Rubato:
Noisy Ciphers for Approximate Homomorphic Encryption

Jincheol Ha¹, Seongkwang Kim²,
Byeonghak Lee¹, Jooyoung Lee¹,
and Mincheol Son¹

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

Eurocrypt 2022
Homomorphic Encryption

• Homomorphic encryption (HE) is an encryption scheme that enables addition and multiplication over encrypted data without decryption key*
  • $a \times b = \text{Dec}(\text{Enc}(a) \times \text{Enc}(b))(\pm \epsilon)$
  • E.g., FV ($\mathbb{Z}_t$, $+$, $\times$), CKKS($\mathbb{C}$, $+$, $\times$)

• 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

Client

![Client Diagram]

Server

![Server Diagram]

Transciphering Framework

Client

Fast encryption speed
Small ciphertext expansion

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

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

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 $\rightarrow$ CKKS Conversion by CKKS bootstrapping
  • FV-evaluation of the cipher
  • CKKS bootstrapping w/o last SlotToCoeff
  • Result: CKKS-ciphertext

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 \( \rightarrow \) Mult. depth = 4
  - (Rasta) 351 bit key size \( \rightarrow \) 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

<table>
<thead>
<tr>
<th>Cipher</th>
<th>LowMC</th>
<th>FLIP</th>
<th>Rasta</th>
<th>Masta</th>
<th>HERA</th>
<th>Pasta</th>
<th>LWE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modulus</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>( t \approx 2^{25} )</td>
<td>( t \approx 2^{25} )</td>
<td>( t \approx 2^{25} )</td>
<td>( t \approx 2^{25} )</td>
</tr>
<tr>
<td>#(Key words)</td>
<td>256</td>
<td>1394</td>
<td>351</td>
<td>16</td>
<td>16</td>
<td>64</td>
<td>1024</td>
</tr>
<tr>
<td>Mult. Depth</td>
<td>14</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>10</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>#Mult / word</td>
<td>10.34</td>
<td>1072</td>
<td>6</td>
<td>7</td>
<td>10</td>
<td>9.81</td>
<td>0</td>
</tr>
<tr>
<td>Random bits / word</td>
<td>0</td>
<td>13287</td>
<td>2464</td>
<td>400</td>
<td>150</td>
<td>250</td>
<td>25600</td>
</tr>
</tbody>
</table>
Idea: Mix Together!

- Stream cipher + Gaussian noise
Idea: Mix Together!

- Stream cipher + Gaussian noise
- Security against algebraic attacks
  - Gröbner basis attack
    - $\text{Gröbner}(n, m, d) \rightarrow \text{Gröbner}(n, m', d') \cdot \text{Guess}(e \leftarrow \chi)$
    - Guessing error takes more time $\rightarrow$ lower degree
- Arora-Ge attack
  - $\prod_{e=-t}^{t}(b_i - (a_i, s) - e) = 0 \rightarrow \prod_{e=-t}^{t}(b_i - F(a_i, s) - e) = 0$
  - Equations gets larger degree with the same success probability
Idea: Mix Together!

- Stream cipher + Gaussian noise
- Security against algebraic attacks
  - Gröbner basis attack
    - Gröbner\((n, m, d) \rightarrow Gröbner(n, m, d') \cdot \text{Guess}(e \leftarrow \chi)\)
    - Guessing error takes more time \(\rightarrow\) lower degree
  - Arora-Ge attack
    - \(\sum_{e=-\tau q}^{\tau q} (b_i - (a_i, s) - e) = 0 \rightarrow \sum_{e=-\tau q}^{\tau q} (b_i - F(a_i, s) - e) = 0\)
    - 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

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

\[ \text{nonce} \rightarrow \text{XOF} \]

\[ k \]

\[ 1 \rightarrow Rnd \rightarrow \cdots \rightarrow Rnd \rightarrow \text{Trunc} \]

\[ t \]

\[ e \leftarrow D_{\alpha q} \]

* Fig. Rubato
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
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

\[ \begin{array}{cccc}
  x_{1,1} & x_{1,2} & \cdots & x_{1,v} \\
  x_{2,1} & x_{2,2} & \cdots & x_{2,v} \\
  \vdots & \vdots & \ddots & \vdots \\
  x_{v,1} & x_{v,2} & \cdots & x_{v,v} \\
\end{array} \quad \begin{array}{cccc}
  y_{1,1} & y_{1,2} & \cdots & y_{1,v} \\
  y_{2,1} & y_{2,2} & \cdots & y_{2,v} \\
  \vdots & \vdots & \ddots & \vdots \\
  y_{v,1} & y_{v,2} & \cdots & y_{v,v} \\
\end{array} \]

\[ \text{Fig 1. Change of states} \]
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 $\circ$ MixColumns

\[
\begin{bmatrix}
  x_{1,1} & x_{1,2} & \cdots & x_{1,v} \\
  x_{2,1} & x_{2,2} & \cdots & x_{2,v} \\
  \vdots & \vdots & \ddots & \vdots \\
  x_{v,1} & x_{v,2} & \cdots & x_{v,v}
\end{bmatrix} \quad \begin{bmatrix}
  y_{1,1} & y_{1,2} & \cdots & y_{1,v} \\
  y_{2,1} & y_{2,2} & \cdots & y_{2,v} \\
  \vdots & \vdots & \ddots & \vdots \\
  y_{v,1} & y_{v,2} & \cdots & y_{v,v}
\end{bmatrix}
\]

**Fig 1.** Change of states

\[
\begin{bmatrix}
  y_{1,c} \\
  y_{2,c} \\
  \vdots \\
  y_{v,c}
\end{bmatrix} = M_v \cdot \begin{bmatrix}
  x_{1,c} \\
  x_{2,c} \\
  \vdots \\
  x_{v,c}
\end{bmatrix} \quad \begin{bmatrix}
  y_{c,1} \\
  y_{c,2} \\
  \vdots \\
  y_{c,v}
\end{bmatrix} = M_v \cdot \begin{bmatrix}
  x_{c,1} \\
  x_{c,2} \\
  \vdots \\
  x_{c,v}
\end{bmatrix}
\]

**Fig 2a.** MixColumns **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 ◦ MixColumns

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

**Fig 1.** Change of states

\[
\begin{bmatrix}
x_{1,1} & x_{1,2} & \cdots & x_{1,v} \\
x_{2,1} & x_{2,2} & \cdots & x_{2,v} \\
\vdots & \vdots & \ddots & \vdots \\
x_{v,1} & x_{v,2} & \cdots & x_{v,v}
\end{bmatrix}
\begin{bmatrix}
y_{1,1} & y_{1,2} & \cdots & y_{1,v} \\
y_{2,1} & y_{2,2} & \cdots & y_{2,v} \\
\vdots & \vdots & \ddots & \vdots \\
y_{v,1} & y_{v,2} & \cdots & y_{v,v}
\end{bmatrix} = \mathbf{M}_v \cdot
\begin{bmatrix}
x_{1,c} \\
x_{2,c} \\
\vdots \\
x_{v,c}
\end{bmatrix}
\]

**Fig 2a.** MixColumns

\[
\begin{bmatrix}
Y_{1,c} \\
Y_{2,c} \\
\vdots \\
Y_{v,c}
\end{bmatrix} = \mathbf{M}_v \cdot
\begin{bmatrix}
X_{1,c} \\
X_{2,c} \\
\vdots \\
X_{v,c}
\end{bmatrix}
\]

**Fig 2b.** MixRows

**Fig 3.** MDS matrices
Design Aspects of Rubato

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

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

- Adding Gaussian noise
  - \( \text{AGN}(x) = (x_1 + e_1, \ldots, x_n + e_n) \)
  - \( e_i \)'s are sampled from a discrete Gaussian distribution
• Feistel network in a row
  • Feistel(x) = (x₁, x₂ + x₁², ..., xₙ + xₙ⁻¹²)
  • Quadratic function

• Truncation
  • Truncₙ,ℓ(x) = (x₁, ..., xₖ)
  • It prevents algebraic meet-in-the-middle attack

• Adding Gaussian noise
  • AGN(x) = (x₁ + e₁, ..., xₙ + eₙ)
  • eᵢ’s are sampled from a discrete Gaussian distribution

---

**Fig.** Round function of Rubato
## MULT-related Value Comparison

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

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

\[(a, F(a, s) + e) \quad \text{guess} \quad \rightarrow \quad (a, F(a, s))\]
Security Analysis of Rubato

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

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

\[
\begin{align*}
(a, F(a, s) + e) & \quad \text{guess} \quad \rightarrow \quad (a, F(a, s)) \\
(s_i s_j = s_{ij}') & \quad \rightarrow \quad (a', \langle a', s' \rangle + e) \\
(a, F(a, s) + e) & \quad \rightarrow \quad (a', \langle a', s' \rangle + e)
\end{align*}
\]
Security Analysis of Rubato

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

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

- Arora-Ge attack

\[
\prod_{e=-t\alpha q}^{t\alpha q} (b_i - \langle a_i, s \rangle - e) = 0 \Rightarrow \prod_{e=-t\alpha q}^{t\alpha q} (b_i - F(a_i, s) - e) = 0
\]
## Selected Parameters of Rubato

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

* $\sigma$ is the standard deviation of the discrete Gaussian distribution
Complexity of the Attacks on Rubato

<table>
<thead>
<tr>
<th>Parameter</th>
<th>GCD</th>
<th>Gröbner</th>
<th>LC</th>
<th>Lattice</th>
<th>Arora-Ge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Par-80S</td>
<td>393.6</td>
<td>80.04</td>
<td>155.9</td>
<td>760.5</td>
<td>80.04</td>
</tr>
<tr>
<td>Par-80M</td>
<td>878.6</td>
<td>84.55</td>
<td>249.9</td>
<td>↑</td>
<td>80.37</td>
</tr>
<tr>
<td>Par-80L</td>
<td>↑</td>
<td>82.73</td>
<td>349.8</td>
<td>↑</td>
<td>82.73</td>
</tr>
<tr>
<td>Par-128S</td>
<td>411.9</td>
<td>128.1</td>
<td>311.7</td>
<td>↑</td>
<td>128.1</td>
</tr>
<tr>
<td>Par-128M</td>
<td>880.7</td>
<td>128.1</td>
<td>249.9</td>
<td>↑</td>
<td>128.1</td>
</tr>
<tr>
<td>Par-128L</td>
<td>↑</td>
<td>169.6</td>
<td>349.8</td>
<td>↑</td>
<td>129.6</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Ct size (B)</th>
<th>Ct. Exp. Ratio</th>
<th>Client</th>
<th>Server</th>
<th>Prec. (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Lat. (cycle)</td>
<td>Thrp. (C/B)</td>
<td>Lat. (s)</td>
</tr>
<tr>
<td>Par-80S</td>
<td>37.5</td>
<td>1.31</td>
<td>5906</td>
<td>199.1</td>
<td>41.23</td>
</tr>
<tr>
<td>Par-80M</td>
<td>100</td>
<td>1.25</td>
<td>11465</td>
<td>143.5</td>
<td>57.15</td>
</tr>
<tr>
<td>Par-80L</td>
<td>187.5</td>
<td>1.25</td>
<td>16679</td>
<td>110.9</td>
<td>115.44</td>
</tr>
<tr>
<td>Par-128S</td>
<td>37.5</td>
<td>1.31</td>
<td>10446</td>
<td>351.8</td>
<td>71.06</td>
</tr>
<tr>
<td>Par-128M</td>
<td>100</td>
<td>1.26</td>
<td>14292</td>
<td>179.7</td>
<td>88.35</td>
</tr>
<tr>
<td>Par-128L</td>
<td>187.5</td>
<td>1.26</td>
<td>16920</td>
<td>113.5</td>
<td>106.43</td>
</tr>
</tbody>
</table>
## Performance Comparison

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

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