend to instantiate $\mathsf{Tr}_{n,\ell} \circ \mathsf{ARK}[\mathbf{k}, \mathsf{nc}, r] \circ \mathsf{MixRows}$ as a whole in real implementation.

Adding Gaussian Noise. At the very last of the cipher, we add Gaussian noise to every component. From an one-dimensional discrete Gaussian distribution $D_{\alpha q}$ with zero mean and variance $(\alpha q)^2/2\pi$, we sample ℓ elements $e_1, \ldots, e_\ell \leftarrow D_{\alpha q}$ independently. For $\mathbf{x} = (x_1, \ldots, x_\ell) \in \mathbb{Z}_q^\ell$, we have

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

Encryption Mode. When a keystream of k blocks (in $(\mathbb{Z}_q^\ell)^k$) is needed for some $k > 0$, the “inner-counter mode” can be used; for $\mathsf{ctr} = 0, 1, \ldots, k-1$, one computes

$$\mathbf{z}[\mathsf{ctr}] = \mathsf{Rubato}\left[\mathbf{k}, \mathsf{nc}\|\mathsf{ctr}\right](\mathbf{ic}),$$

where $\mathbf{ic}$ denotes a constant $(1, 2, \ldots, n) \in \mathbb{Z}_q^n$. For a given message vector $\mathbf{m} \in (\mathbb{R}^\ell)^k$, encryption by Rubato is defined by

$$\mathbf{c} = \lfloor \Delta \cdot \mathbf{m} \rceil + \mathbf{z} \pmod{q}$$

where $\Delta \in \mathbb{R}$ is a scaling factor.

3.2 Parameter Selection

In this section, we recommend some sets of parameters and concrete instantiation of Rubato. Some sets of parameters are selected in Table 2. The notations in the table follow those in Section 3.1. We give three types of parameters: S, M, and L. These imply the size of blocks.

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

Table 2: Selected parameters of Rubato.

When choosing the modulus q, we consider the effect of noise on precision. For a discrete Gaussian distribution $D_{\alpha q}$, the size of noise is expectedly $\mathbb{E}_{e \leftarrow D_{\alpha q}}[\,|e|\,]$. Suppose we obtain p-bit average precision while using the RtF framework with some deterministic cipher (e.g., HERA [25]). It means that, a given message x and the message after transciphering x', $|x - x'| < 1/2^p$. Then, the expected loss of precision bits is upper bounded by

$$p + \log_2 \left[\mathbb{E}_{e \leftarrow D_{\alpha q}} [1/2^p + e/\varDelta] \right].$$

In our instantiation, we enlarge the modulus q to compensate this loss of precision.

The choice of the scaling factor $\varDelta$ should vary along with the $\|\mathbf{m}\|_1$, where $\mathbf{m} \in \mathbb{R}^*$ is a message vector. In our experiment (see Section 5), we constrain $\|\mathbf{m}\|_1 \leq 1$ and choose $\varDelta = q/16$ for the RtF framework [25]. If someone manipulates a message $\|\mathbf{m}\|_1 \leq s$, it is appropriate to choose $\varDelta = q/(16 \cdot s)$.

3.3 Design Rationale

The main observation behind our design is that adding noise increases the algebraic degree of a cipher. Suppose that we are given LWE samples $\{(\mathbf{a}_i, b_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i)\}_i$. In Arora-Ge attack [9], an attacker establishes an equation

$$\prod_{e=-t\alpha q}^{t\alpha q} (b_i - \langle \mathbf{a}_i, \mathbf{s} \rangle - e) = 0$$

in order to solve the LWE instance, where $t \in \mathbb{R}$ determines the adversarial success probability. In this way, the noisy linear equation becomes a polynomial equation of degree $(2t\alpha q+1)$. If the linear equation $\langle \mathbf{a}, \mathbf{s}\rangle$ is replaced by a polynomial $F(\mathbf{a}, \mathbf{s})$ of degree d, the Arora-Ge equation becomes of degree $d(2t\alpha q+1)$.

We choose the discrete Gaussian distribution for sampling noise since the cryptanalysis of LWE has been extensively studied under the discrete Gaussian assumption. In the main body of the stream cipher, we use building blocks from HERA [25] and Pasta [30]. For linear layers and the key schedule, we follow the style of HERA. Although we are aware of generic ways of constructing an MDS matrix [41,40], those approaches result in a matrix with large components. We keep the component of matrices $\mathbf{M}_v$ as small as possible for efficiency. When enlarging the block size n, we computationally find $v \times v$ MDS matrices since we cannot keep the original linear layer of HERA.

For nonlinear layers, Cho et al. [25] claimed that a nonlinear layer whose inverse is of a high degree mitigates algebraic MitM attacks. As there has not been any known quadratic function with the inverse of a high degree over $\mathbb{Z}_q$, a cubic S-box has been used in HERA, which leads to a larger multiplicative depth. After truncation was proposed for an alternative countermeasure for an algebraic MitM attack [29], Dobraunig et al. [30] proposed a Feistel structure for HE-friendly ciphers. Since the Feistel structure is vulnerable to algebraic MitM attacks, a cubic function for the last nonlinear layer and truncation are adopted to Pasta. As we thought that deploying both the cubic function and truncation is superfluous, we conclude that truncation without the cubic function is sufficient for Rubato.

4 Security Analysis

In this section, we provide the security analysis of Rubato. We summarize the analysis result in Table 3. We omit too costly attacks (i.e., time complexity larger than 2^{1000} for all the parameters) such as trivial linearization and interpolation attacks. We computed the complexity of each attack by using Wolfram Mathematica and made the source codes publicly available in our repository[5]. In Appendix E, we give some additional plots on the security analysis.

Assumptions and the Scope of Analysis. In this work, we will consider the standard "secret-key model", where an adversary arbitrarily chooses a nonce and obtains the corresponding keystream without any information on the secret key. The related-key and the known-key models are beyond the scope of this paper.

Since Rubato takes as input counters, an adversary is not able to control the differences of the inputs. Nonces can be adversarially chosen, while they are also fed to the extendable output function, which is modeled as a random oracle. So one cannot control the difference of the internal variables. For this reason, we believe that our construction is secure against any type of chosen-plaintext attacks including (higher-order) differential, truncated differential, invariant subspace trail, and cube attacks. A recent generalization of an integral attack [15]

[5] https://github.com/KAIST-CryptLab/Rubato

Parameter	GCD	Gröbner	LC	Lattice	Arora-Ge
`Par-80S`	393.6	80.04	155.9	760.5	80.04
`Par-80M`	878.6	84.55	249.9	↑	80.37
`Par-80L`	↑	82.73	349.8	↑	82.73
`Par-128S`	411.9	128.1	311.7	↑	128.1
`Par-128M`	880.7	128.1	249.9	↑	128.1
`Par-128L`	↑	169.6	349.8	↑	129.6

Table 3: The log of the complexity of the attacks on Rubato. The upward sign (↑) implies that the complexity is larger than 2^{1000}. The linear algebra constant ω is assumed to be 2.

requires only a small number of chosen plaintexts, while it is not applicable to Rubato within the security bound.

4.1 Cryptanalysis Based on Symmetric Primitive Analysis

Most of the symmetric cryptanalysis assumes that a targeted cipher is a deterministic algorithm. Symmetric cryptanalysis is to find some statistical or algebraic characteristics of the function which is distinguished from its ideal counterpart. However, as the Gaussian noise is added at the end of the cipher, Rubato should be seen as a random sampling. For this reason, most of the conventional symmetric cryptanalyses are not directly applicable to Rubato. Nevertheless, by guessing all the noise, an attacker can try to analyze Rubato using symmetric key cryptanalysis. Since the noise is sampled from discrete Gaussian distribution $D_{\alpha q}$, it is always advantageous for an attacker to guess that the noise is zero when the data are sufficiently given. We denote the probability such that a sample from $D_{\alpha q}$ is zero by $\varepsilon_0 = \Pr\left[e \leftarrow D_{\alpha q} : e = 0\right]$.

4.1.1 Trivial Linearization

Trivial linearization is to solve a system of linear equations by replacing all monomials with new variables. When applied to the r-round Rubato cipher, the number of monomials appearing in this system is upper bounded by

$$S = \sum_{i=0}^{2^r} \binom{n+i-1}{i}.$$

Therefore, at most S equations will be enough to solve this system of equations. All the monomials of degree at most 2^r are expected to appear after r rounds of Rubato (as explained in detail in Appendix C). Therefore, by guessing $e = 0$, we can conclude that this attack requires $O(S^\omega/\varepsilon_0^S)$ time, where $2 \le \omega \le 3$. Since the success probability is too small for $r \ge 1$, it will never be a dominant attack.

4.1.2 GCD Attack

The GCD attack seeks to compute the greatest common divisor (GCD) of univariate polynomials, and it can be useful for a cipher operating on a large field with its representation being a polynomial in a single variable. This attack can be extended to a system of multivariate polynomial equations by guessing all the key variables except one. For r-round Rubato, the complexity of the GCD attack is estimated as $O(q^{n-1}r^2 2^r)$ even if there is no noise. For a security parameter $\lambda \leq 256$, Rubato will be secure against the GCD attack even with a single round as long as $n \geq 16$.

4.1.3 Gröbner Basis Attack

The Gröbner basis attack is an attack by solving a system of equations by computing a Gröbner basis of the system. If such a Gröbner basis is found, then the variables can be eliminated one by one after carefully converting the order of monomials. We refer to [8] for details. In the literature, security against Gröbner basis attack is bounded by the time complexity for Gröbner basis computing.

Suppose that an attacker wants to solve a system of m polynomial equations in n variables over a field $\mathbb{F}_q$,

$$f_1(x_1, \ldots, x_n) = f_2(x_1, \ldots, x_n) = \cdots = f_m(x_1, \ldots, x_n) = 0.$$

The complexity of computing a Gröbner basis of such system is known to be

$$O\left(\binom{n + d_{reg}}{d_{reg}}^{\omega}\right)$$

in terms of the number of operations over the base field, where $2 \leq \omega \leq 3$ and d_{reg} is the *degree of regularity* [14]. With the degree of regularity, one can see how many degrees of polynomial multiples will be needed to find the Gröbner basis. Unfortunately, it is hard to compute the exact degree of regularity for a generic system of equations. When the number of equations is larger than the number of variables, the degree of regularity of a *semi-regular sequence* can be computed as the degree of the first non-positive coefficient in the Hilbert series

$$\mathrm{HS}(z) = \frac{1}{(1-z)^n} \times \prod_{i=1}^{m} (1 - z^{d_i}).$$

As it is conjectured that most sequences are semi-regular [35], we analyze the security of Rubato against the Gröbner basis attack under the (semi-)regular assumption.

Hybrid Approach. One can take a hybrid approach between the guess-and-determine attack and the algebraic attack [13]. Guessing some variables makes the system of equations overdetermined. An overdetermined system becomes easier to solve; the complexity of the hybrid approach after g guesses is given as

$$O\left(q^g \binom{n - g + d_g}{d_g}^{\omega}\right)$$

where d_g is the degree of regularity after g guesses.

Application to Rubato. For the Gröbner basis attack, re-arranging equations may lead to a significant impact on the attack complexity. For example, one may set a system of equations using only plaintext-ciphertext pairs or set a system of equations with new variables standing for internal states. The former will be a higher-degree system in fewer variables, while the latter will be a lower-degree system in more variables.

From a set of nonce-plaintext-ciphertext triples $\{(\mathsf{nc}_i, \mathbf{m}_i, \mathbf{c}_i)\}_i$, an attacker will be able to establish an overdetermined system of equation

$$f_1(k_1, \ldots, k_n) = f_2(k_1, \ldots, k_n) = \cdots = f_m(k_1, \ldots, k_n) = 0$$

where $k_i \in \mathbb{Z}_q$ is the i-th component of the key variable. The degree of regularity of the system is computed as the degree of the first non-positive coefficient in

$$\left(1 - z^{2^r}\right)^{m-n} \left(\sum_{i=0}^{2^r-1} z^i\right)^n$$

where r is the number of rounds. The larger number of equations implies a smaller degree of regularity. Since the summation does not have any negative term, one easily sees that the degree d_{reg} of regularity cannot be smaller than 2^r. We conservatively lower bound the time complexity when there is no noise by

$$O\left(\binom{n+2^r}{2^r}^{\omega}\right)$$

regardless of the number of equations. Since at least n equations are required for the unique root, we can conclude that this attack requires n data and

$$O\left(\binom{n+2^r}{2^r}^{\omega} \varepsilon_0^{-n}\right)$$

time. We note that the hybrid approach always has worse complexity.

Instead of a system of equations of degree 2^r, one can establish a system of $((r-1)n + \ell)k$ quadratic equations in $n(r-1)k + n$ variables, where k is the block length of each query. To get the unique root, it requires that $k \geq n/\ell$. Then, the complexity is

$$O\left(\binom{n(r-1)k + n + d_{reg}(r,k)}{d_{reg}(r,k)}^{\omega} \varepsilon_0^{-\ell k}\right)$$

where the degree $d_{reg}(r,k)$ of regularity is computed under the semi-regular assumption.

Although we explain that the truncation can prevent MitM attack, MitM attack is not a "never-applicable" attack for Rubato. Suppose $\mathbf{y} = (y_1, \ldots, y_\ell)$ be a keystream. By creating new variables $x_{\ell+1}, \ldots, x_n$, an attacker can make n MitM equations in $2n - \ell$ variables $k_1, \ldots, k_n, x_{\ell+1}, \ldots, x_n$. Denoting the first

$\lfloor r/2 \rfloor$-round function by F and the last $\lceil r/2 \rceil$-round function except for $\mathsf{Tr}_{n,\ell}$ and AGN by G,

$$\begin{aligned}\mathsf{Rubato}[\mathbf{k}, \mathsf{nc}_1] &= (y_1, \dots, y_\ell)\\ G \circ F[\mathbf{k}, \mathsf{nc}_1] &= (y_1 - e_1, \dots, y_\ell - e_\ell, x_{\ell+1}, \dots, x_n)\\ F[\mathbf{k}, \mathsf{nc}_1] &= G^{-1}(y_1 - e_1, \dots, y_\ell - e_\ell, x_{\ell+1}, \dots, x_n) \qquad (1)\end{aligned}$$

where e_i's are guessed noise. Equation 1 is of degree $2^{\lceil r/2 \rceil}$ so that the lower bound of the degree of regularity is also $2^{\lceil r/2 \rceil}$. Similarly as above, to get the unique root, the queried block length k should satisfy $n + k(n - \ell) \leq nk$. Then, the time complexity is lower bounded by

$$O\left(\binom{n + (n-\ell)k + 2^{\lceil r/2 \rceil}}{2^{\lceil r/2 \rceil}}^{\omega} \varepsilon_0^{-\ell k}\right).$$

4.1.4 Interpolation Attack

The interpolation attack is to establish an encryption polynomial in plaintext variables without any information on the secret key and to distinguish it from a random permutation [46]. It is known that the data complexity of this attack depends on the number of monomials in the polynomial representation of the cipher.

For the r-round Rubato cipher, let $\mathbf{rc} = (\mathbf{rc}_0, \dots, \mathbf{rc}_r) \in (\mathbb{Z}_q^n)^{r+1}$ be a sequence of the outputs from XOF. For $i = 0, \dots, r$, $\mathbf{rc}_i$ is evaluated by a polynomial of degree 2^{r-i}. As we expect that the r-round Rubato cipher has almost all monomials of degree $\leq 2^r$ in its polynomial representation, the number of monomials is lower bounded by

$$\sum_{j=0}^{r} \sum_{i=0}^{2^j} \binom{n+i-1}{i}.$$

Similarly as the trivial linearization, the success probability is too small for $r \geq 1$, it will never be a dominant attack.

4.1.5 Linear Cryptanalysis

Linear cryptanalysis was originally introduced for binary spaces [51], but it can also be applied to non-binary spaces [11]. Similarly to binary ciphers, for an odd prime number q, the linear probability of a cipher $\mathsf{E} : \mathbb{Z}_q^n \to \mathbb{Z}_q^n$ with respect to input and output masks $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_q^n$ can be defined by

$$\mathrm{LP}^{\mathsf{E}}(\mathbf{a}, \mathbf{b}) = \left| \mathbb{E}_{\mathbf{m}} \left[\exp\left\{ \frac{2\pi i}{q} \Big(-\langle \mathbf{a}, \mathbf{m} \rangle + \langle \mathbf{b}, \mathsf{E}(\mathbf{m}) \rangle \Big) \right\} \right] \right|^2$$

where $\mathbf{m}$ follows the uniform distribution on $\mathbb{Z}_q^n$. When E is a random permutation, the expected linear probability is defined by $\mathrm{ELP}^{\mathsf{E}}(\mathbf{a}, \mathbf{b}) = \mathbb{E}_{\mathsf{E}}[\mathrm{LP}^{\mathsf{E}}(\mathbf{a}, \mathbf{b})]$.

Then, the number of samples required for linear cryptanalysis is known to be $1/\mathrm{ELP}^{\mathsf{E}}(\mathbf{a},\mathbf{b})$. In order to ensure the security against linear cryptanalysis, it is sufficient to bound the maximum linear probability $\max_{\mathbf{a}\neq 0,\mathbf{b}} \mathrm{ELP}^{\mathsf{E}}(\mathbf{a},\mathbf{b})$.

Application to Rubato. Although it seems that the linear cryptanalysis cannot be applied to Rubato directly because of the noise, we give a security bound for linear cryptanalysis assuming no noise. There are two applications of linear cryptanalysis on Rubato according to how to take the input variables: the XOF output variables or the key variables. In the first case, unlike traditional linear cryptanalysis, the probability of any linear trail of Rubato depends on the key since it is multiplied by the input. It seems infeasible to make a plausible linear trail without any information on the key material.

In the second case, the attack is reduced to solving an LWE-like problem as follows; given pairs $(\mathsf{nc}_i, \mathbf{y}_i)$ such that $\mathsf{Rubato}(\mathbf{k}, \mathsf{nc}_i) = \mathbf{y}_i$, one can establish

$$\langle \mathbf{b}, \mathbf{y}_i \rangle = \langle \mathbf{a}, \mathbf{k} \rangle + e_i$$

for some vectors $\mathbf{a} \neq \mathbf{0}, \mathbf{b} \in \mathbb{Z}_q^n$ and error e_i sampled according to a certain distribution χ. An attacker requires $1/\mathrm{ELP}^{\mathsf{E}}(\mathbf{a},\mathbf{b})$ samples to distinguish χ from the uniform distribution [11].

Lemma 1. *For any* $\mathbf{a} = (a_1, \ldots, a_n) \neq \mathbf{0}, \mathbf{b} = (b_1, \ldots, b_n) \in \mathbb{Z}_q^n$ *such that* $\mathsf{hw}(b_2, b_3, \ldots, b_n) = h$*, the linear probability of* Feistel *is*

$$\mathrm{LP}^{\mathsf{Feistel}}(\mathbf{a},\mathbf{b}) \leq \frac{1}{q^h}.$$

Proof. By the definition, we have

$$\begin{aligned}
&\mathrm{LP}^{\mathsf{Feistel}}(\mathbf{a},\mathbf{b}) \\
&= \left| \mathbb{E}_{\mathbf{m}} \left[\exp\left\{ \frac{2\pi i}{q} \left(-\langle \mathbf{a}, \mathbf{m} \rangle + \langle \mathbf{b}, \mathsf{Feistel}(\mathbf{m}) \rangle \right) \right\} \right] \right|^2 \\
&= \left| \mathbb{E}_{\mathbf{m}} \left[\exp\left\{ \frac{2\pi i}{q} \left(\sum_{k=1}^{n-1} (-a_k m_k + b_k m_k + b_{k+1} m_k^2) + (-a_n + b_n) m_n \right) \right\} \right] \right|^2 \\
&= \left| \mathbb{E}_{m_n} \left[\exp\left\{ \frac{2\pi i}{q} \left((-a_n + b_n) m_n \right) \right\} \right] \right|^2 \\
&\quad \times \prod_{i=1}^{n-1} \left| \mathbb{E}_{m_i} \left[\exp\left\{ \frac{2\pi i}{q} \left((-a_i + b_i) m_i + b_{i+1} m_i^2 \right) \right\} \right] \right|^2.
\end{aligned}$$

Carlitz and Uchiyama [20] proved that

$$\left| \sum_{x=0}^{q-1} \exp\left(\frac{2\pi i}{q} \cdot p(x) \right) \right| \leq (r-1)\sqrt{q}$$

for any polynomial $p(x)$ of degree r over $\mathbb{Z}_q$. Therefore, we have

$$\left| \mathbb{E}_{m_i} \left[\exp\left\{ \frac{2\pi i}{q} \left((-a_i + b_i) m_i + b_{i+1} m_i^2 \right) \right\} \right] \right|^2 \leq \left| \frac{1}{q} \cdot \sqrt{q} \right|^2 \leq \frac{1}{q}$$

and it implies that

$$\mathrm{LP}^{\mathsf{Feistel}}(\mathbf{a},\mathbf{b}) \leq \frac{1}{q^h}. \qquad \square$$

Since the branch number of the linear layer of Rubato is $2v$ (as shown in Appendix D), we can conclude that an r-round Rubato cipher provides λ-bit security against linear cryptanalysis when $q^{(2v-2)\cdot\lfloor\frac{r}{2}\rfloor} > 2^{\lambda}$.

4.1.6 Differential Cryptanalysis and Its Variants

Since Rubato takes counters as input, an adversary is not able to control the differences of its inputs. Nonces can be adversarially chosen, while they are also fed to the extendable output function, which is modeled as a random oracle. So one cannot control the difference of the internal variables. For this reason, we believe that our construction will be secure against any type of chosen-plaintext attack including (higher-order) differential, truncated differential, invariant subspace trail, and cube attacks.

Nonetheless, to prevent an unsuspected differential-related attack, we present a computation of a differential characteristic in the following. Given a pair $\mathbf{a},\mathbf{b} \in \mathbb{Z}_q^n$, the differential probability of Feistel is defined by

$$\mathrm{DP}^{\mathsf{Feistel}}(\mathbf{a},\mathbf{b}) = \frac{1}{q^n}\cdot\left|\{\mathbf{x}\in\mathbb{Z}_q^n : \mathsf{Feistel}(\mathbf{x}+\mathbf{a}) - \mathsf{Feistel}(\mathbf{x}) = \mathbf{b}\}\right|.$$

So $\mathrm{DP}^{\mathsf{Feistel}}(\mathbf{a},\mathbf{b})$ is determined by the number of solutions to $\mathsf{Feistel}(\mathbf{x}+\mathbf{a}) - \mathsf{Feistel}(\mathbf{x}) = \mathbf{b}$.

Lemma 2. *For any* $\mathbf{a} = (a_1,\ldots,a_n) \neq \mathbf{0}, \mathbf{b} = (b_1,\ldots,b_n) \in \mathbb{Z}_q^n$ *such that* $\mathsf{hw}(a_1,a_2,\ldots,a_{n-1}) = h$*, the differential probability of* Feistel *is*

$$\mathrm{DP}^{\mathsf{Feistel}}(\mathbf{a},\mathbf{b}) \leq \frac{1}{q^h}.$$

Proof. Our goal is to find the maximum number of solutions to the equation

$$\mathsf{Feistel}(\mathbf{x}+\mathbf{a}) - \mathsf{Feistel}(\mathbf{x}) = (a_1, 2a_1x_1 + a_1^2 + a_2, \ldots, 2a_nx_{n-1} + a_{n-1}^2 + a_n) = \mathbf{b}.$$

For $i \leq n-1$, the equation $2a_ix_i + a_i^2 + a_{i+1} = b_i$ has a unique solution $x_i = (b_i - a_i^2 - a_{i+1})\cdot(2a_i)^{-1}$ if $a_i \neq 0$ and the equation has maximally q solutions if $a_i = 0$. For $i = n$, the variable x_n is free so that the maximal number of solution is q. It implies that

$$\mathrm{DP}^{\mathsf{Feistel}}(\mathbf{a},\mathbf{b}) \leq \frac{1}{q^h}. \qquad \square$$

Since the branch number of the linear layer of Rubato is $2v$ (as shown in Appendix D), we can conclude that an r-round Rubato cipher provides λ-bit security against differential cryptanalysis when $q^{(2v-2)\cdot\lfloor\frac{r}{2}\rfloor} > 2^{\lambda}$.

4.2 Cryptanalysis Based on LWE Analysis

As Rubato is not an LWE instance, algorithms solving LWE are not directly applied to Rubato. However, if someone considers a single component of a keystream block of Rubato as

$$\left((a_{\mathbf{u}})_{\mathbf{u}}, \sum_{\mathbf{u}} a_{\mathbf{u}}\mathbf{k}^{\mathbf{u}} + e\right) \tag{2}$$

where $\mathbf{u} = (u_1, \ldots, u_n) \in \mathbb{Z}^n_{\geq 0}$, and $\mathbf{k}^{\mathbf{u}} = \prod_i k_i^{u_i}$ implies a monomial with degree $\mathbf{u}$, it becomes an LWE instance with the linearized variables whose dimension is

$$S_{n,r} = \sum_{i=1}^{2^r} \binom{n+i-1}{i}$$

where $2^r < q$.

In this section, we will denote notations in a linearized way. For example, we denote Rubato samples by $(\mathbf{A}, \mathbf{c} = \mathbf{A}\mathbf{s} + \mathbf{e})$ where $\mathbf{s}$ stands for the vector $(\mathbf{k}^{\mathbf{u}})_{\mathbf{u}}$ and $\mathbf{A}$ stands for a set of $(a_{\mathbf{u}})_{\mathbf{u}}$ in a certain monomial order.

We remark that we do not explore potential vulnerabilities which can arise from combining symmetric key cryptanalysis and LWE cryptanalysis. We analyze each attack in its original way, not in a mixed way. For example, in our analysis, all the LWE cryptanalysis except Arora-Ge attack [9] assume that $(a_{\mathbf{u}})_{\mathbf{u}}$ is independently sampled from the uniform distribution over $\mathbb{Z}_q$, which is not the case for Rubato.

4.2.1 Exhaustive Search

The most naive approach for solving LWE is the exhaustive search. Given m samples $(\mathbf{A}, \mathbf{c})$, an attacker guesses noise $\mathbf{e} = (e_1, \ldots, e_m)$ and finds $\mathbf{s}$ satisfying $\mathbf{A}\mathbf{s} = \mathbf{c} - \mathbf{e}$ where $\mathbf{A}$ is required to have a left inverse. To attack Rubato, the attacker needs to guess at least $(2t\alpha q + 1)^{S_{n,r}}$ times for success probability $\Pr\left[e \leftarrow D_{\alpha q} : |e_i| \leq t\alpha q \text{ for all } i\right]$ where the expected time complexity is upper bounded by $\varepsilon_0^{S_{n,r}}$. Since the success probability is too small for $r \geq 1$, it will never be a dominant attack.

There is a meet-in-the-middle (MitM) approach mentioned in [10], which is a time-memory trade-off of the exhaustive search. For the same reason, the MitM approach cannot be a dominant attack.

4.2.2 Lattice Attacks

Reduction to a lattice problem is one way to solve LWE. To solve a lattice problem, an attacker needs a short enough basis of the given lattice. This short basis is obtained by using a lattice reduction algorithm such as the BKZ algorithm [60,23].

Core-SVP Hardness of BKZ Algorithms. The BKZ algorithm is a lattice reduction algorithm that uses an (approximate-)SVP oracle of small dimension

β. This algorithm repeatedly calls the SVP oracle as a subroutine to find the shortest vectors in the projected lattice of dimension β. An output from the BKZ-β algorithm is called a "BKZ$_\beta$-reduced basis". The SVP oracle can be instantiated using sieving algorithms or enumeration algorithms.

Unfortunately, it is difficult to predict how many calls will be made to the SVP oracle in the BKZ algorithm. So, we analyze the security of Rubato against the BKZ algorithm using a single call, in which case the underlying hardness assumption is called *core-SVP hardness* [7]. Table 4 compares the expected time complexity of the BKZ algorithm for various instantiations of the SVP oracle in terms of BKZ block size β and root-Hermite factor δ. For Lindner and Peikert [49] and Albrecht et al. [3], the time complexity is estimated by extrapolating their experimental running time of the BKZ algorithm using enumeration methods. For the remaining instantiations, the complexity analysis is theoretically based on the cost of a single call to the SVP-oracle.

When it comes to the quality of a BKZ$_\beta$-reduced basis, Chen [22] gave a limit

$$\lim_{N\to\infty} \delta = \left(v_\beta^{-\frac{1}{\beta}}\right)^{\frac{1}{\beta-1}} \approx \left(\frac{\beta}{2\pi e}(\pi\beta)^{\frac{1}{\beta}}\right)^{\frac{1}{2(\beta-1)}} \tag{3}$$

for the root-Hermite factor δ assuming the Gaussian heuristic and the geometric series assumption. Chen also gave an experimental proof that this limit is a reasonable choice when N is finite. As another estimate of δ for a BKZ$_\beta$-reduced basis, the *lattice rule of thumb* [57], which says $\delta = \beta^{\frac{1}{2\beta}}$, is often used in the literature. We will opt for Chen's limit when we compute β from a fixed value of δ.

Instantiation of the SVP oracle	Complexity (in log)
Lindner and Peikert [49]	$\frac{1.8}{\log\delta} - 110$
Albrecht et al. [3]	$\frac{0.009}{\log^2\delta} - 27$
Enumeration [2]	$\frac{\beta\log\beta}{8} - 0.654\beta + 25.84$
Classical Sieve [12]	$0.292\beta + o(\beta)$
Quantum Sieve [47]	$0.265\beta + o(\beta)$

Table 4: Expected time complexity of the BKZ algorithm for various instantiations of the SVP oracle in terms of BKZ block size β and root-Hermite factor δ.

Primal Attack. Primal attack is the strategy of solving the search-LWE problem via solving the bounded distance decoding (BDD) problem. Given m samples $(\mathbf{A}, \mathbf{c} = \mathbf{As} + \mathbf{e})$ following $L_{\mathbf{s},\chi}$, one can see that $\mathbf{c}$ is near the lattice $L(\mathbf{A})$. Finding the nearest lattice point from $\mathbf{c}$ is equivalent to finding the secret vector $\mathbf{s}$ when $\mathbf{A}$ is (left) invertible. If $\mathbf{A}$ is not invertible, it is sufficient to gather a few more samples.

In order to solve the derived BDD problem, there are two approaches: enumeration [49,37] and reduction to unique-SVP (uSVP) [4,5]. Since the enumeration method is treated as a subroutine in the BKZ algorithm, we do not take it into account as a direct solver of the BDD problem.

The second approach, the reduction to uSVP, was firstly proposed by Albrecht et al. [4]. The main idea of the approach is to solve SVP of the larger lattice $L = L(\mathbf{B})$ of basis

$$\mathbf{B} = \begin{pmatrix} \mathbf{A} & \mathbf{c} \\ \mathbf{0} & u \end{pmatrix}$$

where $u = \mathrm{dist}(\mathbf{c}, L(\mathbf{A}))$. This lattice contains an unusually small vector $(\mathbf{e}, -u)$, which implies the gap $\lambda_2(L)/\lambda_1(L)$ is large. Assuming Gaussian heuristic and linear independence of $\mathbf{A}$, Göpfert [39] showed that an attacker can create the $\lambda_2(L)/\lambda_1(L)$-gap greater than

$$\frac{\min\left\{q, q^{1-N/m}\sqrt{\frac{m}{2\pi e}}\right\}}{\sqrt{m}\cdot\frac{\alpha q}{\sqrt{2\pi}}}.$$

As a lattice reduction satisfying $\lambda_2(L)/\lambda_1(L) > \tau\delta^m$ for some constant $\tau \leq 1$ is sufficient to solve a uSVP instance [36], this approach requires log root-Hermite factor

$$\log\delta = \frac{\log^2(\tau\alpha\sqrt{e})}{4N\log q}$$

if $\min\left\{q, q^{1-N/m}\sqrt{\frac{m}{2\pi e}}\right\} = q^{1-N/m}\sqrt{\frac{m}{2\pi e}}$. Although experimental evidence suggests $\tau \leq 0.4$ [36], we set $\tau = 1$ for the conservative choice of parameters.

Alkim et al. gave an alternative success condition of the attack [7]. Denoting $d = m + N + 1$ and $\sigma = \alpha q/\sqrt{2\pi}$, the requirement is that

$$\sigma\sqrt{\beta} \leq \delta^{2\beta-d} q^{m/d}$$

where δ is computed by Equation 3. We take both into account along with the parameter $N = S_{n,r}$.

Dual Attack. The dual attack, also called the short integer solution (SIS) strategy, is an attack finding small vector $\mathbf{w} \in \mathbb{Z}_q^m$ such that $\mathbf{w}^\intercal\mathbf{A} \equiv 0 \pmod{q}$. Given m samples $(\mathbf{A}, \mathbf{c})$ from $L_{\mathbf{s},\chi}$, finding a short vector satisfying $\mathbf{w}^\intercal\mathbf{A} \equiv \mathbf{0} \pmod{q}$ leads to

$$\mathbf{w}^\intercal\mathbf{c} = \mathbf{w}^\intercal(\mathbf{A}\mathbf{s} + \mathbf{e}) = \mathbf{w}^\intercal\mathbf{e}$$

where the last term is small. The short vector $\mathbf{w}$ should satisfy

$$\|\mathbf{w}\|_2 = \frac{1}{\alpha}\sqrt{\frac{\ln(1/\epsilon)}{\pi}}$$

in order to distinguish $L_{\mathbf{s},\chi}$ from random with advantage ϵ [49]. By the definition of the root-Hermite factor δ, the attack requires that

$$\log\delta = \frac{\log^2\left(\frac{1}{\alpha}\sqrt{\frac{\ln(1/\epsilon)}{\pi}}\right)}{4N\log q}$$

for the LWE instance parametrized by N, α, and q. When evaluating the security of Rubato, we set $N = S_{n,r}$ and $\epsilon \approx 1/23$.

4.2.3 BKW Attack

The original BKW algorithm was proposed for solving the learning parity with noise (LPN) problem [16] by Blum, Kalai, and Wasserman. Regev pointed out that the BKW algorithm can be used for solving LWE, and Albrecht et al. [3] gave the formal analysis of the BKW algorithm for LWE. We briefly explain the BKW attack on LWE, and we refer to [3] for more details.

The BKW attack is a lattice-version of Gaussian elimination parametrized by a and b. Suppose there is an LWE distribution $L_{\mathbf{s},\chi}$ where $\chi = D_{\alpha q}$ is parametrized by the dimension N and the modulus q. Given enough samples $(\mathbf{A}, \mathbf{c})$ from $L_{\mathbf{s},\chi}$, the BKW attack first reduces $\mathbf{A}$ to a kind of block diagonal matrix. The width of the block is b and there will be an $a = \lceil N/b \rceil$ blocks.

As samples from $L_{\mathbf{s},\chi}$ intrinsically include noise, too many additions or subtractions between samples result in a useless equation. Instead, the attacker gathers $(q^b - 1)/2$ samples having all nonzero values up to sign in the first block. Denote this table of samples by T^1. If $L_{\mathbf{s},\chi}$ outputs a sample $(\mathbf{a}_i, c_i)$ which has the same first block in T^1, the attacker makes $(\mathbf{a}_i, c_i)$ to have all-zero first block by adding/subtracting the sample in T^1. Similarly, the attacker can gather $(q^b - 1)/2$ samples having all-zero first block and all nonzero values up to sign in the second block. Denote this table of samples by T^2. By repeating the same process, the attacker can construct a matrix of the form

$$\begin{pmatrix}
T^1 & & & & & & & & & & & \\
0 & \dots & 0 & T^2 & & & & & & & & \\
0 & \dots & 0 & 0 & \dots & 0 & T^3 & & & & & \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & & & & & & \\
0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 0 & T^{a-1} & & \\
0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 0 & 0 & T^a & \\
0 & \dots & 0 & 0 & \dots & 0 & 0 & \dots & 0 & \dots & 0 & M
\end{pmatrix}.$$

After solving a lattice problem with respect to M, the attacker can perform back substitution for the remaining parts.

The time complexity of this attack to solve search-LWE is

$$\left(\frac{q^b - 1}{2}\right) \cdot \left(\frac{a(a-1)}{s} \cdot (N+1)\right) + \left\lceil \frac{q^b}{2} \right\rceil \cdot \left(\left\lceil \frac{N}{d} \right\rceil + 1\right) \cdot d \cdot a + \mathrm{poly}(N)$$

where $d = N - \lfloor N/b \rfloor$. The parameter a should satisfy $a \leq \log(\alpha^{-2})$ in order to distinguish $L_{\mathbf{s},\chi}$ from random [57]. We compute the concrete complexity by using $a = \log(\alpha^{-2})$ without the polynomial terms. As $N = S_{n,r}$ for Rubato, the complexity of the attack is at least $2^{S_{n,r} \log q / (-2 \log \alpha)}$.

4.2.4 Arora-Ge Attack

Arora and Ge proposed an algebraic algorithm to solve the search-LWE problem [9]. The main idea of this attack is that, given LWE samples $\{(\mathbf{a}_i, b_i)\}_i$, the errors fall into some interval $[-t\alpha q, t\alpha q]$ for some large enough t so that the equations

$$\prod_{e=-t\alpha q}^{t\alpha q} (b_i - \langle \mathbf{a}_i, \mathbf{s} \rangle - e) = 0$$

holds. Although the complexity of this attack for LWE is well-organized in [1], we independently describe the lower bound of complexity as the equations are different from LWE.

When guessing noise, an attacker may control the range of guesses to minimize the cost of attacks since the noise is not uniformly distributed. We denote the probability such that a sample from discrete Gaussian $D_{\alpha q}$ lies in the interval $[-t\alpha q, t\alpha q]$ by

$$\varepsilon_t = \Pr\left[e \leftarrow D_{\alpha q} : |e| \leq t\alpha q\right].$$

Since this probability determines the complexity of many attacks, we find the minimum of the complexity among $\{t : -3\sqrt{2\pi} \leq t \leq 3\sqrt{2\pi}\}$ which is equivalent to the range of 6 times the standard deviation.

Let $\mathsf{E}_i(\cdot)$ denote the i-th component of the Rubato cipher without noise. Then, by using Arora-Ge attack, an attacker can make a set of equations as follows. Given m nonce-plaintext-ciphertext triples $\{(\mathsf{nc}_j, \mathbf{m}_j, \mathbf{c}_j)\}_j$,

$$\left\{ \prod_{e_{i,j}=-t\alpha q}^{t\alpha q} (\mathbf{c}_{j,i} - \mathsf{E}_i(\mathsf{nc}_j, \mathbf{m}_j) - e_{i,j}) = 0 \right\}_{\substack{1 \leq i \leq \ell \\ 1 \leq j \leq m}} \tag{4}$$

where $\mathbf{c}_{j,i}$ is the i-th component of $\mathbf{c}_j$. These equations are for the key variable $\mathbf{k}$ of total degree $2^r(2t\alpha q + 1)$.

Now, we give a lower bound of the complexity of solving Equation 4 by using Gröbner basis attack. As discussed in Section 4.1.3, we can conservatively assume the degree of regularity of Equation 4 is $2^r(2t\alpha q + 1)$ regardless of the number of nonce-plaintext-ciphertext triples. We have the time complexity at least

$$O\left(\binom{n + 2^r(2t\alpha q + 1)}{2^r(2t\alpha q + 1)}^{\omega} \varepsilon_t^{-c}\right)$$

where c should be larger than or equal to n to get the unique root. This complexity formula also lower bounds the trivial linearization approach to solving Equation 4.

Meet-in-the-middle Approach. Similar to most of the algebraic attacks, one can try to use the MitM approach for Arora-Ge attack. From Equation 1, the attacker can build the Arora-Ge equations as follows.

$$\left\{ \prod_{(e_{i,j})_i \in C_j} \left(F_i[\mathbf{k}, \mathsf{nc}_j] - (G^{-1})_i(y_1 - e_{1,j}, \ldots, y_\ell - e_{\ell,j}, x_{\ell+1}, \ldots, x_n)\right) = 0 \right\}_{1 \leq j \leq m}$$

where

$$C_j \coloneqq \left\{(e_1, \ldots, e_\ell) \in \mathbb{Z}^\ell : -t\alpha q \le e_i \le t\alpha q \;\text{ for all } i\right\},$$

F_i and $(G^{-1})_i$ are the i-th components of F and G^{-1} respectively. These equations are of degree $2^{\lceil r/2 \rceil}(2t\alpha q + 1)^\ell$ so that the lower bound of the degree of regularity also is $2^{\lceil r/2 \rceil}(2t\alpha q + 1)^\ell$. Similarly as above, to get the unique root, the queried block length k should satisfy $n + k(n - \ell) \le nk$. Then, the time complexity is lower bounded by

$$O\left(\binom{n + (n-\ell)k + 2^{\lceil r/2 \rceil}(2t\alpha q + 1)^\ell}{2^{\lceil r/2 \rceil}(2t\alpha q + 1)^\ell}^{\omega} \varepsilon_t^{-\ell k}\right).$$

We give some plots of the complexity of the Arora-Ge attack according to the choice of t in Appendix E.

5 Performance Evaluation

In this section, we evaluate the performance of the RtF framework combined with the Rubato cipher in terms of encryption speed and ciphertext expansion. The source codes of server-side computation are developed in Golang version 1.16.4 with Lattigo library[6] which implements RNS (residue number system) variants of the FV and the CKKS schemes. For the HE parameters, we use the RtF parameter Par-128a in [25], which uses the arcsin function. For completeness, we summarize the HE parameters in Appendix F. The source codes of client-side computation are developed in C++17, using GNU C++ 7.5.0 compiler with AVX2 instruction set. For the instantiation of the XOF, we use AES128 in counter-mode as well as SHAKE256 in openssl library[7] and XKCP library[8], respectively. Our experiments are done in AMD Ryzen 7 2700X @ 3.70 GHz single-threaded with 64 GB memory.

5.1 Benchmark and Comparison

We measure the performance of the RtF framework along with Rubato, distinguishing two different parts: the client-side and the server-side. The client-side latency includes time for generating pseudorandom numbers (needed to generate a single keystream in $\mathbb{Z}_q^n$), keystream generation from Rubato, message scaling, rounding, and vector addition over $\mathbb{Z}_q$. Since generating pseudorandom numbers from XOF takes significant time on the client-side, we measure the client-side performance according to the instantiations of the XOF.

On the server-side, the latency is divided into offline and online phases as described in Section 2.4. The offline latency includes time for randomized key schedule, homomorphic evaluation of the keystreams from Rubato, and the linear

[6] https://github.com/ldsec/lattigo

[7] https://github.com/openssl/openssl

[8] https://github.com/XKCP/XKCP

transformation SlotToCoeff$^{\mathsf{FV}}$. The online latency includes computing the FV-ciphertext containing the symmetric ciphertexts in its coefficients, homomorphic subtraction, and the modified CKKS-bootstrapping process in the RtF framework (called HalfBoot). We measure the latency until the first HE-ciphertext comes out, while the throughput is measured until all the n HE-ciphertexts come out. Because the XOF running time does not affect the server-side performance as significantly as it does on the client-side, we only use SHAKE256 instantiation for a fair comparison with previous results. We note that our evaluation does not take into account key encryption since the encrypted key will be used over multiple sessions once it is computed. For the same reason, the initialization process of the HE schemes is not considered.

Set	AES128		SHAKE256	
	Latency (cycle)	Throughput (C/B)	Latency (cycle)	Throughput (C/B)
Par-80S	2154	72.63	5906	199.1
Par-80M	3644	49.36	11465	143.5
Par-80L	4957	32.97	16679	110.9
Par-128S	3076	103.6	10446	351.8
Par-128M	4381	55.10	14292	179.7
Par-128L	5323	35.70	16920	113.5

Table 5: Client-side performance of the RtF transciphering framework with Rubato.

Set	CER	Latency		Throughput	Precision
		Offline (s)	Online (s)	(KB/s)	(bits)
Par-80S	1.31	21.48	19.75	6.676	18.8
Par-80M	1.25	37.44	19.71	7.032	19.0
Par-80L	1.25	85.65	19.79	6.520	19.1
Par-128S	1.31	50.78	20.28	6.083	18.8
Par-128M	1.26	68.47	19.88	6.666	18.9
Par-128L	1.26	86.34	20.09	6.712	18.9

Table 6: Server-side performance of the RtF transciphering framework with Rubato.

We summarize our implementation results in Table 5 and Table 6. In Table 5, the client-side latency and throughput for each instantiation of the XOF are given. Table 6 includes ciphertext expansion ratio (CER), time-relevant measurements, and precision. One can see that a larger parameter implies higher throughput at the cost of higher latency on both sides. As Rubato needs a substantial amount of random bits, the client-side performance is significantly influenced by the choice of XOF. On the server-side, we note that Rubato only affects the offline latency while the online latency is affected by the efficiency of CKKS bootstrapping.

Comparison. We compare the result of `Par-128L` to the recent implementation of HERA [25], LWEs-to-RLWE conversions [50], and CKKS itself. The comparison is summarized in Table 7. The result of HERA is obtained from the paper. The source codes of LWEs-to-RLWE conversion are taken from the `OpenPegasus` library[9]. As `OpenPegasus` library does not include symmetric LWE encryption, we implement (seeded) symmetric LWE encryption with AVX2-optimized SHAKE256. We use `Lattigo` library for CKKS encryption.

In this table, the security parameter λ is set to 128. For the fairness of comparison, the remaining levels after transciphering are all set to be 7. For all experiments, we sample the domain of each component of the message vector from the uniform distribution over $(-1, 1)$. When computing the ciphertext expansion ratio, we use the formula $\log q/(p+1)$, which excludes the effect of sending a public nonce. Multiple use of different nonces can be dealt with a counter so that the effect of a nonce to the ratio is asymptotically zero.

Since the `OpenPegasus` library supports only selected sets of parameters in terms of the number of slots and the ciphertext modulus (at the point of submission), we implemented LWEs-to-RLWE for $N = 2^{16}$ and $\ell = 2^{10}$ which does not provide exactly the same functionality as ours with full available slots.

One can see that Rubato with the RtF framework outperforms HERA with respect to the both-side throughput, while it has a worse CER and ciphertext size compared to HERA. Also, Rubato outperforms the LWEs-to-RLWE conversion with respect to CER, ciphertext size and client-side performance, achieving the main purpose of the transciphering framework. Compared to the CKKS-only environment, Rubato with the RtF framework has better CER and client-side performance, while the CKKS-only environment requires no additional computation.

References

1. Albrecht, M., Cid, C., Faugère, J.C., Fitzpatrick, R., Perret, L.: On the Complexity of the Arora-Ge Algorithm against LWE. In: SCC 2012 – Third international conference on Symbolic Computation and Cryptography. pp. 93–99 (Jul 2012)
2. Albrecht, M.R., Bai, S., Li, J., Rowell, J.: Lattice Reduction with Approximate Enumeration Oracles. In: Malkin, T., Peikert, C. (eds.) Advances in Cryptology – CRYPTO 2021. pp. 732–759. Springer (2021)
3. Albrecht, M.R., Cid, C., Faugère, J.C., Fitzpatrick, R., Perret, L.: On the Complexity of the BKW Algorithm on LWE. Designs, Codes and Cryptography **74**(2), 325–354 (Feb 2015)

[9] https://github.com/Alibaba-Gemini-Lab/OpenPEGASUS

Scheme	N	ℓ	Ctxt. Exp.		Client		Server		p
			Ctxt. (KB)	Ratio	Lat. (μs)	Thrp. (MB/s)	Lat. (s)	Thrp. (KB/s)	
RtF-Rubato	2^{16}	2^{16}	0.183	1.26	4.585	31.04	106.4	6.712	18.9
RtF-HERA [25]	2^{16}	2^{16}	0.055	1.24	1.520	25.26	141.58	5.077	19.1
LWE [50]	$2^{16}(2^{10})$	2^{10}	0.007	4.84	21.91	0.051	65.88	0.010	9.3
CKKS	2^{14}	2^{14}	468	23.25	9656	2.035	none		19.1

Table 7: Comparison of the RtF transciphering framework with Rubato to previous environments supporting homomorphic encryption of approximate numbers. All the experiments are done with 128-bit security. Parameter N in parentheses implies the dimension of LWE. The parameter p stands for the bits of precision.

4. Albrecht, M.R., Fitzpatrick, R., Göpfert, F.: On the Efficacy of Solving LWE by Reduction to Unique-SVP. In: Lee, H.S., Han, D.G. (eds.) Information Security and Cryptology – ICISC 2013. pp. 293–310. Springer (2014)
5. Albrecht, M.R., Göpfert, F., Virdia, F., Wunderer, T.: Revisiting the Expected Cost of Solving uSVP and Applications to LWE. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology – ASIACRYPT 2017. pp. 297–322. Springer (2017)
6. Albrecht, M.R., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for MPC and FHE. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015. vol. 9056, pp. 430–454. Springer (2015)
7. Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-Quantum Key Exchange: A New Hope. p. 327–343. SEC'16, USENIX Association, USA (2016)
8. Aly, A., Ashur, T., Ben-Sasson, E., Dhooghe, S., Szepieniec, A.: Design of Symmetric-Key Primitives for Advanced Cryptographic Protocols. IACR Transactions on Symmetric Cryptology **2020**(3) (Sep 2020)
9. Arora, S., Ge, R.: New Algorithms for Learning in Presence of Errors. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) Automata, Languages and Programming. pp. 403–415. Springer (2011)
10. Bai, S., Galbraith, S.D.: Lattice Decoding Attacks on Binary LWE. In: Susilo, W., Mu, Y. (eds.) Information Security and Privacy. pp. 322–337. Springer (2014)
11. Baignères, T., Stern, J., Vaudenay, S.: Linear Cryptanalysis of Non Binary Ciphers. In: Adams, C., Miri, A., Wiener, M. (eds.) Selected Areas in Cryptography. vol. 4876, pp. 184–211. Springer (2007)
12. Becker, A., Ducas, L., Gama, N., Laarhoven, T.: New Directions in Nearest Neighbor Searching with Applications to Lattice Sieving. In: Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms. pp. 10–24. SIAM (2016)
13. Bettale, L., Faugere, J.C., Perret, L.: Hybrid Approach for Solving Multivariate Systems over Finite Fields. Journal of Mathematical Cryptology **3**(3), 177–197 (2009)
14. Bettale, L., Faugère, J.C., Perret, L.: Solving Polynomial Systems over Finite Fields: Improved Analysis of the Hybrid Approach. In: Proceedings of the 37th International Symposium on Symbolic and Algebraic Computation. ISSAC '12, Association for Computing Machinery (2012)

15. Beyne, T., Canteaut, A., Dinur, I., Eichlseder, M., Leander, G., Leurent, G., Naya-Plasencia, M., Perrin, L., Sasaki, Y., Todo, Y., Wiemer, F.: Out of Oddity – New Cryptanalytic Techniques Against Symmetric Primitives Optimized for Integrity Proof Systems. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology – CRYPTO 2020. vol. 12172, pp. 299–328. Springer (2020)
16. Blum, A., Kalai, A., Wasserman, H.: Noise-Tolerant Learning, the Parity Problem, and the Statistical Query Model. J. ACM **50**(4), 506–519 (Jul 2003)
17. Boura, C., Gama, N., Georgieva, M., Jetchev, D.: Simulating Homomorphic Evaluation of Deep Learning Predictions. In: Dolev, S., Hendler, D., Lodha, S., Yung, M. (eds.) Cyber Security Cryptography and Machine Learning. vol. 11527, pp. 212–230. Springer (2019)
18. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) Fully Homomorphic Encryption without Bootstrapping. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference. p. 309–325. ACM (2012)
19. Canteaut, A., Carpov, S., Fontaine, C., Lepoint, T., Naya-Plasencia, M., Paillier, P., Sirdey, R.: Stream ciphers: A Practical Solution for Efficient Homomorphic-Ciphertext Compression. Journal of Cryptology **31**(3), 885–916 (2018)
20. Carlitz, L., Uchiyama, S.: Bounds for Exponential Sums. Duke mathematical Journal **24**(1), 37–41 (1957)
21. Chen, H., Dai, W., Kim, M., Song, Y.: Efficient Homomorphic Conversion Between (Ring) LWE Ciphertexts. In: Sako, K., Tippenhauer, N.O. (eds.) Applied Cryptography and Network Security. pp. 460–479. Springer (2021)
22. Chen, Y.: Réduction de Réseau et Sécurité Concrète du Chiffrement Complètement Homomorphe. Ph.D. thesis (2013), thèse de doctorat dirigée par Nguyen, Phong-Quang Informatique Paris 7 2013
23. Chen, Y., Nguyen, P.Q.: BKZ 2.0: Better Lattice Security Estimates. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology – ASIACRYPT 2011. pp. 1–20. Springer (2011)
24. Cheon, J.H., Kim, A., Kim, M., Song, Y.: Homomorphic Encryption for Arithmetic of Approximate Numbers. In: Takagi, T., Peyrin, T. (eds.) Advances in Cryptology – ASIACRYPT 2017. vol. 10624, pp. 409–437. Springer (2017)
25. Cho, J., Ha, J., Kim, S., Lee, B., Lee, J., Lee, J., Moon, D., Yoon, H.: Transciphering framework for approximate homomorphic encryption. In: Tibouchi, M., Wang, H. (eds.) Advances in Cryptology – ASIACRYPT 2021. vol. 13092, pp. 640–669. Springer (2021)
26. Dinur, I., Liu, Y., Meier, W., Wang, Q.: Optimized Interpolation Attacks on LowMC. In: Iwata, T., Cheon, J.H. (eds.) Advances in Cryptology – ASIACRYPT 2015. vol. 9453, pp. 535–560. Springer (2015)
27. Dobraunig, C., Eichlseder, M., Grassi, L., Lallemand, V., Leander, G., List, E., Mendel, F., Rechberger, C.: Rasta: A Cipher with Low ANDdepth and Few ANDs per Bit. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology – CRYPTO 2018. vol. 10991, pp. 662–692. Springer (2018)
28. Dobraunig, C., Eichlseder, M., Mendel, F.: Higher-Order Cryptanalysis of LowMC. In: Kwon, S., Yun, A. (eds.) Information Security and Cryptology – ICISC 2015. vol. 9558, pp. 87–101. Springer (2016)
29. Dobraunig, C., Grassi, L., Guinet, A., Kuijsters, D.: Ciminion: Symmetric Encryption Based on Toffoli-Gates over Large Finite Fields. In: Canteaut, A., Standaert, F.X. (eds.) Advances in Cryptology – EUROCRYPT 2021. pp. 3–34. Springer (2021)

30. Dobraunig, C., Grassi, L., Helminger, L., Rechberger, C., Schofnegger, M., Walch, R.: Pasta: A Case for Hybrid Homomorphic Encryption. Cryptology ePrint Archive, Report 2021/731 (2021), https://ia.cr/2021/731
31. Doröz, Y., Shahverdi, A., Eisenbarth, T., Sunar, B.: Toward Practical Homomorphic Evaluation of Block Ciphers Using Prince. In: Böhme, R., Brenner, M., Moore, T., Smith, M. (eds.) Financial Cryptography and Data Security. vol. 8438, pp. 208–220. Springer (2014)
32. Ducas, L., Micciancio, D.: FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015. vol. 9056, pp. 617–640. Springer (2015)
33. Dworkin, M.J.: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. Tech. rep., National Institute of Standards and Technology (2015)
34. Fan, J., Vercauteren, F.: Somewhat Practical Fully Homomorphic Encryption. IACR Cryptology ePrint Archive, Report 2012/144 (2012), https://eprint.iacr.org/2012/144
35. Fröberg, R.: An Inequality for Hilbert Series of Graded Algebras. MATHEMATICA SCANDINAVICA **56** (Dec 1985)
36. Gama, N., Nguyen, P.Q.: Predicting Lattice Reduction. In: Smart, N. (ed.) Advances in Cryptology – EUROCRYPT 2008. pp. 31–51. Springer (2008)
37. Gama, N., Nguyen, P.Q., Regev, O.: Lattice Enumeration Using Extreme Pruning. In: Gilbert, H. (ed.) Advances in Cryptology – EUROCRYPT 2010. pp. 257–278. Springer (2010)
38. Gentry, C., Halevi, S., Smart, N.P.: Homomorphic Evaluation of the AES Circuit. In: Safavi-Naini, R., Canetti, R. (eds.) Advances in Cryptology – CRYPTO 2012. vol. 7417, pp. 850–867. Springer (2012)
39. Göpfert, F.: Securely Instantiating Cryptographic Schemes Based on the Learning with Errors Assumption. Ph.D. thesis, Technische Universität, Darmstadt (2016)
40. Grassi, L., Khovratovich, D., Rechberger, C., Roy, A., Schofnegger, M.: Poseidon: A New Hash Function for Zero-Knowledge Proof Systems. In: 30th USENIX Security Symposium (USENIX Security 21). pp. 519–535. USENIX Association (Aug 2021)
41. Guo, J., Peyrin, T., Poschmann, A.: The PHOTON Family of Lightweight Hash Functions. In: Rogaway, P. (ed.) Advances in Cryptology – CRYPTO 2011. pp. 222–239. Springer (2011)
42. Ha, J., Kim, S., Choi, W., Lee, J., Moon, D., Yoon, H., Cho, J.: Masta: An HE-Friendly Cipher Using Modular Arithmetic. IEEE Access **8**, 194741–194751 (2020)
43. Hebborn, P., Leander, G.: Dasta – Alternative Linear Layer for Rasta. IACR Transactions on Symmetric Cryptology **2020**(3), 46–86 (Sep 2020)
44. Hoffmann, C., Méaux, P., Ricosset, T.: Transciphering, Using FiLIP and TFHE for an Efficient Delegation of Computation. In: Bhargavan, K., Oswald, E., Prabhakaran, M. (eds.) Progress in Cryptology – INDOCRYPT 2020. pp. 39–61. Springer International Publishing, Cham (2020)
45. Hong, S., Lee, S., Lim, J., Sung, J., Cheon, D., Cho, I.: Provable Security against Differential and Linear Cryptanalysis for the SPN Structure. In: Goos, G., Hartmanis, J., van Leeuwen, J., Schneier, B. (eds.) Fast Software Encryption – FSE 2000. vol. 1978. Springer (2001)
46. Jakobsen, T., Knudsen, L.R.: The Interpolation Attack on Block Ciphers. In: Biham, E. (ed.) Fast Software Encryption – FSE '97. vol. 1267, pp. 28–40. Springer (1997)
47. Laarhoven, T.: Search Problems in Cryptography: from Fingerprinting to Lattice Sieving. Ph.D. thesis, Mathematics and Computer Science (Feb 2016), proefschrift

48. Lepoint, T., Naehrig, M.: A Comparison of the Homomorphic Encryption Schemes FV and YASHE. In: Pointcheval, D., Vergnaud, D. (eds.) Progress in Cryptology – AFRICACRYPT 2014. vol. 8469, pp. 318–335. Springer (2014)
49. Lindner, R., Peikert, C.: Better Key Sizes (and Attacks) for LWE-Based Encryption. In: Kiayias, A. (ed.) Topics in Cryptology – CT-RSA 2011. pp. 319–339. Springer (2011)
50. Lu, W., Huang, Z., Hong, C., Ma, Y., Qu, H.: PEGASUS: Bridging Polynomial and Non-polynomial Evaluations in Homomorphic Encryption. In: 2021 2021 IEEE Symposium on Security and Privacy (SP). pp. 1057–1073. IEEE Computer Society (May 2021)
51. Matsui, M.: Linear Cryptanalysis Method for DES Cipher. In: Helleseth, T. (ed.) Advances in Cryptology — EUROCRYPT '93. vol. 765, pp. 386–397. Springer (1994)
52. Méaux, P., Carlet, C., Journault, A., Standaert, F.X.: Improved Filter Permutators for Efficient FHE: Better Instances and Implementations. In: Hao, F., Ruj, S., Sen Gupta, S. (eds.) Progress in Cryptology – INDOCRYPT 2019. vol. 11898, pp. 68–91. Springer (2019)
53. Méaux, P., Journault, A., Standaert, F.X., Carlet, C.: Towards Stream Ciphers for Efficient FHE with Low-Noise Ciphertexts. In: Fischlin, M., Coron, J.S. (eds.) Advances in Cryptology – EUROCRYPT 2016. vol. 9665, pp. 311–343. Springer (2016)
54. Mouchet, C., Troncoso-Pastoriza, J.R., Bossuat, J., Hubaux, J.: Multiparty Homomorphic Encryption from Ring-Learning-with-Errors. Proc. Priv. Enhancing Technol. **2021**(4), 291–311 (2021)
55. Naehrig, M., Lauter, K., Vaikuntanathan, V.: Can Homomorphic Encryption be Practical? In: Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop. p. 113–124. ACM (2011)
56. Park, S., Byun, J., Lee, J., Cheon, J.H., Lee, J.: HE-Friendly Algorithm for Privacy-Preserving SVM Training. IEEE Access **8**, 57414–57425 (2020)
57. Player, R.: Parameter Selection in Lattice-based Cryptography. Ph.D. thesis, Royal Holloway, University of London (2018)
58. Rechberger, C., Soleimany, H., Tiessen, T.: Cryptanalysis of Low-Data Instances of Full LowMCv2. IACR Transactions on Symmetric Cryptology **2018**(3), 163–181 (2018)
59. Regev, O.: On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. J. ACM **56**(6) (Sep 2009)
60. Schnorr, C.P., Euchner, M.: Lattice Basis Reduction: Improved Practical Algorithms and Solving Subset Sum Problems. Mathematical Programming **66**(1), 181–199 (Aug 1994)
61. Schnorr, C.P.: Lattice Reduction by Random Sampling and Birthday Methods. In: Alt, H., Habib, M. (eds.) STACS 2003. pp. 145–156. Springer (2003)

A Scenarios for Transciphering

Two-party Scenario. In a client-server model, the main purpose of a transciphering framework is outsourcing computation of the client's data to the server while the privacy of the data is protected against the server. To this end, the symmetric key and the secret key of the HE scheme should be generated by the client. The corresponding evaluation key (and the public key if needed) is initially transferred to the server.

As the current HE schemes still suffer from a large amount of computational overhead, we do not recommend to use HE or a transciphering framework when the function to be evaluated is public and simple. When the server has secret functions to evaluate (e.g., ML inference as a service) or evaluating the function requires too much resource compared to the capacity of the client, using a transciphering framework will be a plausible idea for (oblivious) evaluation of those functions in the following sense.

- When data are supposed to be stored for a long time, a transciphering framework can reduce the memory consumption by the encrypted data.
- When only constrained resources are available on the client side (e.g., Internet of Things), a transciphering framework can alleviate the load of the client in terms of computational power, memory, and entropy gathering.
- When data are not packed in a fixed way, since any (limited-size) collection of symmetric ciphertexts can be transciphered into a single HE ciphertext, a transciphering framework is more forward-compatible to process the data.

Multi-party Scenario. In this scenario, we consider three types of parties: data providers, an evaluator, and a key holder(s). Data providers provide data, and they want data privacy to be protected. An evaluator processes the data from the data providers without knowing the data (as plaintext). Key holders – may be singular or plural – have the secret key of the HE scheme. If there are two or more key holders, then each key holder might have a share of the secret key, while the evaluation and the public keys are generated by multi-party computation. This method is called multi-party homomorphic encryption (MHE) [54]. Assuming no conspiracy of the evaluator and all the key holders, the privacy of the data will be protected. A transciphering framework in the multi-party scenario proceeds as follows.

1. A key holder generates secret key sk, public key pk, and evaluation key evk.
2. The key holder sends pk to the data providers and (pk, evk) to the evaluator.
3. Each data provider generates its own symmetric key $\mathbf{k}_i$, and sends HE-encrypted symmetric key $\mathcal{K}_i = \mathsf{Enc}^{\mathsf{HE}}_{pk}(\mathbf{k}_i)$ to the evaluator.
4. Each data provider encrypts the data using symmetric cipher E, and sends the symmetric ciphertext $\mathbf{c} = \mathsf{E}_{\mathbf{k}_i}(\mathbf{m})$ to the evaluator.
5. The evaluator transciphers $\mathbf{c}$ to $\mathsf{Enc}^{\mathsf{HE}}_{pk}(\mathbf{m})$ using $\mathcal{K}_i$.
6. The evaluator homomorphically processes $\mathsf{Enc}^{\mathsf{HE}}_{pk}(\mathbf{m})$ and sends the result $\mathsf{Enc}^{\mathsf{HE}}_{pk}(f(\mathbf{m}))$ to the key holder.
7. The key holder decrypts $\mathsf{Enc}^{\mathsf{HE}}_{pk}(f(\mathbf{m}))$, obtaining $f(\mathbf{m})$.

This process can be used for secure data collection. The evaluator can collect data from the public domain while preserving the data privacy, and process the data in an offline manner. Similar to the two-party scenario, a transciphering framework might provide better usability compared to the HE-only case, in particular, when a small amount of data are steadily collected for a long time.

B Description of the RtF framework

In this section, we give a detailed description of the RtF framework. The pictorial description of the framework can be found in Figure 5.

Initialization. The RtF transciphering framework uses both FV and CKKS schemes. For a fixed security parameter λ, set parameters such as the degree of the polynomial modulus and the ciphertext moduli, and generate the public-private key pair to satisfy the desired security level λ. For a secret key $\mathbf{k} \in \mathbb{Z}_q^n$ used for a symmetric cipher E, the client computes FV-ciphertext $\mathcal{K}$ of the secret key $\mathbf{k}$ and sends it to the server.

Client-side Computation. Suppose that a nonce $\mathsf{nc} \in \{0,1\}^\lambda$, a secret key $\mathbf{k} \in \mathbb{Z}_q^n$ of the cipher E and a scaling factor $\Delta > 0$ are given. The client encrypts an n-tuple of real messages $\mathbf{m} = (m_0, \ldots, m_{n-1}) \in \mathbb{R}^n$ as follows. It first scales up the message $\mathbf{m}$ by Δ and outputs $\widetilde{\mathbf{m}} \in \mathbb{Z}_q^n$ where

$$\widetilde{\mathbf{m}} = \lfloor \Delta \cdot \mathbf{m} \rceil.$$

Then, the client generates a keystream $\mathbf{z} \in \mathbb{Z}_q^n$ from the cipher E taking the nonce nc as an input. Adding the keystream $\mathbf{z}$ to the scaled message $\widetilde{\mathbf{m}}$ modulo q, the client gets a symmetric ciphertext $\mathbf{c} = [\widetilde{\mathbf{m}} + \mathbf{z}]_q$ and sends it to the server with the nonce nc.

Server-side Offline Computation. In the offline phase, the server evaluates the keystream using a tuple of nonces $(\mathsf{nc}_0, \ldots, \mathsf{nc}_{B-1})$ and the FV-encrypted symmetric key $\mathcal{K}$. As a result, the server gets an FV-ciphertext $\mathcal{V}$ containing the keystreams of E in its slots. Finally, the server performs a linear transformation $\mathsf{SlotToCoeff}^{\mathsf{FV}}$ that moves the data from the slots to the coefficients, obtaining an FV-ciphertext $\mathcal{Z}$ containing the keystreams of E in its coefficients. All of the homomorphic evaluations are done in the FV scheme.

Server-side Online Computation. Given a tuple of symmetric ciphertexts, the server scales up the data into FV-ciphertext space to obtain an FV-ciphertext $\mathcal{C}$ containing the symmetric ciphertexts in its coefficients. Subtracting the homomorphically evaluated keystream $\mathcal{Z}$ from $\mathcal{C}$, the server gets an FV-ciphertext $\mathcal{X}$ containing the scaled messages in its coefficients. The last step is HalfBoot, a modified bootstrapping process used in the RtF transciphering framework. Taking the FV-ciphertext $\mathcal{X}$ as an input, it outputs a CKKS-ciphertext $\mathcal{M}$ containing the real messages in its slots.

C Number of Monomials in Rubato

The round function of Rubato is defined by

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

where the two linear maps MixColumns and MixRows can be represented by $n \times n$-matrices over $\mathbb{Z}_q$. Their product represents MixRows $\circ$ MixColumns as follows.

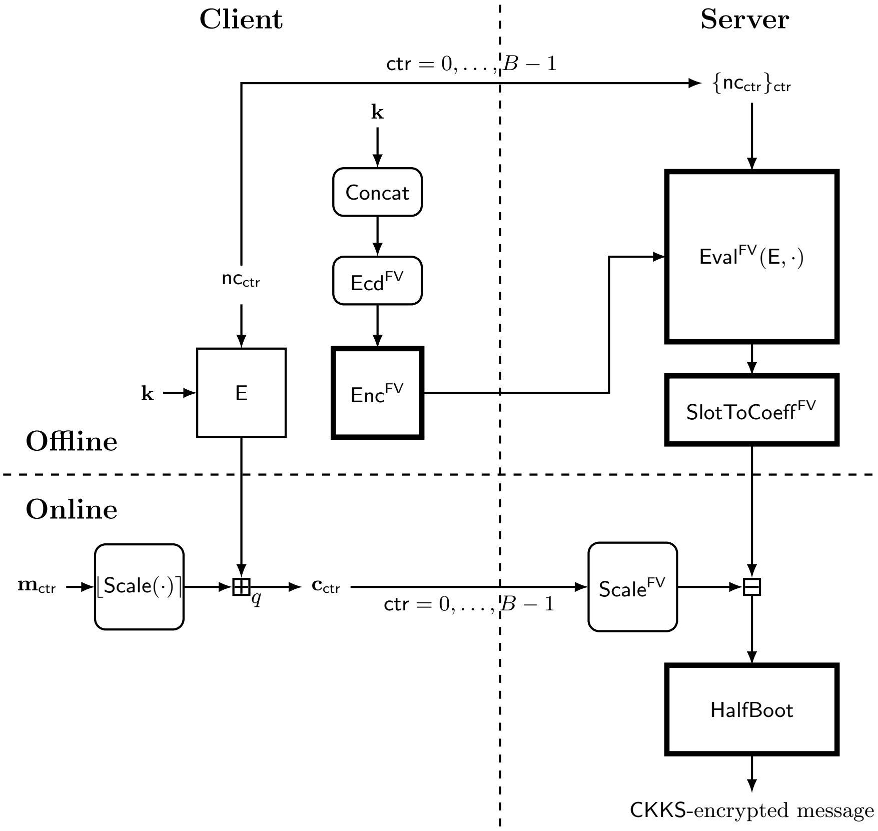


Fig. 5: The RtF transciphering framework. Homomorphic encryption and evaluation is performed in the boxes with thick lines. Operations in the boxes with rounded corners do not use any secret information. The vertical dashed line distinguishes the client-side and the server-side computation, while the horizontal dashed line distinguishes the offline and the online computation. The client sends ciphertexts block by block, while the server gathers B ciphertext blocks and recovers the CKKS-encrypted message of the ciphertexts.

Similar to HERA [25], we check that the matrix representation of $\mathsf{MixRows} \circ \mathsf{MixColumns}$ has no zero entry. It implies that $\mathsf{MixRows} \circ \mathsf{MixColumns}$ contains all the linear monomials in its polynomial representation, and hence RF contains all the quadratic monomials. More precisely, if $a_i \neq 0$ for $i = 1, \ldots, n$, then we have

$$\begin{aligned}(a_1x_1 + a_2x_2 + \cdots + a_nx_n + b)^2 &= \sum_{i,j} a_ia_jx_ix_j + \sum_i 2a_ibx_i + b^2 \\ &= \sum_{i \leq j} \alpha(i,j)a_ia_jx_ix_j + \sum_i 2a_ibx_i + b^2,\end{aligned}$$

where

$$\alpha(i,j) = \begin{cases} 1 & \text{if either } i = j; \\ 2 & \text{if } i < j. \end{cases}$$

Since the plaintext modulus q is prime and $q > 2^{20}$, every quadratic has a nonzero coefficient.

We can estimate the number of monomials in Rubato with more rounds. Let $\mathbf{b} = (b_1, \ldots, b_n)$ be the output of the first round function. The second round function will contain all the quadratic monomials in $\mathbf{b}$. When we view the second round function as a polynomial in $b_1, \ldots, b_n$, some coefficients might become zero, while this happens only with a probability of $1/q$. Heuristically (with the independence assumption), each monomial will remain at the second round with probability $1 - (1/q)^n$. This heuristic is confirmed by our computation, showing all possible monomials at the end of the second round. We conjecture that this property will hold for more than two rounds.

D Branch Number of the Linear Layer in Rubato

Linear branch number B_ℓ and differential branch number B_d for a given a $v \times v$-matrix $\mathbf{M}$ are defined by

$$B_\ell(\mathbf{M}) = \min_{\mathbf{x} \neq 0}\{\mathsf{hw}(\mathbf{x}) + \mathsf{hw}(\mathbf{M}^\intercal\mathbf{x})\},$$

$$B_d(\mathbf{M}) = \min_{\mathbf{x} \neq 0}\{\mathsf{hw}(\mathbf{x}) + \mathsf{hw}(\mathbf{M}\mathbf{x})\}$$

respectively, where hw denotes the word-wise hamming weight function. It is easily seen that $2 \leq B_\ell(\mathbf{M}), B_d(\mathbf{M}) \leq v+1$ for an invertible matrix $\mathbf{M}$. A $v \times v$-matrix $\mathbf{M}$ is a *maximum distance separable* (MDS) matrix if $B_\ell(\mathbf{M}) = B_d(\mathbf{M}) = v+1$. It is also known that $B_\ell(\mathbf{M}) = v+1$ if and only if $B_d(\mathbf{M}) = v+1$ [45].

The branch number of the linear layer of Rubato is 8 when $v = 4$ [25]. In this section, we give the branch number of the linear layer of Rubato for general v.

Theorem 1. *If $\mathbf{M}_v$ is $v \times v$-MDS matrix, then the linear and the differential branch numbers of*

$$\mathsf{MixRows} \circ \mathsf{MixColumns}$$

are all $2v$ where $\mathsf{MixColumns}$ (resp. $\mathsf{MixRows}$) multiplies the matrix $\mathbf{M}_v$ to each column (resp. row) of the state.

Proof. We will prove that the differential branch number of $\mathsf{MixRows}\circ\mathsf{MixColumns}$ is $2v$. The linear branch number is computed similarly. We use the notations in Figure 6. We define $r_i^{(x)}, c_i^{(x)}, r_i^{(y)}, c_i^{(y)}, r_i^{(z)}, c_i^{(z)}$ as follows:

$$\begin{aligned}
r_i^{(x)} &= \mathsf{hw}((x_{i,1}, x_{i,2}, \ldots, x_{i,v}));\\
c_i^{(x)} &= \mathsf{hw}((x_{1,i}, x_{2,i}, \ldots, x_{v,i}));\\
r_i^{(y)} &= \mathsf{hw}((y_{i,1}, y_{i,2}, \ldots, y_{i,v}));\\
c_i^{(y)} &= \mathsf{hw}((y_{1,i}, y_{2,i}, \ldots, y_{v,i}));\\
r_i^{(z)} &= \mathsf{hw}((z_{i,1}, z_{i,2}, \ldots, z_{i,v}));\\
c_i^{(z)} &= \mathsf{hw}((z_{1,i}, z_{2,i}, \ldots, z_{v,i})).
\end{aligned}$$

Let a be the number of non-zero columns in state X. By reordering the indices, one can get

$$\begin{aligned}
&c_1^{(x)} \geq c_2^{(x)} \geq \ldots c_a^{(x)} > 0,\\
&c_{a+1}^{(x)} = c_{a+2}^{(x)} = \cdots = c_v^{(x)} = 0,
\end{aligned}$$

which implies that $\mathsf{hw}(X) = \sum_{k=1}^{v} c_k^{(x)} \geq a \cdot c_a^{(x)}$.

Let b be the number of non-zero row in Y. Since the branch number of $\mathsf{MixColumns}$ is $v+1$, $c_a^{(y)} \geq v+1-c_a^{(x)}$ which implies $b \geq v+1-c_a^{(x)}$. Since the number of non-zero columns are a, $r_i^{(y)} \leq a$. By reordering the indices, one can get

$$\begin{aligned}
&a \geq r_1^{(y)} \geq r_2^{(y)} \cdots \geq r_b^{(y)} > 0,\\
&r_{b+1}^{(y)} = r_{b+2}^{(y)} = \cdots = r_v^{(y)} = 0.
\end{aligned}$$

Then, we have

$$\begin{aligned}
\mathsf{hw}(Z) = \sum_{k=1}^{v} r_k^{(z)} &= \sum_{k=1}^{b} r_k^{(z)} + \sum_{k=b+1}^{v} r_k^{(z)}\\
&\geq \sum_{k=1}^{b} (v+1-r_k^{(y)}) \geq b \cdot (v+1-a)\\
&\geq \left(v+1-c_a^{(x)}\right) \cdot (v+1-a).
\end{aligned}$$

Combining the above inequalities, we have

$$\mathsf{hw}(X) + \mathsf{hw}(Z) \geq a \cdot c_a^{(x)} + (v+1-c_a^{(x)}) \cdot (v+1-a).$$

The right hand side is a linear expression with respect to each variable when the other one is fixed, so that it is enough to check

$$\left(a, c_a^{(x)}\right) \in \{(1,1), (1,v), (v,1), (v,v)\}$$

in order to find the minimum value. By substituting those values, we have

$$\mathsf{hw}(X) + \mathsf{hw}(Z) \geq \min(v^2 + 1, 2v) = 2v.$$

Finally, the following example in Figure 7 shows that the branch number of $\mathsf{MixColumns} \circ \mathsf{MixRows}$ is $2v$. □

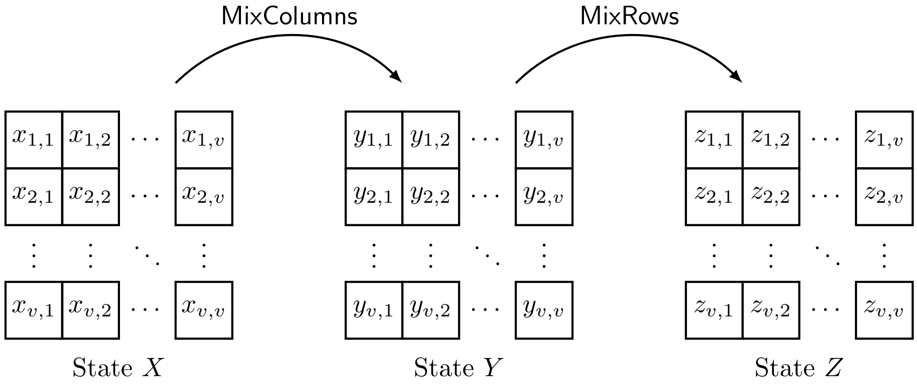

MixColumns
MixRows
$x_{1,1}$ $x_{1,2}$ $\cdots$ $x_{1,v}$
$x_{2,1}$ $x_{2,2}$ $\cdots$ $x_{2,v}$
$x_{v,1}$ $x_{v,2}$ $\cdots$ $x_{v,v}$
$y_{1,1}$ $y_{1,2}$ $\cdots$ $y_{1,v}$
$y_{2,1}$ $y_{2,2}$ $\cdots$ $y_{2,v}$
$y_{v,1}$ $y_{v,2}$ $\cdots$ $y_{v,v}$
$z_{1,1}$ $z_{1,2}$ $\cdots$ $z_{1,v}$
$z_{2,1}$ $z_{2,2}$ $\cdots$ $z_{2,v}$
$z_{v,1}$ $z_{v,2}$ $\cdots$ $z_{v,v}$
State X
State Y
State Z

Fig. 6: Diagram of state change in Rubato.

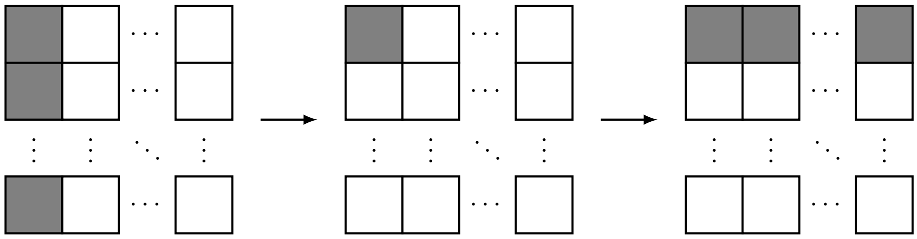


Fig. 7: Pictorial representation of an example satisfying $B_d = 2v$.

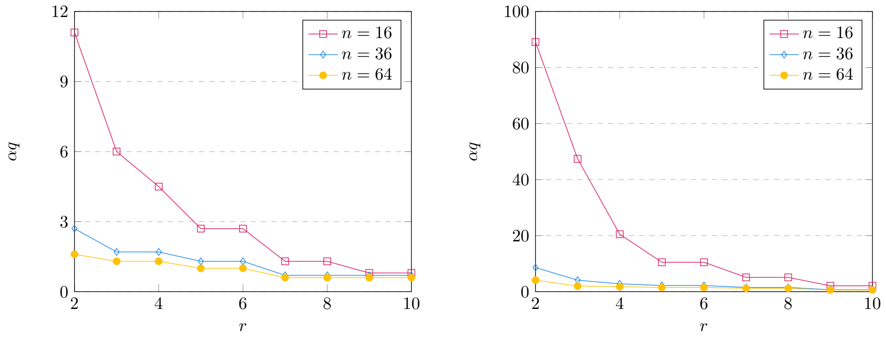
E Additional Plots on Security Analysis

In this section, we give some additional plots on the security analysis of Rubato. In Figure 8, we plot a trade-off relation between the number of rounds r and the width of the discrete Gaussian distribution αq. One can use a customized set of parameters for various applications.

In the Arora-Ge attack, the probability such that a noise sampled from a discrete Gaussian is denoted by

$$\varepsilon_t = \Pr\left[e \leftarrow D_{\alpha q} : |e| \leq t\alpha q\right].$$

The value $t\alpha q$ determines the success probability of the attacks and necessarily affects the total complexity of the attacks. We give a plot of the complexity of Arora-Ge attack, which is dominating attacks, according to $t\alpha q$ in Figure 9.



(a) 80-bit security

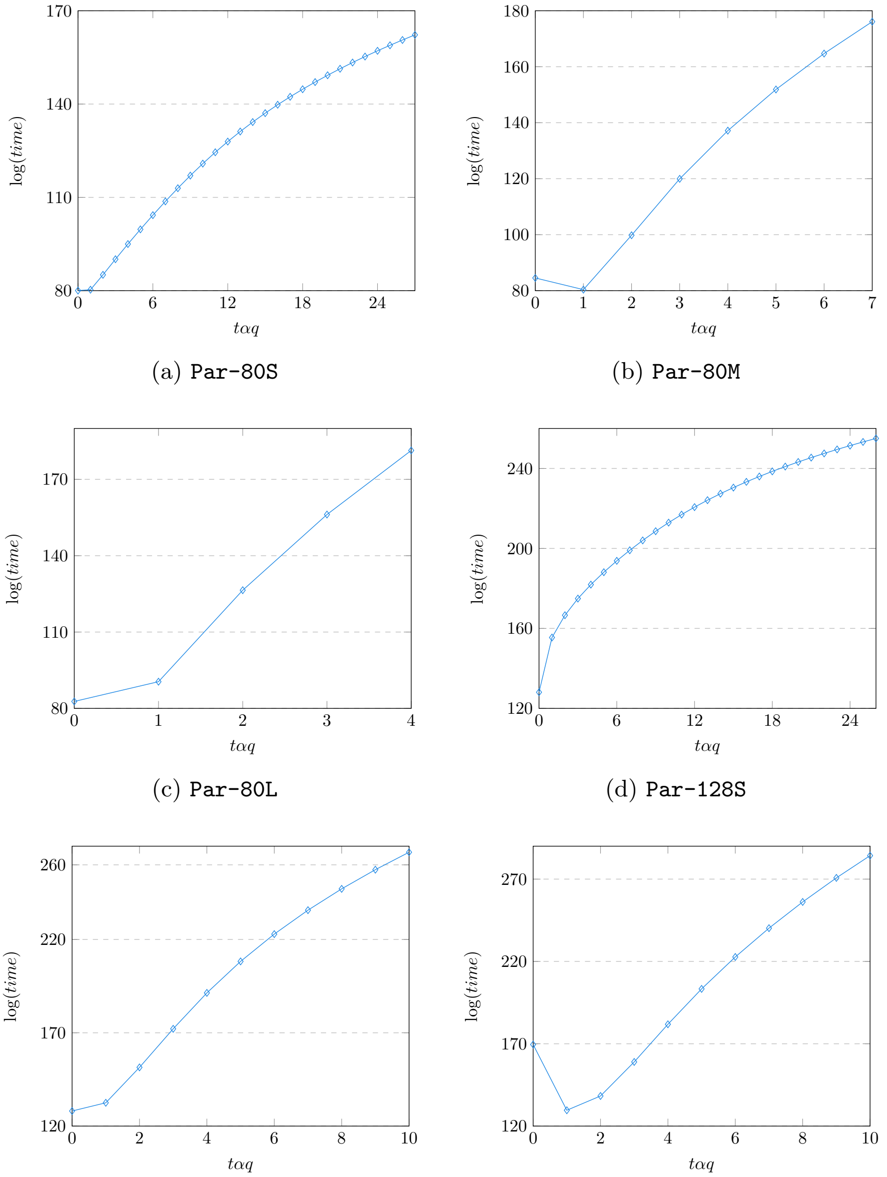
(b) 128-bit security

Fig. 8: The appropriate choice of the number of rounds r and the width of discrete Gaussian distribution αq for a given security level.

F HE Parameters of the Implementation

In Section 5, we give the performance evaluation of Rubato along with the RtF framework. We basically follow the Par-128a HE parameters in [25] in our implementation. Specifically,

- the hamming weight h of the secret key is 192;
- the range K of sign evaluation is 25;
- the number R of double angle formula is 2;
- the degree of sign evaluation is 63;
- the degree of inverse sine evaluation is 7;
- the degree N of the polynomial modulus is 2^{16};
- the number of slots in the FV scheme in the RtF framework is 2^{16};
- the scale of the final CKKS-ciphertext is 2^{45};
- the level of the final CKKS-ciphertext is 7;
- the bit-length of the largest ciphertext modulus including special primes is 1533;
- the ratio of the first ciphertext modulus to the bootstrapping scaling factor is 16.



(a) `Par-80S`

(b) `Par-80M`

(c) `Par-80L`

(d) `Par-128S`

(e) `Par-128M`

(f) `Par-128L`

Fig. 9: The log of time complexity of the Arora-Ge attack as a function in $t\alpha q$.