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

Lorenzo Grassi, Martha Norberg Hovd, **Irati Manterola Ayala**, Morten Øygarden,
Håvard Raddum, and Qingju Wang

RUHR UNIVERSITÄT BOCHUM RUB

Simula UiB

TELECOM Paris

Rubato

- Family of ciphers proposed by Ha et al. at Eurocrypt 2022[1]

[1] 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.

Rubato

- Family of ciphers proposed by Ha et al. at Eurocrypt 2022[1]

- Usecase: transciphering framework for approximate FHE

[1] 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.

Rubato

- Family of ciphers proposed by Ha et al. at Eurocrypt 2022[1]

- Usecase: transciphering framework for approximate FHE

- Idea: introduce noise to a symmetric cipher of a low algebraic degree

[1] 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.

Rubato

- Family of ciphers proposed by Ha et al. at Eurocrypt 2022[1]
- Usecase: transciphering framework for approximate FHE
- Idea: introduce noise to a symmetric cipher of a low algebraic degree
- Similar to HERA[2] ***BUT*** defined over a ring $\mathbb{Z}_q$

[1] 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.

[2] 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*, volume 13092 of LNCS, pages 640–669. Springer, 2021.

Rubato

Notation:

- $q \geq 2$ integer
- $\mathbb{Z}_q := \mathbb{Z} \cap (-q/2, q/2]$
- State of Rubato $= X \in \mathbb{Z}_q^{v \times v}$
- Block size $n = v^2$

Rubato

Notation:

- $q \geq 2$ integer
- $\mathbb{Z}_q := \mathbb{Z} \cap (-q/2, q/2]$
- State of Rubato $= X \in \mathbb{Z}_q^{v \times v}$
- Block size $n = v^2$

$\mathbf{k} \in \mathbb{Z}_q^n$ symmetric key

$\mathbf{nc} \in \{0,1\}^\lambda$ nonce

$i \in \mathbb{Z}_{\geq 0}$ counter

$$\text{Rubato}[\mathbf{k}, \mathbf{nc}, i] : \underset{\substack{\| \\ (1,2,\ldots,n)}}{\mathbf{is}} \longmapsto \mathbf{z} \in \mathbb{Z}_q^\ell \qquad \ell < n$$

Rubato

Notation:

- $q \geq 2$ integer
- $\mathbb{Z}_q := \mathbb{Z} \cap (-q/2, q/2]$
- State of Rubato $= X \in \mathbb{Z}_q^{v \times v}$
- Block size $n = v^2$

$\mathbf{k} \in \mathbb{Z}_q^n$ symmetric key

$\mathbf{nc} \in \{0,1\}^\lambda$ nonce

$i \in \mathbb{Z}_{\geq 0}$ counter

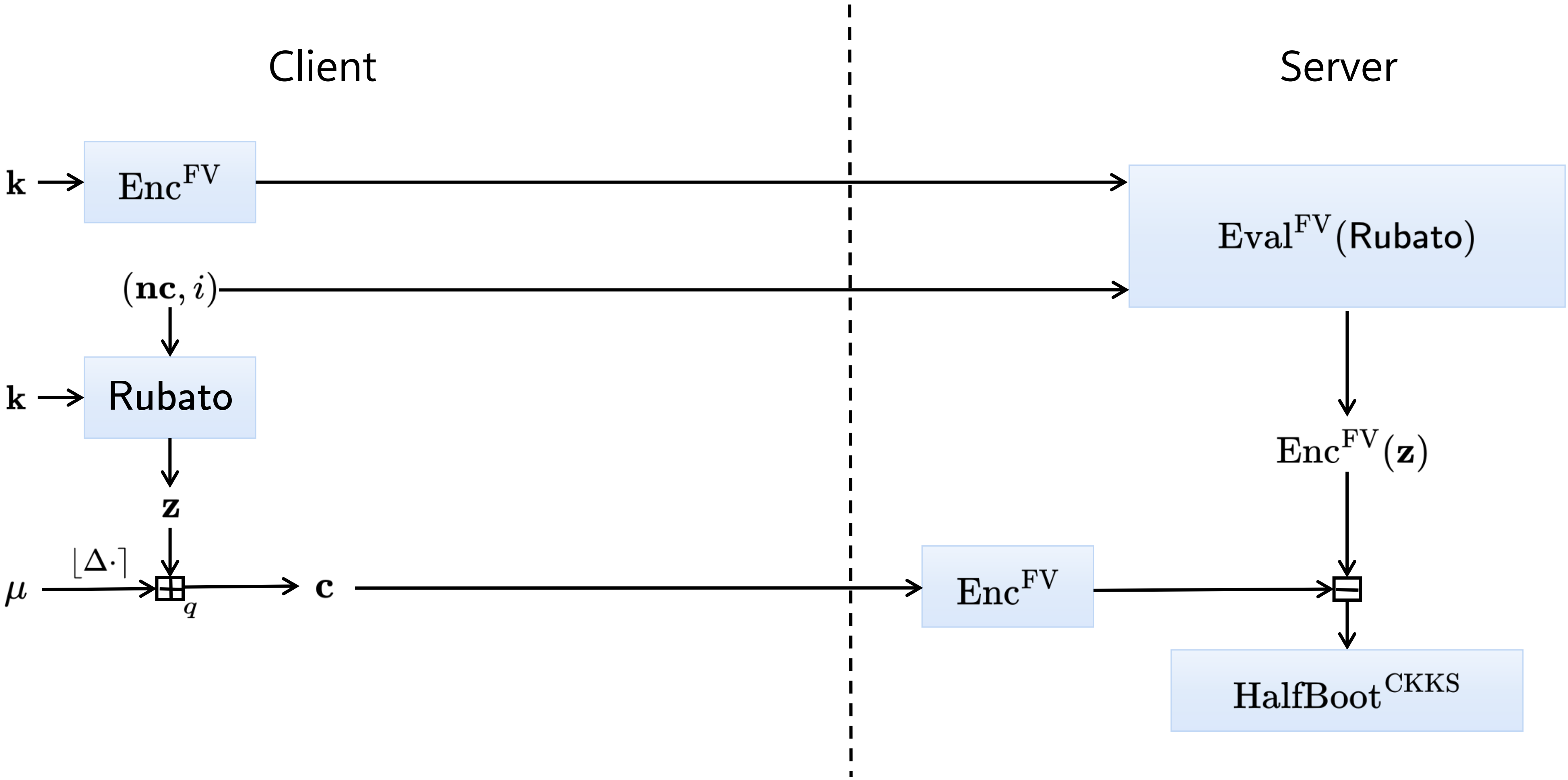
$$\text{Rubato}[\mathbf{k}, \mathbf{nc}, i] : \underset{\substack{\| \\ (1,2,\ldots,n)}}{\mathbf{is}} \longmapsto \mathbf{z} \in \mathbb{Z}_q^\ell \qquad \ell < n$$

Encryption of $\mu \in \mathbb{R}^\ell$

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

Rubato

Rubato in the RtF framework[1]



Client
Server
$\mathbf{k}$
$\mathrm{Enc}^{\mathrm{FV}}$
$\mathrm{Eval}^{\mathrm{FV}}$(Rubato)
$(\mathbf{nc}, i)$
$\mathbf{k}$
Rubato
$\mathrm{Enc}^{\mathrm{FV}}(\mathbf{z})$
$\mathbf{z}$
$\lfloor \Delta \cdot \rceil$
μ
$\boxplus_q$
$\mathbf{c}$
$\mathrm{Enc}^{\mathrm{FV}}$
$\boxminus$
$\mathrm{HalfBoot}^{\mathrm{CKKS}}$

[1] 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*, volume 13092 of LNCS, pages 640–669. Springer, 2021.

Rubato

Components of **Rubato**

Rubato

Components of Rubato

- Add-Round Key (ARK)

$$\boxed{\mathrm{ARK}[\mathbf{k}, i] : \overset{\overset{\mathbb{Z}_q^n}{\cup}}{\mathbf{x}} \mapsto \mathbf{x} + \mathbf{k} \bullet \underset{\underset{\mathrm{XOF} : (\mathbf{nc}, i)}{\uparrow}}{\overset{\overset{(\mathbb{Z}_q^\times)^n}{\cup}}{\mathbf{rc_i}}}}$$

Rubato

Components of Rubato

- Add-Round Key (ARK)

$$\boxed{\mathrm{ARK}[\mathbf{k}, i] : \overset{\overset{\mathbb{Z}_q^n}{\rotatebox{90}{$\in$}}}{\mathbf{x}} \mapsto \mathbf{x} + \mathbf{k} \bullet \underset{\overset{\uparrow}{\mathrm{XOF} : (\mathbf{nc}, i)}}{\overset{\overset{(\mathbb{Z}_q^\times)^n}{\rotatebox{90}{$\in$}}}{\mathbf{rc_i}}}}$$

- MixColumns (MC) and MixRows (MR)

$$\boxed{X \xrightarrow{\mathrm{MC}} M_v \times X \xrightarrow{\mathrm{MR}} (M_v \times X) \times M_v^T}$$

$$M_v = \begin{bmatrix} \mathbf{y_v} \\ \mathbf{y_v} \lll 1 \\ \vdots \\ \mathbf{y_v} \lll v-1 \end{bmatrix} \in \mathbb{Z}_q^{v \times v}$$

$$\begin{aligned} \mathbf{y_4} &= [2, 3, 1, 1]; \\ \mathbf{y_6} &= [4, 2, 4, 3, 1, 1]; \\ \mathbf{y_8} &= [5, 3, 4, 3, 6, 2, 1, 1]; \end{aligned}$$

Rubato

Components of **Rubato**

- Add-Round Key (ARK)

$$\boxed{\mathrm{ARK}[\mathbf{k}, i] : \overset{\overset{\mathbb{Z}_q^n}{\rotatebox{90}{$\in$}}}{\mathbf{x}} \mapsto \mathbf{x} + \mathbf{k} \bullet \overset{\overset{(\mathbb{Z}_q^\times)^n}{\rotatebox{90}{$\in$}}}{\underset{\underset{\mathrm{XOF}:(\mathbf{nc}, i)}{\uparrow}}{\mathbf{rc_i}}}}$$

- MixColumns (MC) and MixRows (MR)

$$\boxed{X \xrightarrow{\mathrm{MC}} M_v \times X \xrightarrow{\mathrm{MR}} (M_v \times X) \times M_v^T}$$

$$M_v = \begin{bmatrix} \mathbf{y_v} \\ \mathbf{y_v} \lll 1 \\ \vdots \\ \mathbf{y_v} \lll v-1 \end{bmatrix} \in \mathbb{Z}_q^{v \times v}$$

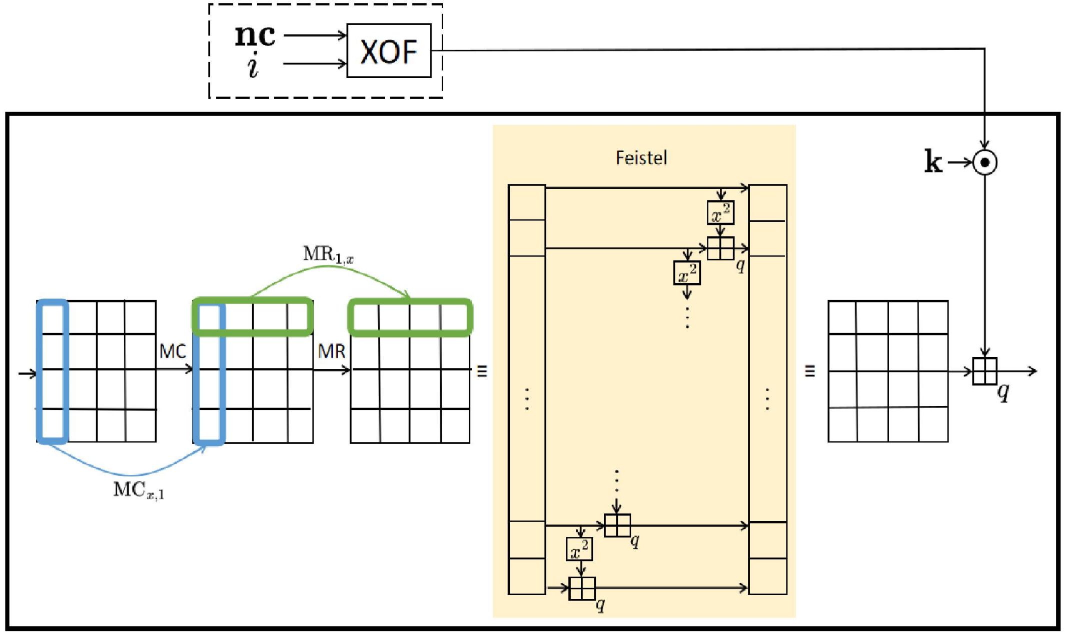
$$\begin{aligned} \mathbf{y_4} &= [2, 3, 1, 1]; \\ \mathbf{y_6} &= [4, 2, 4, 3, 1, 1]; \\ \mathbf{y_8} &= [5, 3, 4, 3, 6, 2, 1, 1]; \end{aligned}$$

- Feistel

$$\boxed{\mathrm{Feistel} : \overset{\overset{\mathbb{Z}_q^n}{\rotatebox{90}{$\in$}}}{\mathbf{x}} = (x_1, \ldots, x_n) \longmapsto (x_1, x_2 + x_1^2, x_3 + x_2^2, \ldots, x_n + x_{n-1}^2)}$$

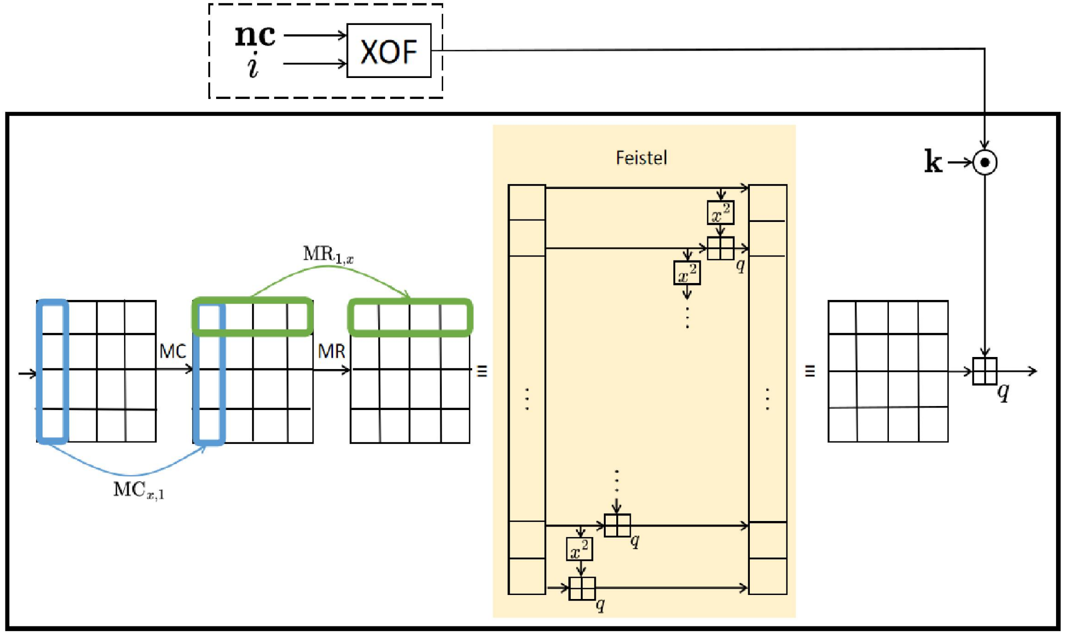
Rubato

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



Rubato

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

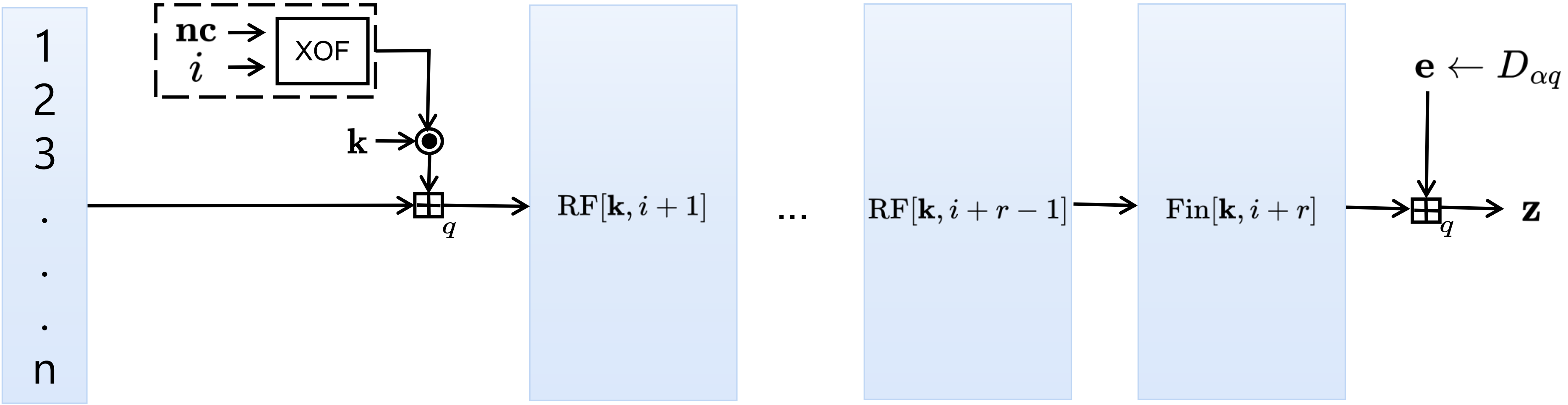


Final round: $\mathrm{Fin}[\mathbf{k}, i+r] = \mathrm{Tr}_{n,\ell} \circ \mathrm{ARK}[\mathbf{k}, i+r] \circ \mathrm{MR} \circ \mathrm{MC} \circ \mathrm{Feistel} \circ \mathrm{MR} \circ \mathrm{MC}$

$$\mathrm{Tr}_{n,\ell}(x_1, \ldots, x_n) = (x_1, \ldots, x_\ell)$$

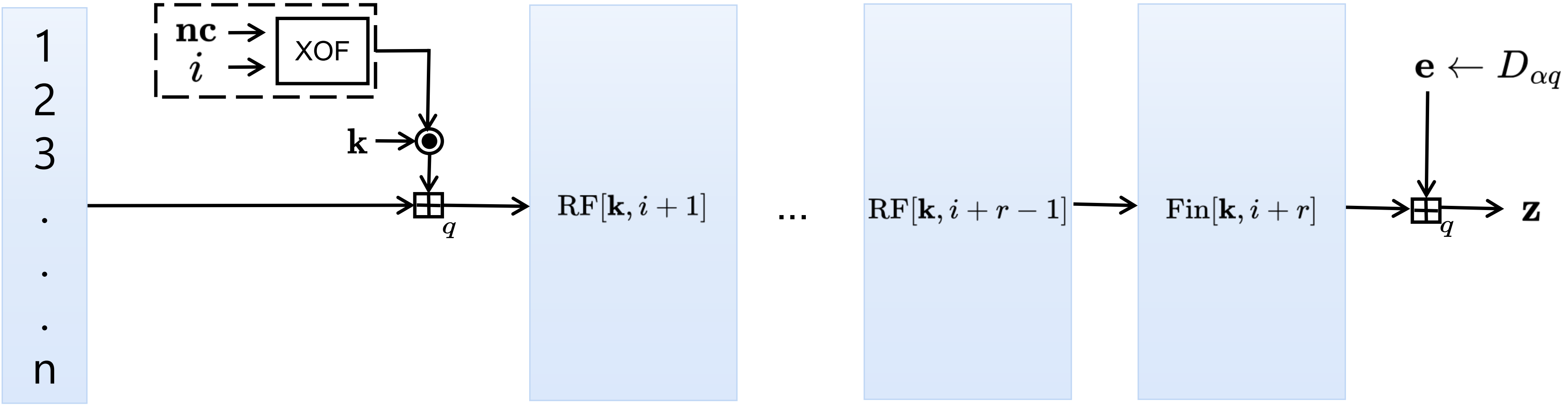
Rubato

$r-$ round Rubato


1
2
3
.
.
.
n
$\mathbf{nc}$
i
XOF
$\mathbf{k}$
q
$\mathrm{RF}[\mathbf{k}, i+1]$
...
$\mathrm{RF}[\mathbf{k}, i+r-1]$
$\mathrm{Fin}[\mathbf{k}, i+r]$
$\mathbf{e} \leftarrow D_{\alpha q}$
q
$\mathbf{z}$

Rubato

$r-$ round Rubato



1
2
3
.
.
.
n
nc
i
XOF
$\mathbf{k}$
q
$\mathrm{RF}[\mathbf{k}, i+1]$
...
$\mathrm{RF}[\mathbf{k}, i+r-1]$
$\mathrm{Fin}[\mathbf{k}, i+r]$
$\mathbf{e} \leftarrow D_{\alpha q}$
q
$\mathbf{z}$

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

Proposed parameters of Rubato

Key Recovery Attack on Rubato

1. Recover key and noise $\bmod\ m$, $m|q$
2. Recover positions in key stream with 0 noise
3. Set up system of polynomial equations and solve by linearization

Key Recovery Attack on Rubato

1. Recover key and noise $\bmod\ m$, $m|q$
2. Recover positions in key stream with 0 noise
3. Set up system of polynomial equations and solve by linearization

Notation:

- $\mathsf{Ru} = \mathsf{Rubato}$ without noise
- $w = \mathsf{Ru}[\mathbf{k}, \mathbf{nc}, i]$
- $\mathsf{Rubato}_m, \mathsf{Ru}_m$ execute steps in $\mathbb{Z}_m$

1. Recover key and noise $\bmod \ m|q$

Let $(k_1, \ldots, k_n) \in \mathbb{Z}_q^n$ and $z_i = \mathsf{Rubato}[\mathbf{k}, \mathbf{nc}, i]$, $1 \le i \le s$

$$\Rightarrow z_i = w_i + e_i \mod q \, , \, e_i \leftarrow D_{\alpha q}$$

1. Recover key and noise $\mathrm{mod}\ \ m|q$

Let $(k_1, \ldots, k_n) \in \mathbb{Z}_q^n$ and $z_i = \mathsf{Rubato}[\mathbf{k}, \mathbf{nc}, i]$, $1 \le i \le s$

$$\Rightarrow z_i = w_i + e_i \mod q\,,\; e_i \leftarrow D_{\alpha q}$$

Guess $\tilde{\mathbf{k}} = (\tilde{k}_1, \ldots, \tilde{k}_n) \in \mathbb{Z}_m^n \rightarrow \tilde{w}_i = \mathsf{Ru}_m[\tilde{\mathbf{k}}, \mathbf{nc}, i]$

$$\tilde{e}_i = (z_i \mod m) - \tilde{w}_i$$

$$\tilde{e}_i = e_i \mod m?$$

1. Recover key and noise $\bmod m | q$

Let $(k_1, \ldots, k_n) \in \mathbb{Z}_q^n$ and $z_i = \text{Rubato}[\mathbf{k}, \mathbf{nc}, i]$, $1 \leq i \leq s$

$$\Rightarrow z_i = w_i + e_i \mod q \,,\, e_i \leftarrow D_{\alpha q}$$

Guess $\tilde{\mathbf{k}} = (\tilde{k}_1, \ldots, \tilde{k}_n) \in \mathbb{Z}_m^n \to \tilde{w}_i = \text{Ru}_m[\tilde{\mathbf{k}}, \mathbf{nc}, i]$

$$\tilde{e}_i = (z_i \mod m) - \tilde{w}_i$$
$$\tilde{e}_i = e_i \mod m?$$

$$\begin{cases} \tilde{e}_i \text{ unif. random} \Rightarrow \text{WRONG GUESS} \\ \tilde{e}_i \leftarrow D_{\alpha q} \mod m \Rightarrow \text{RIGHT GUESS} \end{cases}$$

1. Recover key and noise $\bmod\ m|q$

Let $(k_1, \ldots, k_n) \in \mathbb{Z}_q^n$ and $z_i = \text{Rubato}[\mathbf{k}, \mathbf{nc}, i]\,,\ 1 \le i \le s$

$$\Rightarrow z_i = w_i + e_i \mod q\,,\ e_i \leftarrow D_{\alpha q}$$

Guess $\tilde{\mathbf{k}} = (\tilde{k}_1, \ldots, \tilde{k}_n) \in \mathbb{Z}_m^n \rightarrow \tilde{w}_i = \text{Ru}_m[\tilde{\mathbf{k}}, \mathbf{nc}, i]$

$$\tilde{e}_i = (z_i \mod m) - \tilde{w}_i$$

$$\tilde{e}_i = e_i \mod m?$$

$$\begin{cases} \tilde{e}_i \text{ unif. random} \Rightarrow \text{WRONG GUESS} \\ \tilde{e}_i \leftarrow D_{\alpha q} \mod m \Rightarrow \text{RIGHT GUESS} \end{cases}$$

$$e_i \mod m$$

$$k_i \mod m$$

2. Recover positions with 0 noise

Let $f|(q/m)$ such that $f = f_1 \cdots f_b\,,\; f_j \leq m$:

2. Recover positions with 0 noise

Let $f|(q/m)$ such that $f = f_1 \cdots f_b \, , \, f_j \leq m$:

- Repeat step 1 $\mod f_j m \Rightarrow k_i, e_i \mod f_j m \Rightarrow k_i, e_i \mod fm$

2. Recover positions with 0 noise

Let $f|(q/m)$ such that $f = f_1 \cdots f_b\,,\ f_j \leq m$:

- Repeat step 1 $\mod\ f_j m \Rightarrow\ k_i, e_i \quad \mod\ f_j m \Rightarrow\ k_i, e_i \quad \mod\ fm$
- f such that $|e_i| < fm$ with high probability so that

$$e_i \mod\ fm = 0 \Rightarrow e_i \mod\ q = 0$$

Note: $0 \leftarrow D_{\alpha q}$ at rate $1/\alpha q$

2. Recover positions with 0 noise

Let $f|(q/m)$ such that $f = f_1 \cdots f_b \,,\, f_j \leq m$:

- Repeat step 1 $\mod f_j m \Rightarrow k_i, e_i \mod f_j m \Rightarrow k_i, e_i \mod fm$
- f such that $|e_i| < fm$ with high probability so that

$$e_i \mod fm = 0 \Rightarrow e_i \mod q = 0$$

Note: $0 \leftarrow D_{\alpha q}$ at rate $1/\alpha q$

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

3. Key recovery for full Rubato key

For $i \in \mathcal{I} : w_i = z_i$

Set up system

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

3. Key recovery for full **Rubato** key

For $i \in \mathcal{I} : w_i = z_i$

Set up system

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

Solve by Gaussian elimination for every prime factor of q and combine using CRT

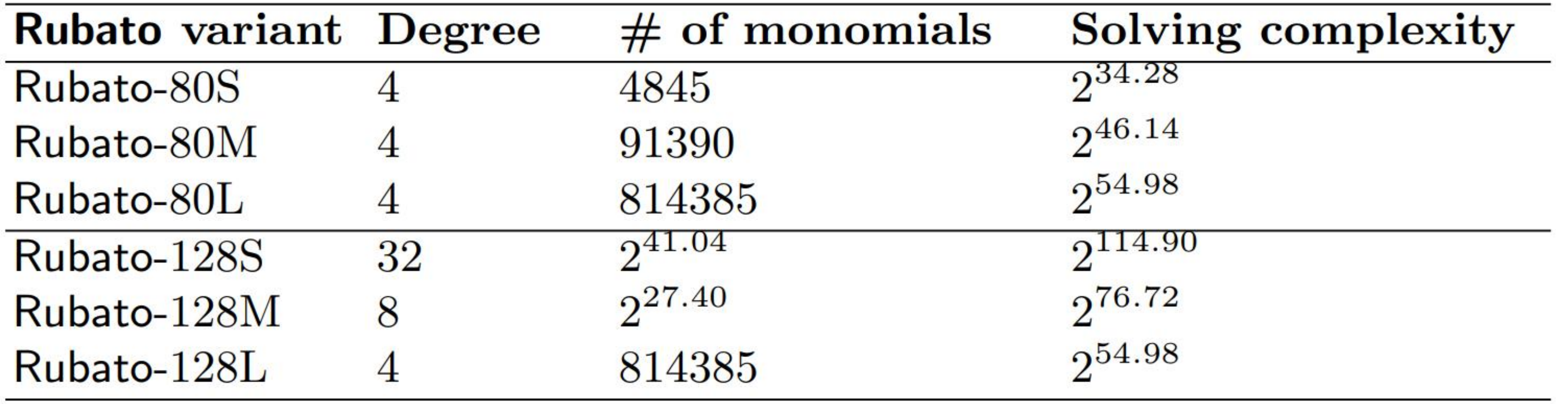
3. Key recovery for full Rubato key

For $i \in \mathcal{I} : w_i = z_i$

Set up system

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

Solve by Gaussian elimination for every prime factor of q and combine using CRT



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	$2^{54.98}$

Solving complexities for solving a linearized system of equations $\bmod\ p|q$

Assumptions on q

Assumption 1: $\exists m \in \mathbb{Z} : m|q\,,\ m_{\min} \leq m \leq m_{\max}$

Assumptions on q

Assumption 1: $\exists m \in \mathbb{Z} : m|q \, , \, m_{\min} \leq m \leq m_{\max}$

Determining $m_{\max}$

- In order to have a valid attack, we need $m^n < 2^\lambda$ for step 1

$$\Rightarrow m_{\max} = \lfloor 2^{\lambda/n} \rfloor$$

Assumptions on q

Assumption 1: $\exists m \in \mathbb{Z} : m|q \, , \, m_{\min} \leq m \leq m_{\max}$

Determining $m_{\min}$

- Smallest $m_{\min} \in \mathbb{Z}$ where it is possible to distinguish the correct key guess $\tilde{k} \mod m_{\min}$ from all the wrong ones

Assumptions on q

Assumption 1: $\exists m \in \mathbb{Z} : m|q\,,\; m_{\min} \le m \le m_{\max}$

Determining $m_{\min}$

- Smallest $m_{\min} \in \mathbb{Z}$ where it is possible to distinguish the correct key guess $\tilde{k} \mod m_{\min}$ from all the wrong ones
- The score function measures how much the candidate noise value produced by $\tilde{k}$ deviates from the uniform distribution (in the same way as values drawn from $D_{\alpha q} \mod m$ do)

Assumptions on q

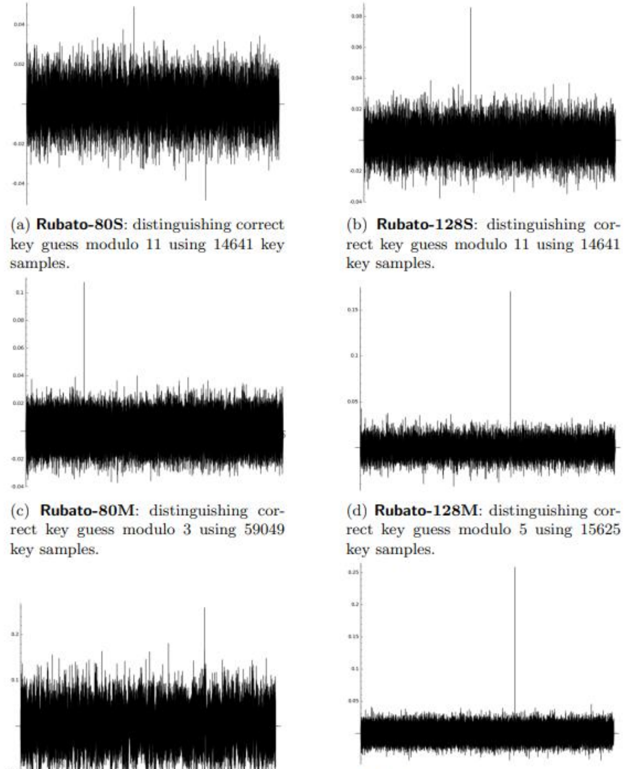
Assumption 1: $\exists m \in \mathbb{Z} : m|q\,,\ m_{\min} \leq m \leq m_{\max}$

Determining $m_{\min}$

- Smallest $m_{\min} \in \mathbb{Z}$ where it is possible to distinguish the correct key guess $\tilde{k} \mod m_{\min}$ from all the wrong ones
- The score function measures how much the candidate noise value produced by $\tilde{k}$ deviates from the uniform distribution (in the same way as values drawn from $D_{\alpha q} \mod m$ do)

→ High score $\Rightarrow$ RIGHT GUESS

→ Score ≈ 0 $\Rightarrow$ WRONG GUESS



(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.

Assumptions on q

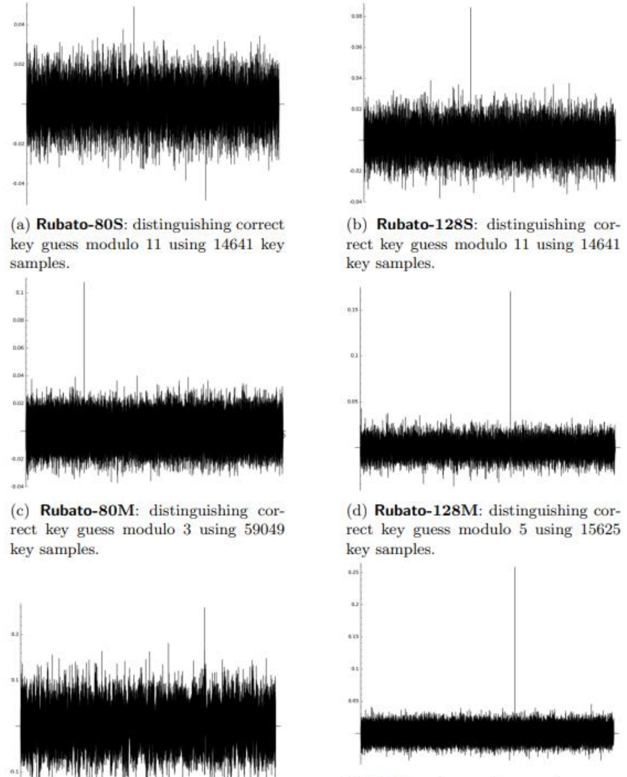
Assumption 1: $\exists m \in \mathbb{Z} : m | q\,,\ m_{\min} \leq m \leq m_{\max}$

Determining $m_{\min}$

- Smallest $m_{\min} \in \mathbb{Z}$ where it is possible to distinguish the correct key guess $\tilde{k} \mod m_{\min}$ from all the wrong ones
- The score function measures how much the candidate noise value produced by $\tilde{k}$ deviates from the uniform distribution (in the same way as values drawn from $D_{\alpha q} \mod m$ do)

  → High score $\Rightarrow$ RIGHT GUESS
  → Score ≈ 0 $\Rightarrow$ WRONG GUESS

- Caveat: wrong key guesses do NOT produce noise values that are distributed uniformly at random



(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.

$\Rightarrow$ Find $m_{\min}$ heuristically

Assumptions on q

Assumption 2: $\exists f \in \mathbb{Z}$: prime factors of $f \leq m\,,\; fm|q\,,\; fm > t$

Assumptions on q

Assumption 2: $\exists f \in \mathbb{Z}$: prime factors of $f \leq m$, $fm|q$, $fm > t$

Determining t

- Find smallest $t \in \mathbb{Z}$ such that for $fm > t, fm|q$ whp

$$e \mod (fm) = 0 \Rightarrow e \mod q = 0$$

Assumptions on q

Assumption 2: $\exists f \in \mathbb{Z}$: prime factors of $f \leq m$, $fm|q$, $fm > t$

Determining t

- Find smallest $t \in \mathbb{Z}$ such that for $fm > t, fm|q$ whp

$\Downarrow$ $$e \mod (fm) = 0 \Rightarrow e \mod q = 0$$

- Find smallest $t \in \mathbb{Z}$ such that $|e_i| < t$ whp $\forall 1 \leq i \leq s$

Assumptions on q

Assumption 2: $\exists f \in \mathbb{Z}$: prime factors of $f \leq m$, $fm|q$, $fm > t$

Determining t

- Find smallest $t \in \mathbb{Z}$ such that for $fm > t, fm|q$ whp

$\Downarrow$

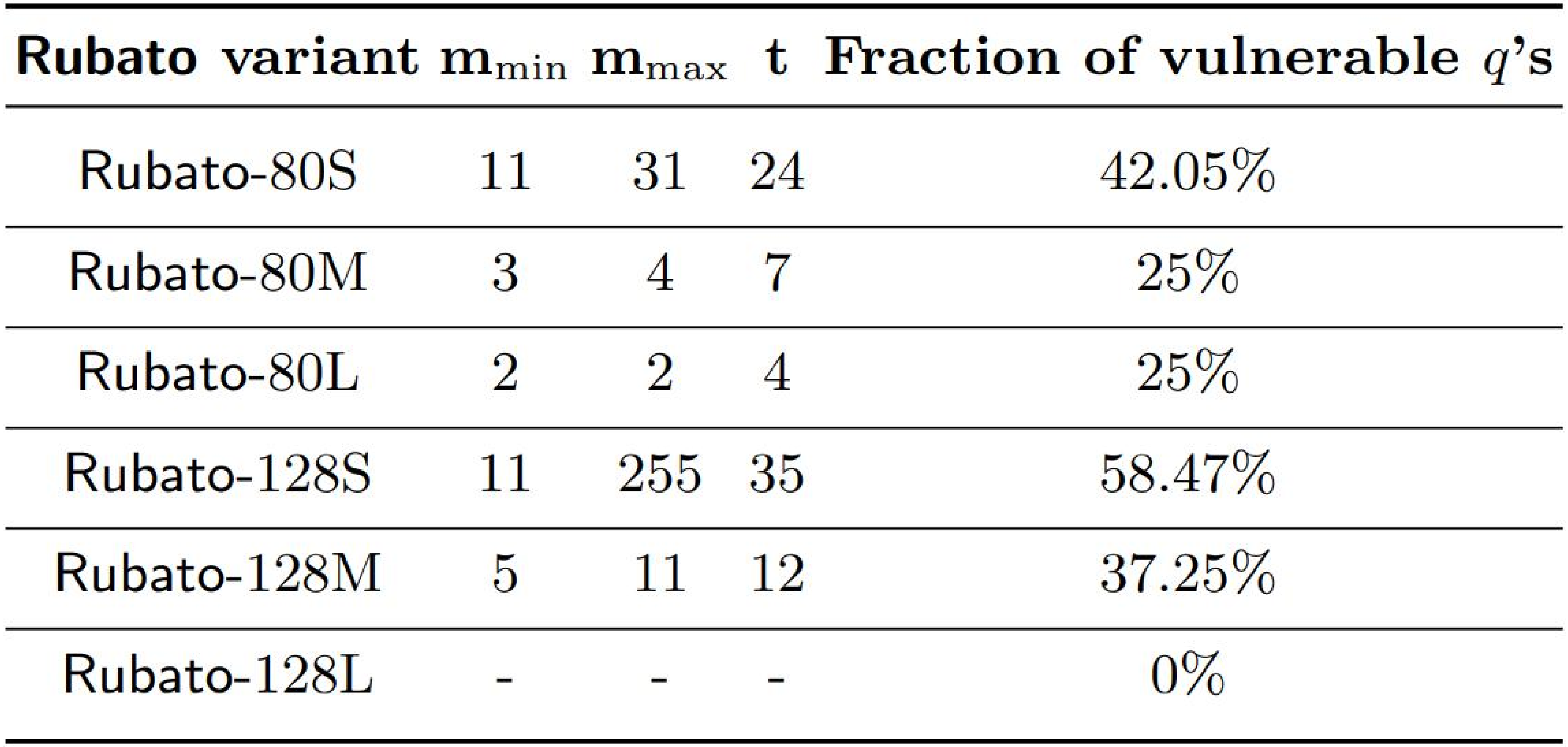
$$e \mod (fm) = 0 \Rightarrow e \mod q = 0$$

- Find smallest $t \in \mathbb{Z}$ such that $|e_i| < t$ whp $\forall 1 \leq i \leq s$

$\Downarrow$

- Find smallest $t \in \mathbb{Z}$ such that $0.99 \leq \Pr(|e_i| < t)^s = \left(\sum_{x=-t}^{t} \frac{1}{\alpha q} e^{-x^2/2\sigma^2} \right)^s$

Set of susceptible values



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%

Practical verification of the attack

Experiments[1] determining $m_{\min}$

[1] Experimental verification at https://github.com/Simula-UiB/RubatoAttack

Practical verification of the attack

Experiments[1] determining $m_{\min}$

- Select a 25 or 26-bit q with some small factors k at random → Produce 10000 elements of **Rubato** key stream

[1] Experimental verification at https://github.com/Simula-UiB/RubatoAttack

Practical verification of the attack

Experiments[1] determining $m_{\min}$

- Select a 25 or 26-bit q with some small factors k at random → Produce 10000 elements of **Rubato** key stream
- Fix value of m → make $11^4 - 2^{16}$ guesses on the key $\bmod\ m$ (including $k \mod m$) and store score values in a file

[1] Experimental verification at https://github.com/Simula-UiB/RubatoAttack

Practical verification of the attack

Experiments[1] determining $m_{\min}$

- Select a 25 or 26-bit q with some small factors k at random → Produce 10000 elements of Rubato key stream
- Fix value of $m \rightarrow$ make $11^4 - 2^{16}$ guesses on the key $\bmod m$ (including $k \mod m$) and store score values in a file
- Make plots of the score values as a bar chart

[1] Experimental verification at https://github.com/Simula-UiB/RubatoAttack

Practical verification of the attack

Experiments[1] determining $m_{\min}$

- Select a 25 or 26-bit q with some small factors k at random → Produce 10000 elements of Rubato key stream
- Fix value of m → make $11^4 - 2^{16}$ guesses on the key $\bmod\ m$ (including $k \mod m$) and store score values in a file
- Make plots of the score values as a bar chart
- Verify that the maximum score value seen corresponds to $k \mod m$

[1] Experimental verification at https://github.com/Simula-UiB/RubatoAttack

Attack complexities

Lowest attack complexity when $m = m_{\min}$, $f = 2^g = 2^{\lceil \log(t/m) \rceil}$

Key recovery attack complexity:

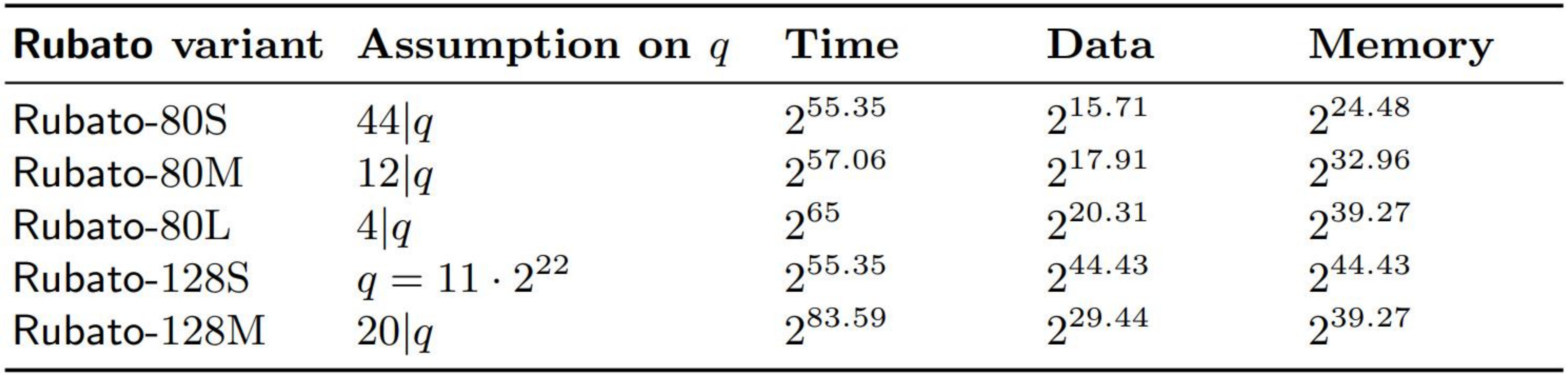
$$C_{kr} = \underset{\substack{\uparrow \\ \text{Step 1}}}{m^n} + \underset{\substack{\uparrow \\ \text{Step 2}}}{g} \cdot 2^n + \underset{\substack{\uparrow \\ \text{Step 3}}}{C_{lin}}$$

Attack complexities

Lowest attack complexity when $m = m_{\min}$, $f = 2^g = 2^{\lceil \log(t/m) \rceil}$

Key recovery attack complexity:

$$C_{kr} = \underset{\substack{\uparrow \\ \text{Step 1}}}{m^n} + \underset{\substack{\uparrow \\ \text{Step 2}}}{g} \cdot 2^n + \underset{\substack{\uparrow \\ \text{Step 3}}}{C_{lin}}$$



Rubato variant	**Assumption on** q	**Time**	**Data**	**Memory**
Rubato-80S	$44\|q$	$2^{55.35}$	$2^{15.71}$	$2^{24.48}$
Rubato-80M	$12\|q$	$2^{57.06}$	$2^{17.91}$	$2^{32.96}$
Rubato-80L	$4\|q$	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	$20\|q$	$2^{83.59}$	$2^{29.44}$	$2^{39.27}$

Lowest time complexities of key recovery attack

Restoring the security of Rubato

- Prime q

+ Immune to small-factor attack

Restoring the security of Rubato

- Prime q

  + Immune to small-factor attack

- Bigger αq

  + Cannot distinguish $k \mod m$ for $m < 2^{\lambda/n}$
  - More noise $\Rightarrow$ loss of precision and accuracy

Restoring the security of Rubato

- Prime q

  + Immune to small-factor attack

- Bigger αq

  + Cannot distinguish $k \mod m$ for $m < 2^{\lambda/n}$

  - More noise $\Rightarrow$ loss of precision and accuracy

- More rounds

  + Linearization step high solving complexity

  - Higher multiplicative depth $\Rightarrow$ loss of efficiency

Restoring the security of Rubato

- Prime q

  + Immune to small-factor attack

- Bigger αq

  + Cannot distinguish $k \mod m$ for $m < 2^{\lambda/n}$

  - More noise $\Rightarrow$ loss of precision and accuracy

- More rounds

  + Linearization step high solving complexity

  - Higher multiplicative depth $\Rightarrow$ loss of efficiency

- Non-polynomial S-boxes

  + No polynomial representation of S-box

  - Loss of efficiency

Conclusion

- Key recovery attack of 5/6 instances of Rubato for at least 25% of the choices of q
- Experimental verification of the attack
- Security of symmetric primitives over rings $\mathbb{Z}_q$

Thank you for your attention

More details in https://eprint.iacr.org/2023/822