Rubato:
Noisy Ciphers for Approximate Homomorphic Encryption

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

[1]KAIST, Daejeon, Korea
[2]Samsung SDS, Seoul, Korea

KAIST SAMSUNG SDS

Eurocrypt 2022

Homomorphic Encryption

- Homomorphic encryption (HE) is an encryption scheme that enables addition and multiplication over encrypted data without decryption key*
  - $\mathrm{a} * \mathrm{b} = \mathrm{Dec}\big(\mathrm{Enc}(a) * \mathrm{Enc}(b)\big)(+\epsilon)$
  - E.g., FV ($\mathbb{Z}_t$, +, ×), CKKS($\mathbb{C}$, +, ×)
- HE can protect data even when it is being used
  - E.g., ML inference, statistics of sensitive data on a cloud server

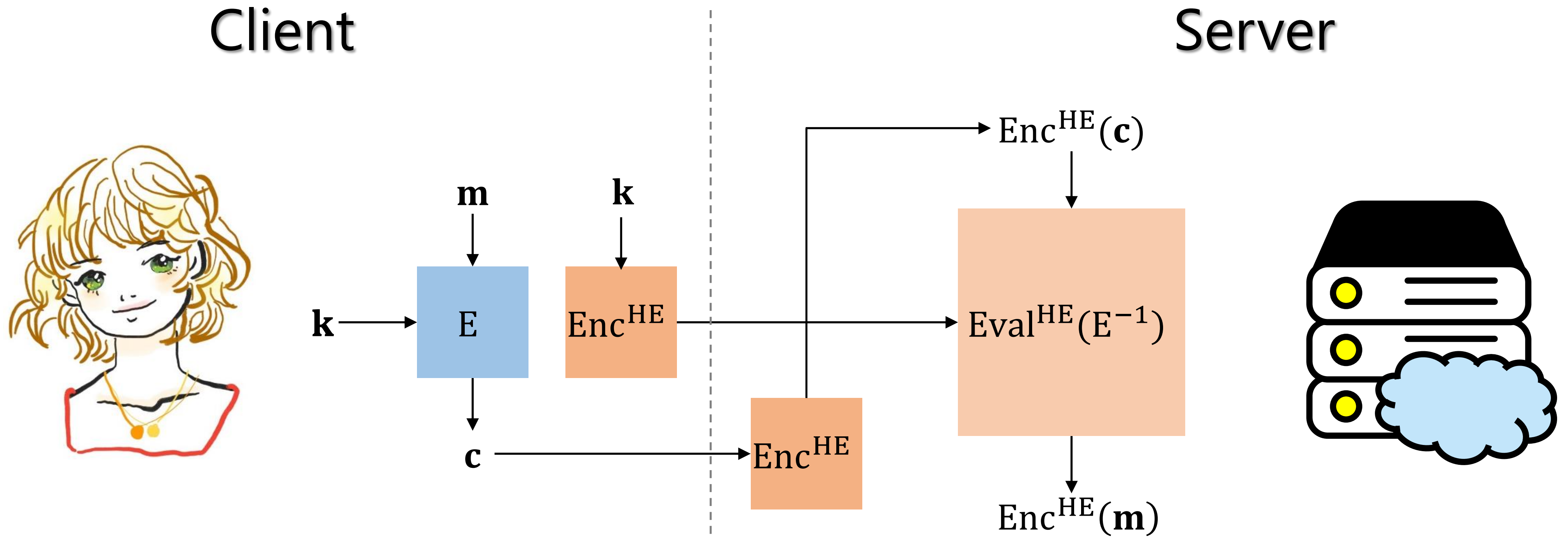
* We do not take into account partially homomorphic encryption (PHE) in this talk.

Demerit of HE

- Slow encryption speed
  - Slower than usual public key encryption
  - Inadequate to bulk encryption

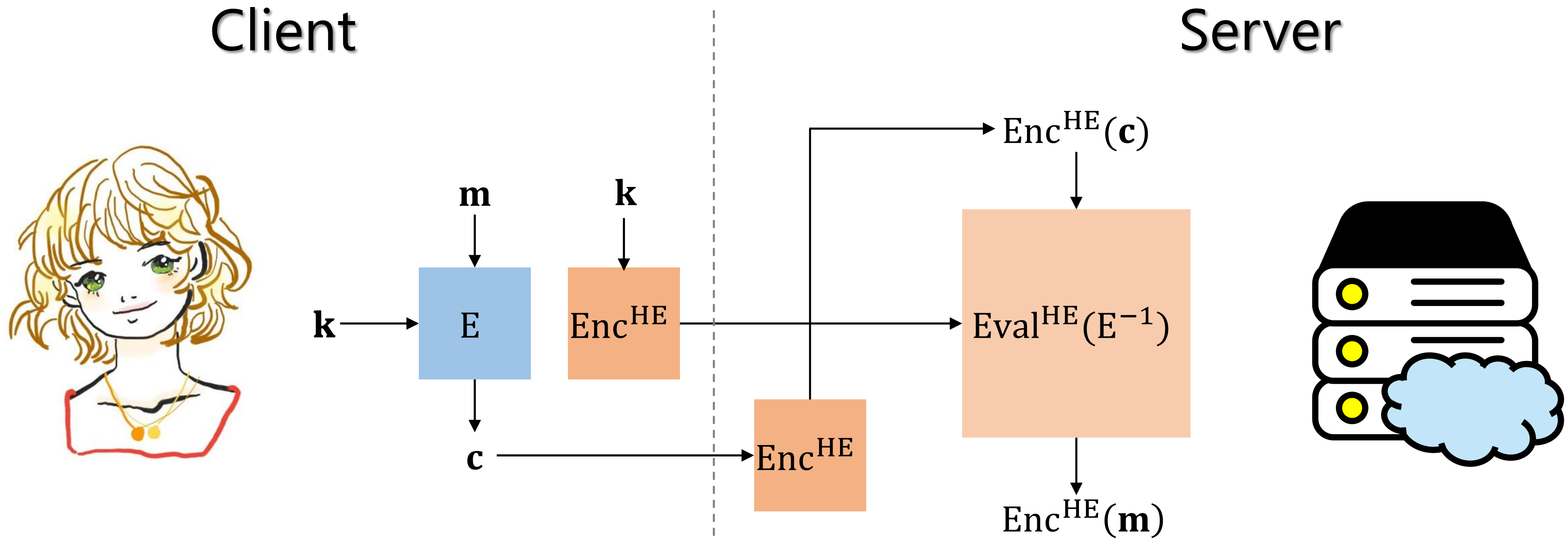
- Large ciphertext expansion
  - 10x – 1,000,000x according to the choice of parameters
  - Disadvantage for encryption of small messages
  - Large memory & network bandwidth overhead

Transciphering Framework



* K. Lauter et al., “Can Homomorphic Encryption Be Practical?”, ACM CCSW 2011

Transciphering Framework

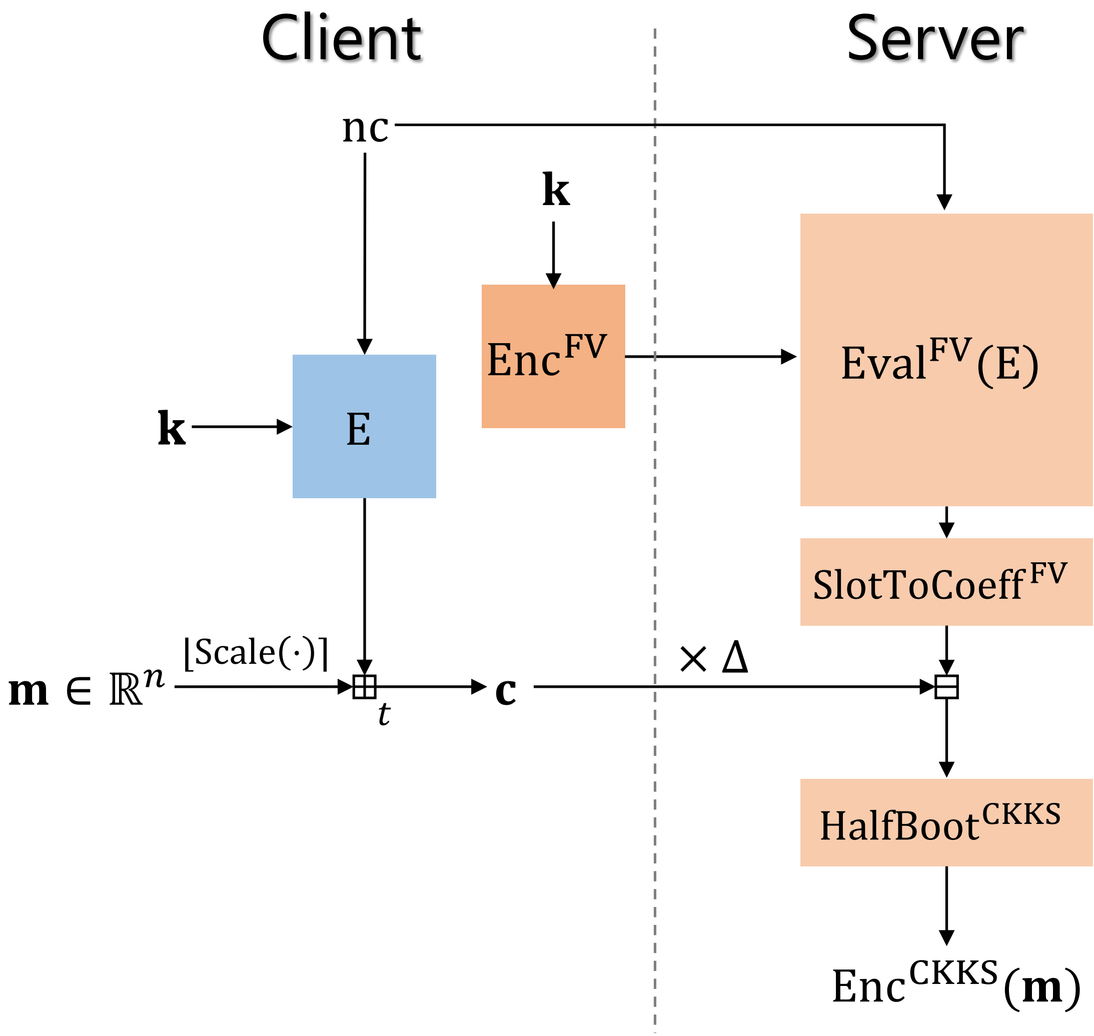


Fast encryption speed
Small ciphertext expansion

* K. Lauter et al., "Can Homomorphic Encryption Be Practical?", ACM CCSW 2011

RtF Transciphering Framework

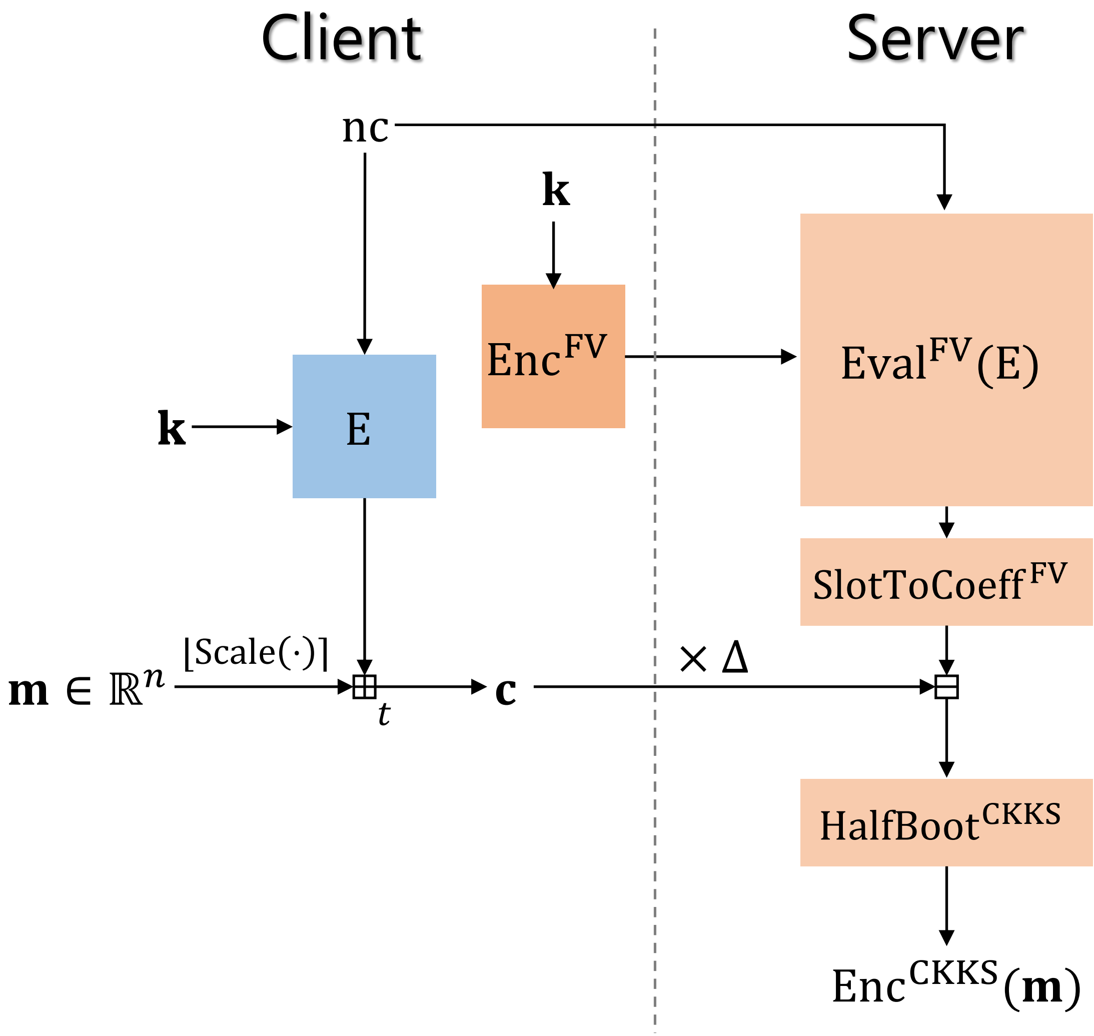
- In Asiacrypt 2021, RtF framework was proposed for approximate numbers*



* J. Cho et al., "Transciphering Framework for Approximate Homomorphic Encryption", Asiacrypt 2021

RtF Transciphering Framework

- In Asiacrypt 2021, RtF framework was proposed for approximate numbers*
- Client-side symmetric encryption over $\mathbb{Z}_t$
  - Message in $\mathbb{R}$
  - Ciphertext in $\mathbb{Z}_t^*$
- FV → CKKS Conversion by CKKS bootstrapping
  - FV-evaluation of the cipher
  - CKKS bootstrapping w/o last SlotToCoeff
  - Result: CKKS-ciphertext



* J. Cho et al., "Transciphering Framework for Approximate Homomorphic Encryption", Asiacrypt 2021

HE-friendly Ciphers

- HE-friendly cipher is a cipher which is efficiently evaluated using HE
- New design strategy is required
  - So far, AND gates and XOR gates are roughly the same in most hardware
  - However, cost of XOR gate (addition) is way cheaper than AND gate (multiplication) in HE setting
  - Low multiplicative depth/complexity required

HE-friendly Ciphers

- HE-friendly cipher is a cipher which is efficiently evaluated using HE
- New design strategy is required
  - So far, AND gates and XOR gates are roughly the same in most hardware
  - However, cost of XOR gate (addition) is way cheaper than AND gate (multiplication) in HE setting
  - Low multiplicative depth/complexity required
- Domain-critical cipher
  - Computation on that domain after transciphering
  - Binary: LowMC, Kreyvium, FLIP, Rasta, Dasta
  - Modulo: Masta, Pasta, HERA
  - Approximate: HERA, Rubato

Main Question

Is there any way to reduce
the multiplicative depth drastically?

Observation

- Deterministic cipher requires a certain amount of multiplicative depth with reasonable key size
  - (FLIP) 1394 bit key size → Mult. depth = 4
  - (Rasta) 351 bit key size → Mult. depth = 6

Observation

- Deterministic cipher requires a certain amount of multiplicative depth with reasonable key size
  - (FLIP) 1394 bit key size → Mult. depth = 4
  - (Rasta) 351 bit key size → Mult. depth = 6
- LWE encryption does not require non-scalar multiplication (and is secure!)
  - But it requires large key size and lots of random bits
  - Client-side encryption speed is too slow

Observation

- Deterministic cipher requires a certain amount of multiplicative depth with reasonable key size
  - (FLIP) 1394 bit key size → Mult. depth = 4
  - (Rasta) 351 bit key size → Mult. depth = 6
- LWE encryption does not require non-scalar multiplication (and is secure!)
  - But it requires large key size and lots of random bits
  - Client-side encryption speed is too slow

Cipher	LowMC	FLIP	Rasta	Masta	HERA	Pasta	LWE
Modulus	2	2	2	$t \approx 2^{25}$	$t \approx 2^{25}$	$t \approx 2^{25}$	$t \approx 2^{25}$
#(Key words)	256	1394	351	16	16	64	1024
Mult. Depth	14	4	6	7	10	5	0
#Mult / word	10.34	1072	6	7	10	9.81	0
Random bits / word	0	13287	2464	400	150	250	25600

Idea: Mix Together!

- Stream cipher + Gaussian noise

Idea: Mix Together!

- Stream cipher + Gaussian noise
- Security against algebraic attacks
  - Gröbner basis attack
    - $\text{Gröbner}(n, m, d)$ → $\text{Gröbner}(n, m, d') \cdot \text{Guess}(e \leftarrow \chi)$
    - Guessing error takes more time → lower degree
  - Arora-Ge attack
    - $\prod_{e=-t\alpha q}^{t\alpha q}(b_i - \langle \boldsymbol{a}_i, \boldsymbol{s} \rangle - e) = 0$ → $\prod_{e=-t\alpha q}^{t\alpha q}(b_i - F(\boldsymbol{a}_i, \boldsymbol{s}) - e) = 0$
    - Equations gets larger degree with the same success probability

Idea: Mix Together!

- Stream cipher + Gaussian noise
- Security against algebraic attacks
  - Gröbner basis attack
    - $\mathrm{Gröbner}(n, m, d)$ → $\mathrm{Gröbner}(n, m, d') \cdot \mathrm{Guess}(e \leftarrow \chi)$
    - Guessing error takes more time → lower degree
  - Arora-Ge attack
    - $\prod_{e=-t\alpha q}^{t\alpha q}(b_i - \langle \boldsymbol{a}_i, \boldsymbol{s} \rangle - e) = 0$ → $\prod_{e=-t\alpha q}^{t\alpha q}(b_i - F(\boldsymbol{a}_i, \boldsymbol{s}) - e) = 0$
    - Equations gets larger degree with the same success probability

- LWE decryption needs round-off function
  - Originally, round-off function denoise the LWE noise
  - For approximate computation, LWE noise can be regarded as error
  - No need to round off

Noisy Cipher Rubato*

- Stream cipher + Gaussian noise
- SPN with randomized key schedule
- HERA-like linear layer + Pasta-like S-box layer
- Fixed constant input

* Tempo rubato: (musical term) expressive and rhythmic freedom

Noisy Cipher Rubato*

- Stream cipher + Gaussian noise
- SPN with randomized key schedule
- HERA-like linear layer + Pasta-like S-box layer
- Fixed constant input

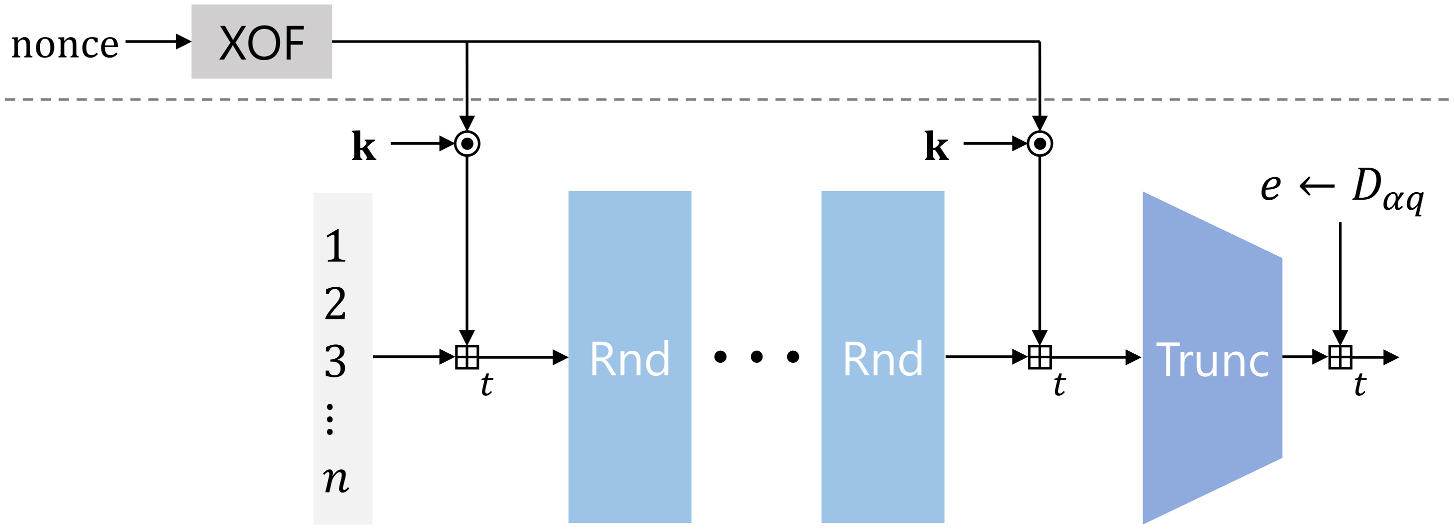


Fig. Rubato

* Tempo rubato: (musical term) expressive and rhythmic freedom

Design Aspects of Rubato

- Various block size (S:16, M:36, L:64)
  - When block size is larger, the required number of rounds decreases
  - Trade-off between throughput and latency
- HERA-like linear layers
  - Invertible MDS circulant matrix
  - Small component size
  - $\text{MixRows} \circ \text{MixColumns}$

Design Aspects of Rubato

- Various block size (S:16, M:36, L:64)
  - When block size is larger, the required number of rounds decreases
  - Trade-off between throughput and latency
- HERA-like linear layers
  - Invertible MDS circulant matrix
  - Small component size
  - MixRows ∘ MixColumns

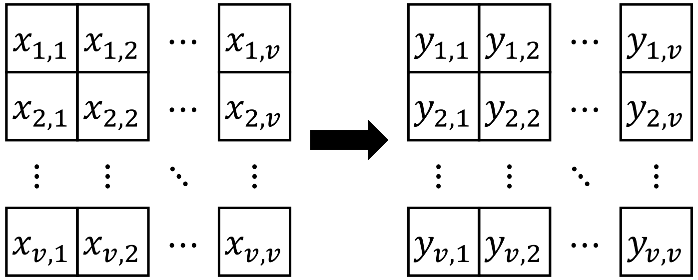


Fig 1. Change of states

Design Aspects of Rubato

- Various block size (S:16, M:36, L:64)
  - When block size is larger, the required number of rounds decreases
  - Trade-off between throughput and latency
- HERA-like linear layers
  - Invertible MDS circulant matrix
  - Small component size
  - MixRows ∘ MixColumns

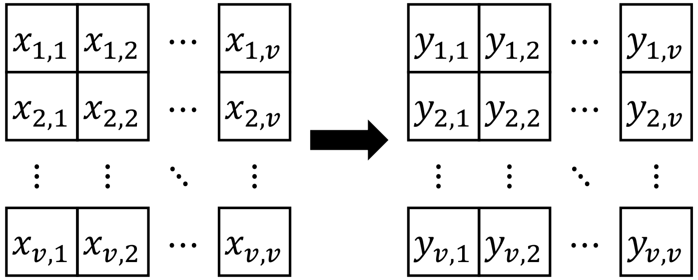


Fig 1. Change of states

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

Fig 2a. MixColumns

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

Fig 2b. MixRows

Design Aspects of Rubato

- Various block size (S:16, M:36, L:64)
  - When block size is larger, the required number of rounds decreases
  - Trade-off between throughput and latency
- HERA-like linear layers
  - Invertible MDS circulant matrix
  - Small component size
  - MixRows $\circ$ MixColumns

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

Fig 3. MDS matrices

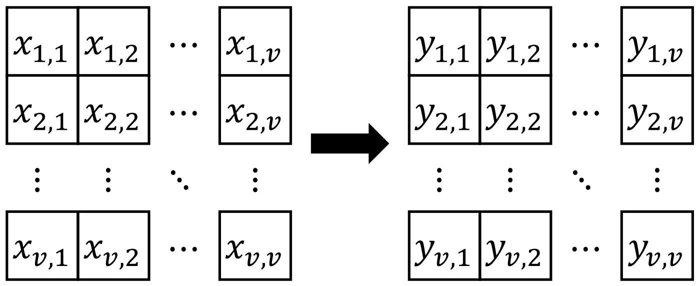


Fig 1. Change of states

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

Fig 2a. MixColumns

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

Fig 2b. MixRows

Design Aspects of Rubato

- Feistel network in a row
  - $\mathrm{Feistel}(\mathbf{x}) = (x_1, x_2 + x_1^2, \dots, x_n + x_{n-1}^2)$
  - Quadratic function

- Truncation
  - $\mathrm{Trunc}_{\mathrm{n},\ell}(\mathbf{x}) = (x_1, \dots, x_\ell)$
  - It prevents algebraic meet-in-the-middle attack

- Adding Gaussian noise
  - $\mathrm{AGN}(\mathbf{x}) = (x_1 + e_1, \dots, x_n + e_n)$
  - e_i's are sampled from a discrete Gaussian distribution

Design Aspects of Rubato

- Feistel network in a row
  - $\text{Feistel}(\mathbf{x}) = (x_1, x_2 + x_1^2, \dots, x_n + x_{n-1}^2)$
  - Quadratic function
- Truncation
  - $\text{Trunc}_{\text{n},\ell}(\mathbf{x}) = (x_1, \dots, x_\ell)$
  - It prevents algebraic meet-in-the-middle attack
- Adding Gaussian noise
  - $\text{AGN}(\mathbf{x}) = (x_1 + e_1, \dots, x_n + e_n)$
  - e_i's are sampled from a discrete Gaussian distribution

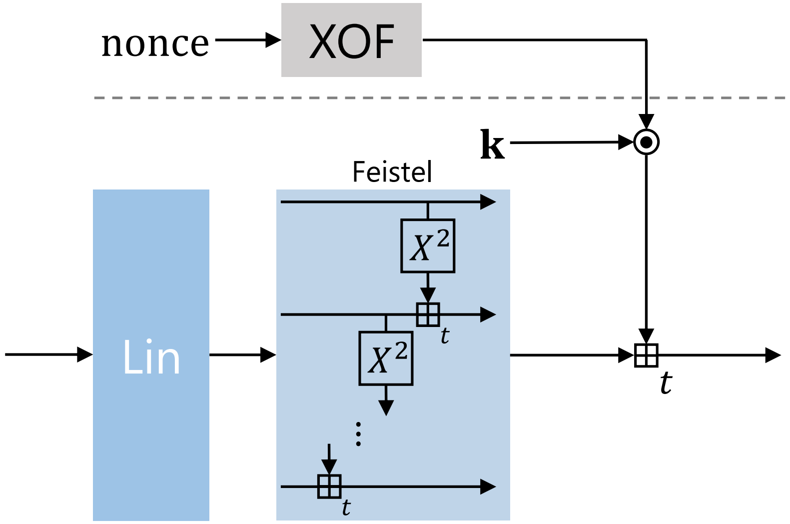


Fig. Round function of Rubato

MULT-related Value Comparison

Cipher	LowMC	FLIP	Rasta	Masta	HERA	Pasta	LWE	**Rubato**
Modulus	2	2	2	$t \approx 2^{25}$	$t \approx 2^{25}$	$t \approx 2^{25}$	$t \approx 2^{25}$	$t \approx 2^{25}$
#(Key words)	256	1394	351	16	16	64	1024	64
Mult. Depth	14	4	6	7	10	5	0	2
#Mult / word	10.34	1072	6	7	10	9.81	0	2.1
Random bits / word	0	13287	2464	400	150	250	25600	80

Security Analysis of Rubato

- Symmetric cryptanalysis with guess
  - LC / DC
  - Trivial linearization / Interpolation attack
  - GCD / Gröbner basis attack

$$(\boldsymbol{a}, F(\boldsymbol{a}, \boldsymbol{s}) + e) \Rightarrow (\boldsymbol{a}, F(\boldsymbol{a}, \boldsymbol{s}))$$

guess

Security Analysis of Rubato

- Symmetric cryptanalysis with guess
  - LC / DC
  - Trivial linearization / Interpolation attack
  - GCD / Gröbner basis attack

$$(\boldsymbol{a}, F(\boldsymbol{a}, \boldsymbol{s}) + e) \Rightarrow (\boldsymbol{a}, F(\boldsymbol{a}, \boldsymbol{s}))$$

guess

- LWE cryptanalysis with linearization
  - Lattice attacks (e.g., SIS, BDD, uSVP strategy)
  - BKW attack

$$s_i s_j = s_{ij}'$$

$$(\boldsymbol{a}, F(\boldsymbol{a}, \boldsymbol{s}) + e) \Rightarrow (\boldsymbol{a}', \langle \boldsymbol{a}', \boldsymbol{s}' \rangle + e)$$

Security Analysis of Rubato

- Symmetric cryptanalysis with guess
  - LC / DC
  - Trivial linearization / Interpolation attack
  - GCD / Gröbner basis attack

$$(\boldsymbol{a}, F(\boldsymbol{a}, \boldsymbol{s}) + e) \Rightarrow (\boldsymbol{a}, F(\boldsymbol{a}, \boldsymbol{s}))$$

guess

- LWE cryptanalysis with linearization
  - Lattice attacks (e.g., SIS, BDD, uSVP strategy)
  - BKW attack

$$s_i s_j = s_{ij}'$$

$$(\boldsymbol{a}, F(\boldsymbol{a}, \boldsymbol{s}) + e) \Rightarrow (\boldsymbol{a}', \langle \boldsymbol{a}', \boldsymbol{s}' \rangle + e)$$

- Arora-Ge attack

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

Selected Parameters of Rubato

Parameter	Sec.	Block size	Trunc. size	$\log t$	$\sigma/\sqrt{2\pi}$ *	Round
Par-80S	80	16	12	26	11.1	2
Par-80M		36	32	25	2.7	2
Par-80L		64	60	25	1.6	2
Par-128S	128	16	12	26	10.5	5
Par-128M		36	32	25	4.1	3
Par-128L		64	60	25	4.1	2

* σ is the standard deviation of the discrete Gaussian distribution

Complexity of the Attacks on Rubato

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

Table. The log of the complexity of the attacks on Rubato ($\omega = 2$)

Performance

- Performance is evaluated with AVX2 instruction/RtF framework
- XOF: SHAKE256
- (N, #slots, remaining level) = (2^{16}, 2^{16}, 7)

Scheme	Ct size (B)	Ct. Exp. Ratio	Client		Server		Prec. (bits)
			Lat. (cycle)	Thrp. (C/B)	Lat. (s)	Thrp. (KB/s)	
Par-80S	37.5	1.31	5906	199.1	41.23	6.676	18.8
Par-80M	100	1.25	11465	143.5	57.15	7.032	19.0
Par-80L	187.5	1.25	16679	110.9	115.44	6.520	19.1
Par-128S	37.5	1.31	10446	351.8	71.06	6.083	18.8
Par-128M	100	1.26	14292	179.7	88.35	6.666	18.9
Par-128L	187.5	1.26	16920	113.5	106.43	6.712	18.9

Performance Comparison

Scheme	$\log N$	Log of #slots	Ct size (KB)	Ct. Exp. Ratio	Client		Server		Prec. (bits)	Level
					Lat. (μs)	Thrp. (MB/s)	Lat. (s)	Thrp. (KB/s)		
RtF-HERA	16	16	0.055	1.24	1.520	25.26	141.58	5.077	19.1	7
RtF-Rubato	16	16	0.183	1.26	4.585	31.04	106.4	6.712	18.9	7
LWE *	16	9	0.007	4.84	21.91	0.051	65.88	0.010	9.3	7
CKKS only	14	14	468	23.25	9596	2.035	none		19.1	7

* W. Lu et al., "Pegasus: Bridging Polynomial and Non-polynomial Evaluations in Homomorphic Encryption", IEEE S&P 2021

Conclusion

- Summary
  - We present a family of noisy ciphers for approximate homomorphic encryption
  - It is a combination of stream cipher and Gaussian noise
  - We give modular cryptanalysis for noisy ciphers
  - We show that the noisy ciphers are efficient in approximate homomorphic encryption

- Further question
  - Is there any other application of noisy ciphers?
  - Is there any cryptanalysis which exploits both stream cipher structure and noise?
    - Linearized lattice problem?

Thank you!

Check out the full version at
ia.cr/2022/537