Cryptanalysis of Symmetric Primitives over Rings and a Key Recovery Attack on Rubato [⋆]

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

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

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

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

1 Introduction

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

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

⋆ Author list in alphabetical order.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The story is slightly different for FHE, as these schemes have most often been defined over polynomial rings, but also here the associated primitives have typically been defined over fields. There are, however, two recently proposed symmetric schemes defined over rings: Elisabeth [27] and Rubato [49]. Furthermore, these schemes are not used to construct an FHE scheme, but rather combined with already existing FHE schemes as part of a larger framework.

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

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

1.2 Our Contributions

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

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

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

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

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

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

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

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

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

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

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

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

2.1 Notation and Preliminaries

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

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

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

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

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

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

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

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

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

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

2.2 Solving Polynomial Systems Modulo q

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

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

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

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

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

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

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

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

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

2.3 Impact on Security

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

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

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

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

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

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

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

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

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

3.1 Linearization Attacks

Linearization is a well-known class of techniques used to solve multivariate polynomial systems of equations over finite fields (see, e.g., [54]). The core idea is to turn a system of non-linear equations into a linear system by treating each monomial as a separate variable. In general, the method generates polynomials of some degree, up to the point where the number of equations exceeds the number of monomials so a solution can be found by linear algebra.

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

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

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

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

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

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

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

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

3.2 Gröbner Basis Attack

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

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

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

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

3.3 Interpolation Attack

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

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

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

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

3.4 Higher-Order Differential Attack

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

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

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

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

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

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

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

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

□

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

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

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

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

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

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

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

4.2 Learning from Elisabeth: Look-up Tables

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

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

4.3 "Cut and Sew" Approach

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

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

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

Let us consider the two cases in more detail.

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

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

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

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

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

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

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

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

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

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

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

$$x = x_2 \cdot p_2 + x_1 \mapsto S(x) = S_2(x_2) \cdot p_2 + S_1(x_1)$$
$$x = x_1' \cdot p_1 + x_2' \mapsto S'(x) = S_1'(x_1') \cdot p_1 + S_2'(x_2')$$

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

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

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

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

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

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

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

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

5 Rubato

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

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

5.1 Description of Rubato

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

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

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

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

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

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

Components. We introduce the following components of Rubato:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

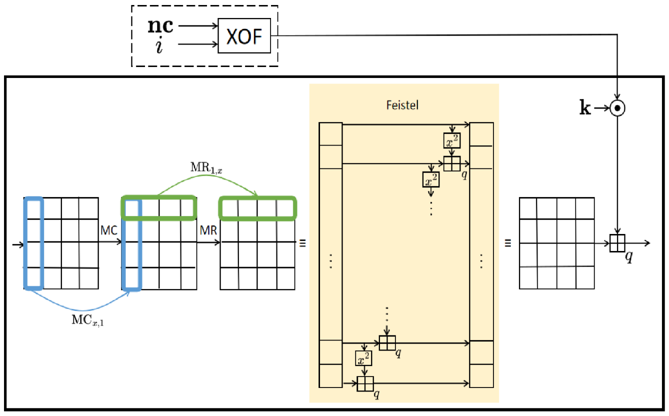


Fig. 1: The round function of Rubato.

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

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

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

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

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

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

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

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

The choice of the parameter q greatly impacts the security of Rubato, and so to better understand the different aspects of this choice, we recall the RtF (Real-to-Finite field) transciphering framework, which is the greater context the Rubato ciphers are intended for. We stress, in particular, that there is no requirement for q to be prime from an applicability perspective of this framework.

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

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

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

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

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

Definition 1 (MDS [29]). *The* branch number *of* $M \in \mathbb{Z}_q^{n\times n}$ *is defined as* $\mathcal{B}(M) = \min_{x\in\mathbb{Z}_q^n\setminus\{0\}}\{\mathrm{hw}(x) + \mathrm{hw}(M(x))\}$, *where* $\mathrm{hw}(\cdot)$ *is the bundle weight in wide trail terminology. A matrix* $M \in \mathbb{Z}_q^{n\times n}$ *is called a Maximum Distance Separable (MDS) matrix if and only if* $\mathcal{B}(M) = n + 1$.

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

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

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

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

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

6 Key Recovery Attack on Rubato

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

1. First, we recover the correct key and noise modulo m, when m is a factor of q lying in a particular interval.

2. Then, we recover the positions in the key stream where the noise added by AGN($\cdot$) is exactly 0.
3. Finally, we recover the secret key by setting up a system of polynomial equations using the knowledge of positions with no noise, and solving the system by re-linearization.

For ease of exposition, we specify some further notation:

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

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

6.1 Recovering Key and Noise Modulo a Small Factor of q

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

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

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

We then have the equations

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

6.3 Key-Recovery of the full Rubato Key

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

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

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

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

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

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

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

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

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

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

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

7 Assumptions and Cost of the Attack on Rubato

7.1 Assumptions on q

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

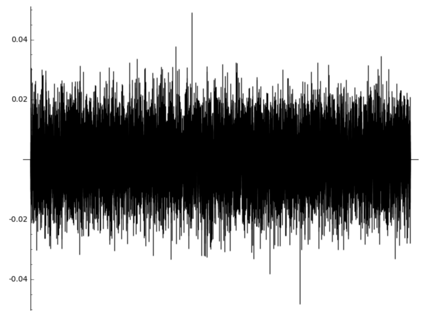
Score Value for $\tilde{\mathbf{k}}$. For a given key guess $\tilde{\mathbf{k}} = (\tilde{k}_1, \dots, \tilde{k}_n)$ modulo m, we now define a score value for $\tilde{\mathbf{k}}$. First, execute $\mathsf{Ru}_m[\tilde{\mathbf{k}}]$ producing the stream $\tilde{w}_1, \dots, \tilde{w}_s$. From the known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

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

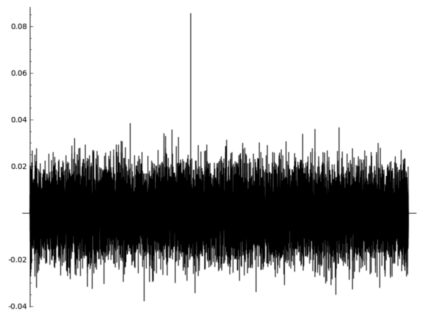
The score function gives a measure of how much the candidate noise value produced by $\tilde{\mathbf{k}}$ deviates from the uniform distribution in the same way as values drawn from $D_{\alpha q}$ modulo m will deviate from uniform. When $\tilde{\mathbf{k}}$ is the correct guess modulo m, we expect $\mathrm{Sc}(\tilde{\mathbf{k}})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

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

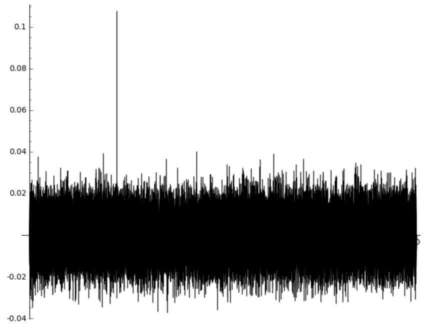
Set of Susceptible Values. We list the bounds $m_{\min}$ and $m_{\max}$ and the threshold t that allow attacks with complexity lower than 2^{λ} for each parameter set defined for Rubato in Table 4. We performed an exhaustive search on 26-bit numbers (for Rubato-80S and Rubato-128S) and 25-bit numbers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



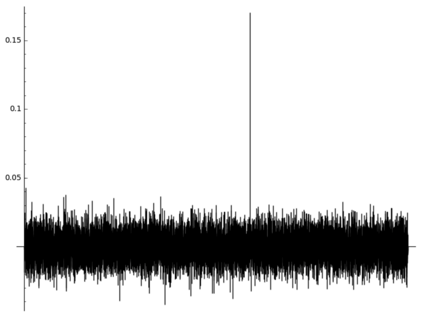
(a) **Rubato-80S**: distinguishing correct key guess modulo 11 using 14641 key samples.



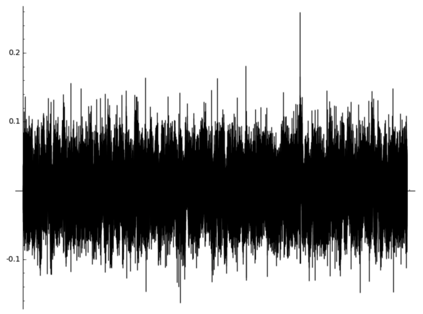
(b) **Rubato-128S**: distinguishing correct key guess modulo 11 using 14641 key samples.



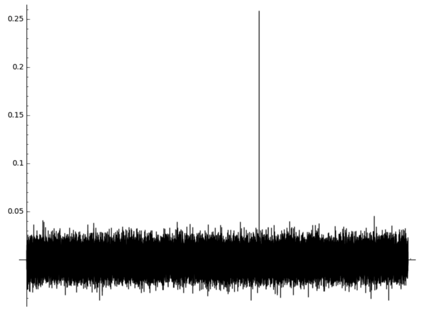
(c) **Rubato-80M**: distinguishing correct key guess modulo 3 using 59049 key samples.



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



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



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

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

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

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

7.2 Practical Verification of the Attack

We have verified the attack described in Section 6 experimentally[5]. We also report on the experiments determining the smallest m for which we can distinguish a correct key guess modulo m from the wrong ones.

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

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

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

7.3 Attack Complexities

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

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

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

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

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

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

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

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

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

8 Final Remarks

8.1 Restoring the Security of Rubato

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

Restricting q to Prime Numbers. The easiest way to prevent our attack is to simply restrict q to be prime. Since our attack is based on the assumption that q contains small factors, this restriction immediately gives Rubato instances that are immune to any small-factor attack. As already mentioned in Section 5.2, it is most common to choose the plaintext modulus of FV, the other part q plays in the RtF framework, to be prime. However, we stress that this choice is made for efficiency [24] *not* security in the FV scheme [1,35]. As our attack has demonstrated, restricting q to be prime is a choice made for security in Rubato, not convenience.

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

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

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

8.2 Conclusion

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

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

References

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

10. C. Baum, L. Braun, A. Munch-Hansen, and P. Scholl. Moz$\mathbb{Z}_{2^k}$arella: Efficient Vector-OLE and Zero-Knowledge Proofs over $\mathbb{Z}_{2^k}$. In *Advances in Cryptology–CRYPTO 2022: 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15–18, 2022, Proceedings, Part IV*, pages 329–358. Springer, 2022.
11. R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers. The SIMON and SPECK lightweight block ciphers. In *Proceedings of the 52nd Annual Design Automation Conference, 2015*, pages 175:1–175:6. ACM, 2015.
12. C. Beierle, A. Biryukov, L. C. dos Santos, J. Großschädl, L. Perrin, A. Udovenko, V. Velichkov, and Q. Wang. Lightweight AEAD and hashing using the SPARKLE permutation family. Submission to the NIST lightweight cryptographic standardization process (Finalist).
13. D. J. Bernstein. The Salsa20 Family of Stream Ciphers. In *New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 of *LNCS*, pages 84–97. Springer, 2008.
14. T. Beyne, A. Canteaut, I. Dinur, M. Eichlseder, G. Leander, G. Leurent, M. Naya-Plasencia, L. Perrin, Y. Sasaki, Y. Todo, and F. Wiemer. Out of Oddity - New Cryptanalytic Techniques Against Symmetric Primitives Optimized for Integrity Proof Systems. In *Advances in Cryptology - CRYPTO 2020*, volume 12172 of *LNCS*, pages 299–328. Springer, 2020.
15. E. Biham, A. Biryukov, and A. Shamir. Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. In *Advances in Cryptology - EUROCRYPT 1999*, volume 1592 of *LNCS*, pages 12–23. Springer, 1999.
16. E. Biham, O. Dunkelman, and N. Keller. The Rectangle Attack - Rectangling the Serpent. In *Advances in Cryptology - EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 340–357. Springer, 2001.
17. E. Biham and A. Shamir. Differential Cryptanalysis of DES-like Cryptosystems. *J. Cryptol.*, 4(1):3–72, 1991.
18. E. Biham and A. Shamir. *Differential Cryptanalysis of the Data Encryption Standard*. Springer, 1993.
19. A. Bogdanov, D. Khovratovich, and C. Rechberger. Biclique Cryptanalysis of the Full AES. In *Advances in Cryptology - ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 344–371. Springer, 2011.
20. N. Bordes, J. Daemen, D. Kuijsters, and G. V. Assche. Thinking Outside the Superbox. In *Advances in Cryptology - CRYPTO 2021*, volume 12827 of *LNCS*, pages 337–367. Springer, 2021.
21. W. Bosma, J. J. Cannon, C. Fieker, and A. Steel (eds.). Gröbner Bases over Euclidean Rings. In *Magma Handbook v.2.27*. Computational Algebra Group, School of Mathematics and Statistics, University of Sydney. https://magma.maths.usyd.edu.au/magma/handbook/text/1259#14396.
22. C. Bouvier, P. Briaud, P. Chaidos, L. Perrin, R. Salen, V. Velichkov, and D. Willems. New Design Techniques for Efficient Arithmetization-Oriented Hash Functions:Anemoi Permutations and Jive Compression Mode. Cryptology ePrint Archive, Paper 2022/840, 2022. https://eprint.iacr.org/2022/840.
23. A. Caminata and E. Gorla. Solving multivariate polynomial systems and an invariant from commutative algebra. In *Arithmetic of Finite Fields: 8th International Workshop, WAIFI 2020, Rennes, France, July 6–8, 2020, Revised Selected and Invited Papers 8*, pages 3–36. Springer, 2021.
24. H. Chen, K. Laine, and R. Player. Simple encrypted arithmetic library - seal v2.1. Cryptology ePrint Archive, Paper 2017/224, 2017. https://eprint.iacr.org/2017/224.

25. J. Cho, J. Ha, S. Kim, B. Lee, J. Lee, J. Lee, D. Moon, and H. Yoon. Transciphering framework for approximate homomorphic encryption. In M. Tibouchi and H. Wang, editors, *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part III*, pages 640–669. Springer, 2021.
26. C. Cid, L. Grassi, A. Gunsing, R. Lüftenegger, C. Rechberger, and M. Schofnegger. Influence of the Linear Layer on the Algebraic Degree in SP-Networks. *IACR Trans. Symmetric Cryptol.*, 2022(1):110–137, 2022.
27. O. Cosseron, C. Hoffmann, P. Méaux, and F.-X. Standaert. Towards Globally Optimized Hybrid Homomorphic Encryption - Featuring the Elisabeth Stream Cipher. In *Advances in Cryptology–ASIACRYPT 2022: 28th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, December 5–9, 2022, Proceedings, Part III*, pages 487–516. Springer, 2023.
28. R. Cramer, I. Damgård, D. Escudero, P. Scholl, and C. Xing. SPD$\mathbb{Z}_{2^k}$: Efficient MPC mod 2^k for Dishonest Majority. In *Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part II*, pages 769–798. Springer, 2018.
29. J. Daemen and V. Rijmen. The Wide Trail Design Strategy. In *Cryptography and Coding - IMACC 2001*, volume 2260 of *LNCS*, pages 222–238. Springer, 2001.
30. A. P. Dalskov, D. Escudero, and M. Keller. Fantastic Four: Honest-Majority Four-Party Secure Computation With Malicious Security. In *USENIX Security Symposium*, pages 2183–2200, 2021.
31. J. D. Dixon. Exact Solution of Linear Equations Using P-Adic Expansions. *Numerische Mathematik*, 40(1):137–141, 1982.
32. C. Dobraunig, M. Eichlseder, L. Grassi, V. Lallemand, G. Leander, E. List, F. Mendel, and C. Rechberger. Rasta: A Cipher with Low ANDdepth and Few ANDs per Bit. In *Advances in Cryptology - CRYPTO 2018*, volume 10991 of *LNCS*, pages 662–692. Springer, 2018.
33. C. Dobraunig, L. Grassi, A. Guinet, and D. Kuijsters. Ciminion: Symmetric encryption based on toffoli-gates over large finite fields. In *Advances in Cryptology - EUROCRYPT 2021*, volume 12697 of *LNCS*, pages 3–34. Springer, 2021.
34. M. Eichlseder, L. Grassi, R. Lüftenegger, M. Øygarden, C. Rechberger, M. Schofnegger, and Q. Wang. An Algebraic Attack on Ciphers with Low-Degree Round Functions: Application to Full MiMC. In *Advances in Cryptology - ASIACRYPT 2020*, volume 12491 of *LNCS*, pages 477–506. Springer, 2020.
35. J. Fan and F. Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Paper 2012/144, 2012. https://eprint.iacr.org/2012/144.
36. J.-C. Faugère. A new efficient algorithm for computing Gröbner bases (F_4). *Journal of pure and applied algebra*, 139(1-3):61–88, 1999.
37. J.-C. Faugère, P. Gianni, D. Lazard, and T. Mora. Efficient Computation of Zero-Dimensional Gröbner Bases by Change of Ordering. *Journal of Symbolic Computation*, 16(4):329–344, 1993.
38. C. Ganesh, A. Nitulescu, and E. Soria-Vazquez. Rinocchio: SNARKs for Ring Arithmetic. Cryptology ePrint Archive, Paper 2021/322, 2021. https://eprint.iacr.org/2021/322.
39. R. Geelen, I. Iliashenko, J. Kang, and F. Vercauteren. On Polynomial Functions Modulo p^e and Faster Bootstrapping for Homomorphic Encryption. Cryptology ePrint Archive, Paper 2022/1364, 2022. https://eprint.iacr.org/2022/1364.

40. P. Gopalan. Query-Efficient Algorithms for Polynomial Interpolation over Composites. *SIAM Journal on Computing*, 38(3):1033–1057, 2008.
41. L. Grassi. Bounded Surjective Quadratic Functions over $\mathbb{F}_p^n$ for MPC-/ZK-/HE-Friendly Symmetric Primitives. Cryptology ePrint Archive, Paper 2022/1313, 2022. https://eprint.iacr.org/2022/1313.
42. L. Grassi, Y. Hao, C. Rechberger, M. Schofnegger, R. Walch, and Q. Wang. Horst Meets Fluid-SPN: Griffin for Zero-Knowledge Applications. Cryptology ePrint Archive, Paper 2022/403, 2022. https://eprint.iacr.org/2022/403.
43. L. Grassi, D. Khovratovich, R. Lüftenegger, C. Rechberger, M. Schofnegger, and R. Walch. Reinforced Concrete: A Fast Hash Function for Verifiable Computation. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 202*, pages 1323–1335. ACM, 2022.
44. L. Grassi, D. Khovratovich, C. Rechberger, A. Roy, and M. Schofnegger. Poseidon: A New Hash Function for Zero-Knowledge Proof Systems. In *30th USENIX Security Symposium, USENIX Security 2021*, pages 519–535. USENIX Association, 2021.
45. L. Grassi, D. Khovratovich, S. Rønjom, and M. Schofnegger. The Legendre Symbol and the Modulo-2 Operator in Symmetric Schemes over $\mathbb{F}_p^n$ Preimage Attack on Full Grendel. *IACR Trans. Symmetric Cryptol.*, 2022(1):5–37, 2022.
46. L. Grassi, R. Lüftenegger, C. Rechberger, D. Rotaru, and M. Schofnegger. On a Generalization of Substitution-Permutation Networks: The HADES Design Strategy. In *Advances in Cryptology - EUROCRYPT 2020*, volume 12106 of *LNCS*, pages 674–704. Springer, 2020.
47. L. Grassi, S. Onofri, M. Pedicini, and L. Sozzi. Invertible Quadratic Non-Linear Layers for MPC-/FHE-/ZK-Friendly Schemes over $\mathbb{F}_p^n$ Application to Poseidon. *IACR Trans. Symmetric Cryptol.*, 2022(3):20–72, 2022.
48. L. Grassi, M. Øygarden, M. Schofnegger, and R. Walch. From Farfalle to Megafono via Ciminion: The PRF Hydra for MPC Applications. Cryptology ePrint Archive, Paper 2022/342, 2022. https://eprint.iacr.org/2022/342.
49. J. Ha, S. Kim, B. Lee, J. Lee, and M. Son. Rubato: Noisy Ciphers for Approximate Homomorphic Encryption. In *Advances in Cryptology - EUROCRYPT 2022*, volume 13275 of *LNCS*, pages 581–610. Springer, 2022.
50. T. Jakobsen and L. R. Knudsen. The Interpolation Attack on Block Ciphers. In *Fast Software Encryption – FSE 1997*, volume 1267 of *LNCS*, pages 28–40. Springer, 1997.
51. N. Keller and A. Rosemarin. Mind the Middle Layer: The HADES Design Strategy Revisited. In *Advances in Cryptology - EUROCRYPT 2021*, volume 12697 of *LNCS*, pages 35–63. Springer, 2021.
52. A. J. Kempner. Polynomials and their residue systems. *Transactions of the American Mathematical Society*, 22(2):240–266, 1921.
53. A. Kesarwani, S. K. Pandey, S. Sarkar, and A. Venkateswarlu. Recursive MDS matrices over finite commutative rings. *Discrete Applied Mathematics*, 304:384–396, 2021.
54. A. Kipnis and A. Shamir. Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization. In *Advances in Cryptology - CRYPTO 1999*, volume 1666 of *LNCS*, pages 19–30. Springer, 1999.
55. L. R. Knudsen. Truncated and Higher Order Differentials. In *Fast Software Encryption - FSE 1994*, volume 1008 of *LNCS*, pages 196–211. Springer, 1994.
56. L. R. Knudsen and D. A. Wagner. Integral Cryptanalysis. In *Fast Software Encryption – FSE 2002*, volume 2365 of *LNCS*, pages 112–127. Springer, 2002.

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

SUPPLEMENTARY MATERIAL

A Notations

Table 6: Notations

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

B Proof of Lemma 1

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

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

C Comparison of Gröbner Basis Computation

This appendix presents a qualitative comparison of the running time and memory usage of Gröbner basis computations over $\mathbb{Z}_{p^y}$ and $\mathbb{F}_{p^y}$. In each experiment we randomly generate m quadratic polynomials in n variables, i.e., the coefficients of all monomials of degree ≤ 2 are sampled using the in-built random function over $\mathbb{Z}_{p^y}$ or $\mathbb{F}_{p^y}$. A random n-tuple $\mathbf{a}$ is picked as a common solution to the polynomials $f_1(\mathbf{x}), \dots, f_m(\mathbf{x})$ by considering the system $f_i(\mathbf{x}) - f_i(\mathbf{a}) = 0$, $1 \leq i \leq m$. All systems are overdetermined with $\mathbf{a}$ as a unique solution, and the resulting Gröbner basis is always $\{x_1 - a_1, \dots, x_n - a_n\}$. The time and memory usage given in Table 7 are from running the built-in F_4 routine with the computer algebra system Magma V2.27-1 [21].

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

□

Based on this proposition, the following result follows immediately:

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

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

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

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

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

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

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

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

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

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

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

F Details of the "Cut and Sew" Design Strategy

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

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

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

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

Lemma 3. *Let p_1 and p_2 be two distinct primes such that $p_1 > p_2$, and let $q = p_1 \cdot p_2$. Let S_2 be the identity function over $\mathbb{F}_{p_1}$, and let S_1 be an orthomorphism over $\mathbb{F}_{p_2}$, that is, both $x \mapsto S_1(x)$ and $x \mapsto S_1(x) - x$ are invertible. Then* *the function S over $\mathbb{Z}_q$ defined as $S(x \equiv x_2 \cdot p_2 + x_1) = x_2 \cdot p_2 + S_1(x_1)$ for $x_1 \in \{0, 1, \ldots, p_2 - 1\}$ and $x_2 \in \{0, 1, \ldots, p_1 - 1\}$ does not admit a polynomial representation.*

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

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

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

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

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

In particular, we point out that

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

G Equivalent Representation of Rubato

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

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

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

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

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

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

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

H Invertible and MDS Linear Layers for Rubato

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

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

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

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

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

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

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

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

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

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

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

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

I Pseudo-Code of **Rubato** Attack

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

Algorithm 1 Key-Recovery Attack on Rubato

```
Require: Key stream $\mathbf{z} \in \mathbb{Z}_q^s$ generated by Rubato, where
  $m_{\min} \le m \le m_{\max}$,
  $f = \prod_{i=1}^{b} f_i$,
  $f_i \le m$ for $i = 1, \ldots, b$,
  $mf | q$, and
  $mf \ge t$.
Ensure: The secret key $\mathbf{k}$ used to generate $\mathbf{z}$.
  maxSc $\leftarrow 0$
  for each $\tilde{\mathbf{k}} \in \mathbb{Z}_m^n$ do
      $\tilde{\mathbf{w}} \leftarrow \mathsf{Ru}_m[\tilde{\mathbf{k}}]$
      if $\mathrm{Sc}(\tilde{\mathbf{k}}) >$ maxSc then
          maxSc $\leftarrow \mathrm{Sc}(\tilde{\mathbf{k}})$
          $\mathbf{k}^{(0)} \leftarrow \tilde{\mathbf{k}}$
          $\tilde{\mathbf{e}} \leftarrow (\tilde{\mathbf{z}} - \tilde{\mathbf{w}}) \mod m$
      end if
  end for
                                                    ▷ $\mathbf{k}^{(0)} = \mathbf{k} \mod m$
  for $i = 1, \ldots, b$ do
      maxSc $\leftarrow 0$
      for $\mathbf{v} \in \mathbb{Z}_{f_i}^n$ do
          $\tilde{\mathbf{k}} \leftarrow \mathbf{k}^{(i-1)} + (m \prod_{a=1}^{i-1} f_a) \cdot \mathbf{v}$
          $\tilde{\mathbf{w}} \leftarrow \mathsf{Ru}_{m \prod_{a=1}^{i} f_a}[\tilde{\mathbf{k}}]$
          if $\mathrm{Sc}(\tilde{\mathbf{k}}) >$ maxSc then
              maxSc $\leftarrow \mathrm{Sc}(\tilde{\mathbf{k}})$
              $\mathbf{k}^{(i)} \leftarrow \tilde{\mathbf{k}}$
              $\tilde{\mathbf{e}} \leftarrow (\tilde{\mathbf{z}} - \tilde{\mathbf{w}}) \mod m \prod_{a=1}^{i} f_a$
          end if
      end for
  end for
      ▷ $mf$ sufficiently large that all $|\tilde{e}_i| < mf$ for $\tilde{\mathbf{e}} = (\tilde{e}_1, \ldots, \tilde{e}_s)$ with probability at
  least 99%
  $\mathcal{I} \leftarrow \{i | \tilde{e}_i = 0\}$
  $j \leftarrow 1$
  for $i \in \mathcal{I}$ do
      $F_i(k_1, \ldots, k_n) \leftarrow$ polynomial expressing $w_i$ generated by $\mathsf{Ru}_q[\mathbf{k}]$
      $A[j] \leftarrow$ coefficient vector of $F_i$
      $u[j] \leftarrow z_i$
      $j \leftarrow j + 1$
  end for
  $\mathbf{x} \leftarrow$ solution of $A\mathbf{x} = \mathbf{u} \mod q$
  $\mathbf{k} \leftarrow$ values in $\mathbf{x}$ for the monomials $k_1, \ldots, k_n$.
```