Fregata：Faster Homomorphic Evaluation of AES via TFHE

Benqiang Wei, Ruida Wang, Zhihao Li, Qinju Liu and Xianhui Lu

1. Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
2. School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

Fully Homomorphic Encryption

May 9, 2024

中国科学院信息工程研究所 INSTITUTE OF INFORMATION ENGINEERING,CAS

中国科学院大学 University of Chinese Academy of Sciences

Outline

- Background and Motivation
  - Fully Homomorphic encryption
  - Hybrid Homomorphic encryption
- Homomorphic evaluation of AES
  - AES-128
  - The state-of-the-art
  - Our evaluation
  - Recent improvement
- Performance

Fully Homomorphic Encryption

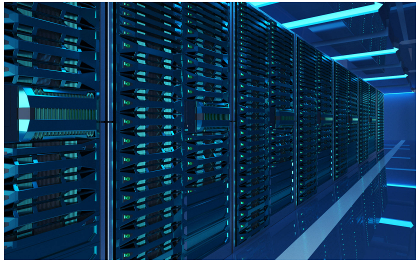
"Holy Grail of Cryptography"

Fully homomorphic encryption (FHE) enables the computation of arbitrary functions to be performed on encrypted data without decryption

(1) $\mathrm{C} = \mathrm{Enc}_{\mathrm{HE}}(m)$

C

C'



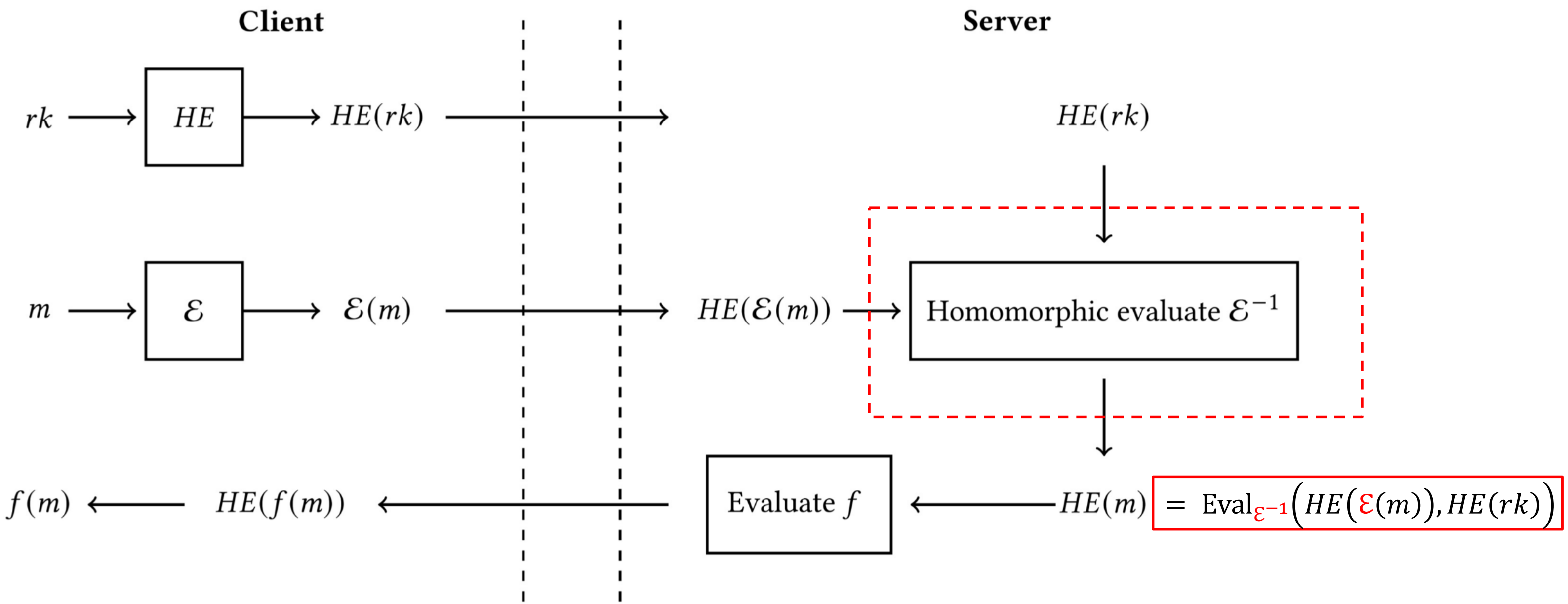
(3) $f(m) = \mathrm{Dec}_{\mathrm{HE}}(\mathrm{C}')$

(2) $\mathrm{C}' = \mathrm{Eval}_f(\mathrm{C})$

Drawbacks of FHE application:

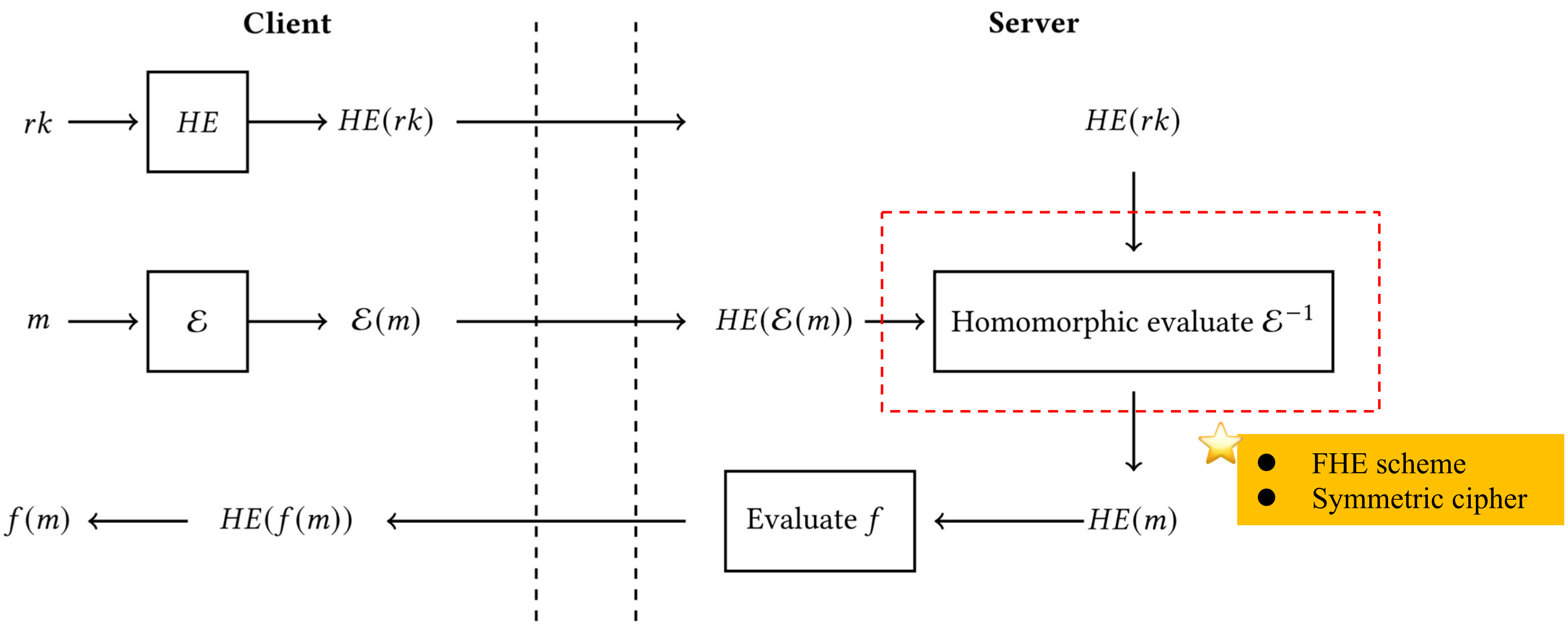
- Slow evaluation: $Eval_f$
- Ciphertext size expansion: LWE($O(n \log q)$), RLWE($O(\log q)$) C

Hybrid Homomorphic Encryption（HHE）



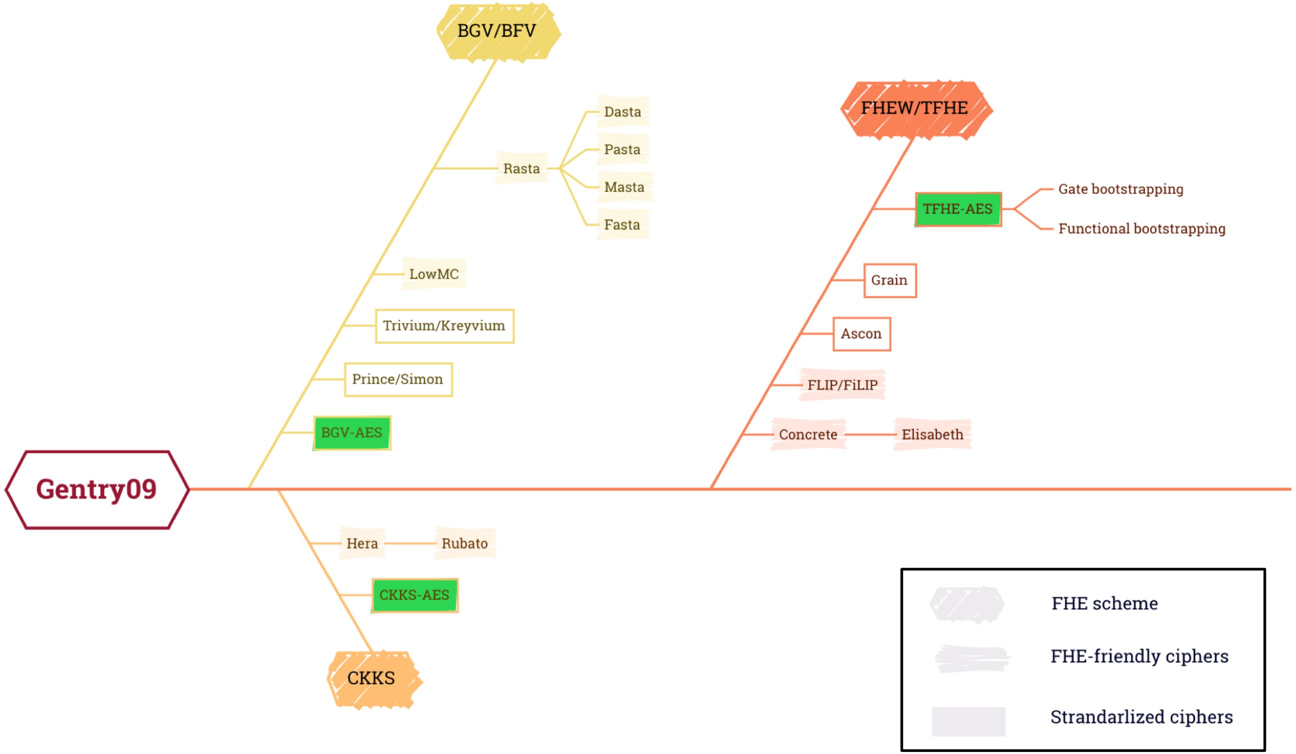
HHE(aka, Transciphering)[NLV11] can reduce the transmission communication cost between the client and the server by integrating a symmetric encryption scheme (ε)

Hybrid Homomorphic Encryption（HHE）



HHE(aka, Transciphering)[NLV11] can reduce the transmission communication cost between the client and the server by integrating a symmetric encryption scheme $(\mathcal{E})$

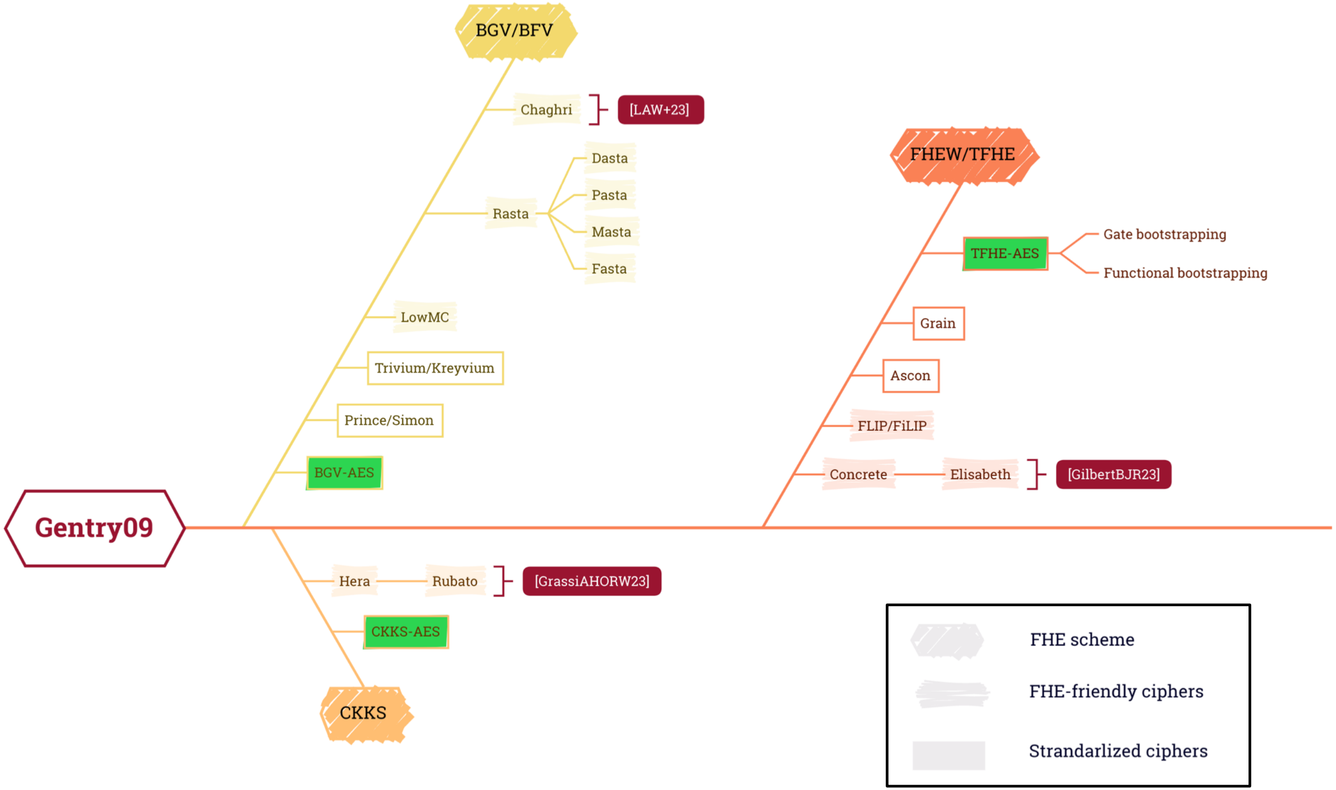
Overview of FHE and HHE



FHE and Ciphers in HHE

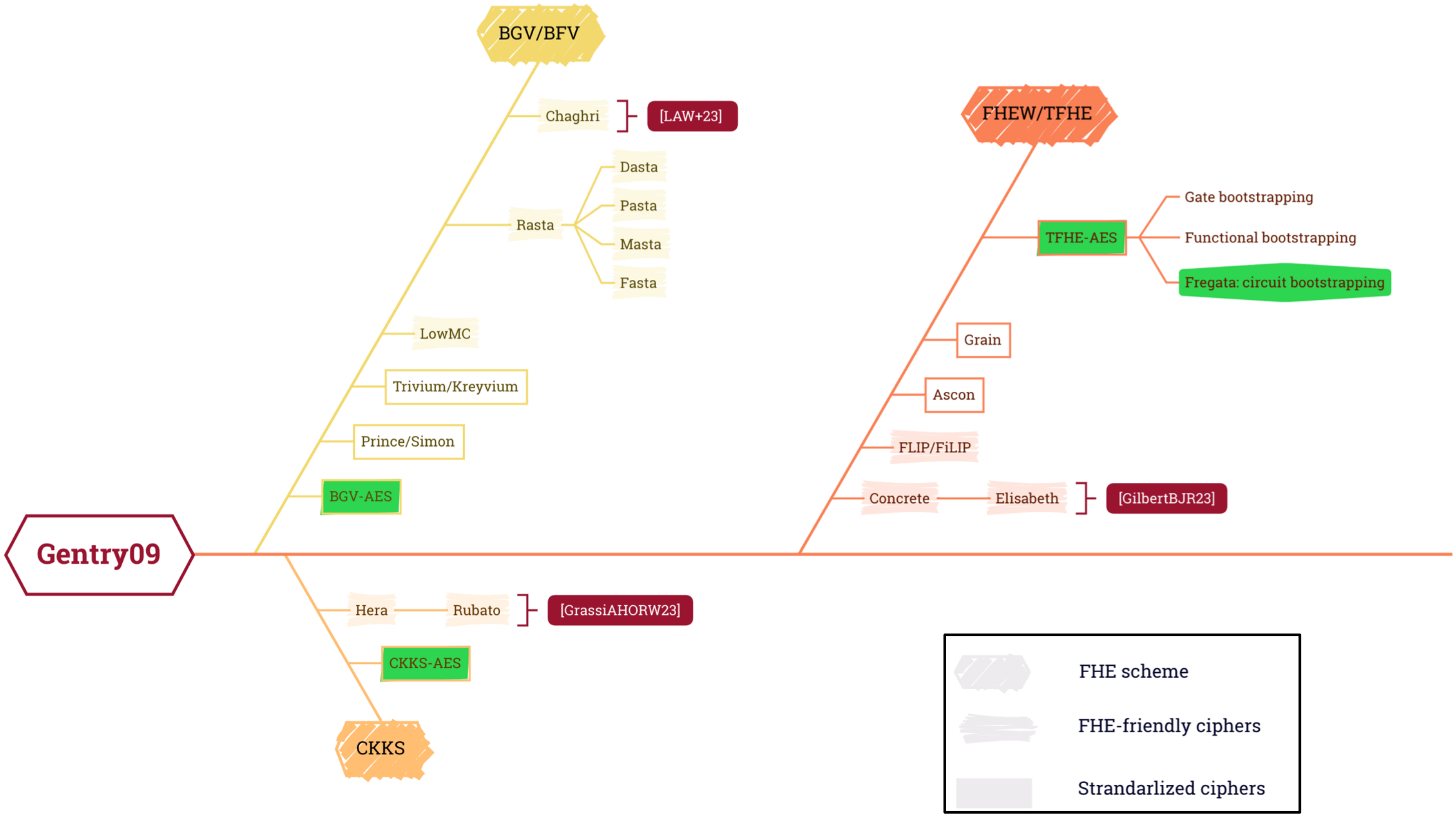
FHE-friendly ciphers: (1) low multiplicative complexity (2) low multiplicative depth (3) secure

Some Attacks about FHE-friendly Ciphers



[LAW+23]Fukang Liu, Ravi Anand, Libo Wang, Willi Meier, and Takanori Isobe. Co efficient grouping: Breaking chaghri and more. In Advances in Cryptology-EUROCRYPT2023
[GrassiAHORW23]Lorenzo Grassi, Irati Manterola Ayala, Martha Norberg Hovd, MortenØy garden, Håvard Raddum, and Qingju Wang. Cryptanalysis of symmetric primitives over rings and a key recovery attack on rubato. In Advances in Cryptology-CRYPTO2023
[Gilbert BJR23] Henri Gilbert, Rachelle Heim Boissier, Jérémy Jean, and Jean-René Reinhard. Cryptanalysis of elisabeth-4. IACR Cryptol. ePrint Arch., page 1436, 2023- ASIACRYPT2023.

Motivation: Focus on Standarlized Cipher AES



AES-128 specification

State matrix $\begin{pmatrix} A_0 & A_4 & A_8 & A_{12} \\ A_1 & A_5 & A_9 & A_{13} \\ A_2 & A_6 & A_{10} & A_{14} \\ A_3 & A_7 & A_{11} & A_{15} \end{pmatrix} \in \mathbb{F}_{2^8}^{16}$

➢ SubBytes

$$\begin{pmatrix} B_0 & B_4 & B_8 & B_{12} \\ B_1 & B_5 & B_9 & B_{13} \\ B_2 & B_6 & B_{10} & B_{14} \\ B_3 & B_7 & B_{11} & B_{15} \end{pmatrix} = \begin{pmatrix} S(A_0) & S(A_4) & S(A_8) & S(A_{12}) \\ S(A_1) & S(A_5) & S(A_9) & S(A_{13}) \\ S(A_2) & S(A_6) & S(A_{10}) & S(A_{14}) \\ S(A_3) & S(A_7) & S(A_{11}) & S(A_{15}) \end{pmatrix}$$

➢ ShiftRows

$$\begin{pmatrix} B_0 & B_4 & B_8 & B_{12} \\ B_1 & B_5 & B_9 & B_{13} \\ B_2 & B_6 & B_{10} & B_{14} \\ B_3 & B_7 & B_{11} & B_{15} \end{pmatrix} \Longrightarrow \begin{pmatrix} B_0 & B_4 & B_8 & B_{12} \\ B_5 & B_9 & B_{13} & B_1 \\ B_{10} & B_{14} & B_2 & B_6 \\ B_{15} & B_3 & B_7 & B_{11} \end{pmatrix}$$

➢ MixColumns

$$\begin{pmatrix} C_0 & C_4 & C_8 & C_{12} \\ C_1 & C_5 & C_9 & C_{13} \\ C_2 & C_6 & C_{10} & C_{14} \\ C_3 & C_7 & C_{11} & C_{15} \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} B_0 & B_4 & B_8 & B_{12} \\ B_5 & B_9 & B_{13} & B_1 \\ B_{10} & B_{14} & B_2 & B_6 \\ B_{15} & B_3 & B_7 & B_{11} \end{pmatrix}$$

➢ AddRoundKey

$$\begin{pmatrix} A_0 & A_4 & A_8 & A_{12} \\ A_1 & A_5 & A_9 & A_{13} \\ A_2 & A_6 & A_{10} & A_{14} \\ A_3 & A_7 & A_{11} & A_{15} \end{pmatrix} = \begin{pmatrix} C_0 & C_4 & C_8 & C_{12} \\ C_1 & C_5 & C_9 & C_{13} \\ C_2 & C_6 & C_{10} & C_{14} \\ C_3 & C_7 & C_{11} & C_{15} \end{pmatrix} + \begin{pmatrix} rk_{i,0} & rk_{i,4} & rk_{i,8} & rk_{i,12} \\ rk_{i,1} & rk_{i,5} & rk_{i,9} & rk_{i,13} \\ rk_{i,2} & rk_{i,6} & rk_{i,10} & rk_{i,14} \\ rk_{i,3} & rk_{i,7} & rk_{i,11} & rk_{i,15} \end{pmatrix}$$

- ➢ Linear function: ShiftRows, MixColumns, AddRoundKey
- ➢ Nonlinear function: SubBytes(S-box)

The state-of-the-art

Different methods

BGV: SIMD addition and multiplication [GHS12]

- ✓ SubBytes: $X^{-1} = X^{254}$ + affine transformation(Frobenius automorphism)
- ✓ RowShifts: rotation(automorphism)
- ✓ MixColumns: affine transformation
- ✓ AddRoundKey： plaintext modolus $t = 2$

CKKS: approximate computation [ADE+23]

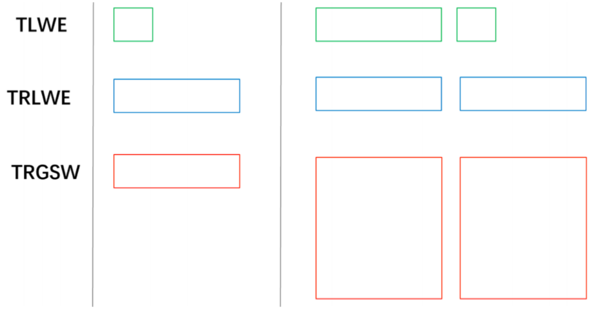
- ✓ SubBytes: lookup table by comparing
- ✓ RowShifts: free
- ✓ MixColumns: XOR + bit-shifting
- ✓ AddRoundKey: $(X - Y)^2$

TFHE: bit(s)-wise encryption [TCBS23]

- ✓ SubBytes: S-box lookup table(Functional bootstrapping)
- ✓ RowShifts: free
- ✓ MixColumns: Mulx2 and Mulx3 table(Functional bootstrapping)
- ✓ AddRoundKey： 4-by-4-bit XOR table (Functional bootstrapping)

The TFHE cryptosystem

1、Ciphertext type



2、Building blocks

- External Multiplication $\boxdot$: TRGSW × TRLWE → TRLWE
- CMux gate: CMux$(c, d_1, d_0) = c \boxdot (d_1 - d_0) + d_0$
- SampleExtraction：
  - TRLWE-to-TLWE
- KeySwitching：
  - TLWE-to-TLWE、TLWE-to-TRLWE、TRLWE-to-TRLWE
- BlindRotation：
  - Rotate the test polynomial blindly using encrypted numbers.

3、Bootstrapping type

- Identity bootstrapping
  - Keep the message still while refreshing the noise
- Gate bootstrapping
  - Perform gate operations while refreshing noise
- Functional bootstrapping or Programmable bootstrapping
  - Evaluate arbitrary function lookup table while refreshing noise
- Multi-value bootstrapping
  - Evaluate multiple functions at the same time using just one bootstrapping
- Circuit bootstrapping
  - TLWE-to-TRGSW ciphertext conversion

Our observation

Message Encoding and Evaluation Strategy are very important in FHE

- Linear function: ShiftRows, MixColumns, AddRoundKey

Plaintext modulus p=2

$$b_0b_1b_2b_3b_4b_5b_6b_7 \xrightarrow{\times 1} b_0b_1b_2b_3b_4b_5b_6b_7$$

$$b_0b_1b_2b_3b_4b_5b_6b_7 \xrightarrow{\times x} b_7b_0b_1b_2b_3b_4b_5b_6 \oplus 0b_70b_7b_7000$$

$$b_0b_1b_2b_3b_4b_5b_6b_7 \xrightarrow{\times (x+1)} b_0b_1b_2b_3b_4b_5b_6b_7 \oplus b_7b_0b_1b_2b_3b_4b_5b_6 \oplus 0b_70b_7b_7000$$

Cheap

- Nonlinear function: SubBytes(S-box)

Expensive

Our observation

Message Encoding and Evaluation Strategy are very important in FHE

- Linear function: ShiftRows, MixColumns, AddRoundKey

Plaintext modulus p=2

$$b_0b_1b_2b_3b_4b_5b_6b_7 \xrightarrow{\times 1} b_0b_1b_2b_3b_4b_5b_6b_7$$

$$b_0b_1b_2b_3b_4b_5b_6b_7 \xrightarrow{\times x} b_7b_0b_1b_2b_3b_4b_5b_6 \oplus 0b_70b_7b_7000$$

$$b_0b_1b_2b_3b_4b_5b_6b_7 \xrightarrow{\times (x+1)} b_0b_1b_2b_3b_4b_5b_6b_7 \oplus b_7b_0b_1b_2b_3b_4b_5b_6 \oplus 0b_70b_7b_7000$$

Cheap

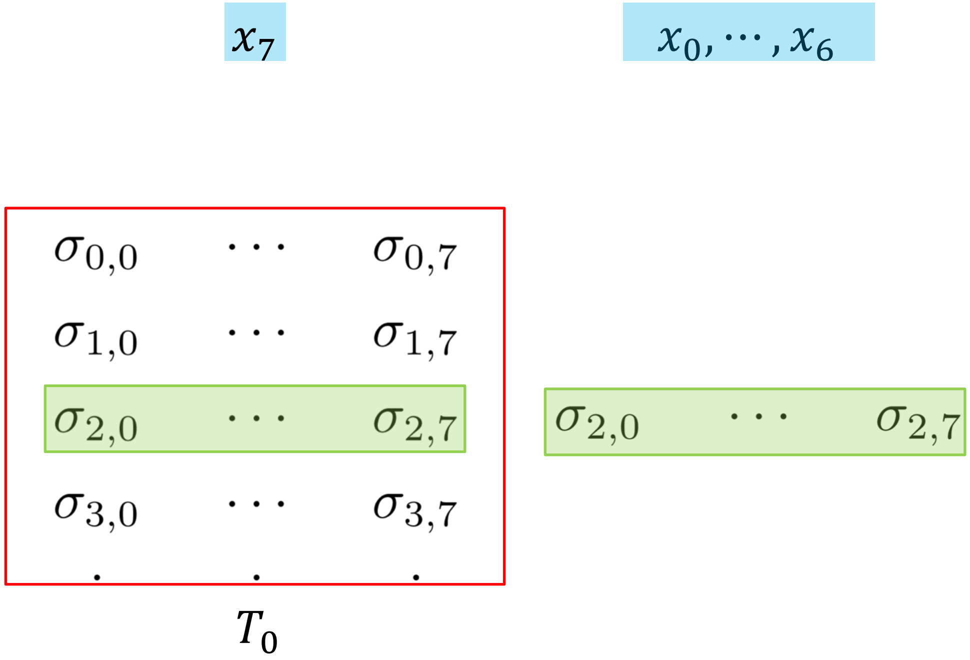
- Nonlinear function: SubBytes(S-box)

Expensive → CMux → Cheap

S-box LUT via CMux gate and mixed packing

Suppose the dimension of the ring polynomial is $N = 1024$

x_0	$\cdots$	x_7	f_0	$\cdots$	f_7	
0	$\cdots$	0	$\sigma_{0,0}$	$\cdots$	$\sigma_{0,7}$	T_0
1	$\cdots$	0	$\sigma_{1,0}$	$\cdots$	$\sigma_{1,7}$	
0	$\cdots$	0	$\sigma_{2,0}$	$\cdots$	$\sigma_{2,7}$	
1	$\cdots$	0	$\sigma_{3,0}$	$\cdots$	$\sigma_{3,7}$	
$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
0	$\cdots$	1	$\sigma_{252,0}$	$\cdots$	$\sigma_{252,7}$	
1	$\cdots$	1	$\sigma_{253,0}$	$\cdots$	$\sigma_{253,7}$	T_1
0	$\cdots$	1	$\sigma_{254,0}$	$\cdots$	$\sigma_{254,7}$	
1	$\cdots$	1	$\sigma_{255,0}$	$\cdots$	$\sigma_{255,7}$	



Note that selector ciphertext must be TRGSW, output ciphertext is TLWE

Detailed Algorithm

Algorithm 1. LUTMixedPacking

Input: Eight TRGSW ciphertexts $C_0, \cdots, C_7$
Input: Two TRLWE ciphertexts used for packing S-box d_0, d_1
Output: Eight TLWE ciphertexts $c_0, \cdots, c_7$

```
ACC ← CMUX(C_7, d_1, d_0)
for i = 0 to 6 do
    ACC ← CMUX(C_i, X^{-8·2^i (mod 2N)} · ACC, ACC)
end for
for i = 0 to 7 do
    c'_i ← SampleExtract(ACC, i)
    c_i ← KeySwitch(c'_i)
end for
return c_0, ··· , c_7
```

Circuit Bootstrapping(TLWE-to-TRGSW)

How？

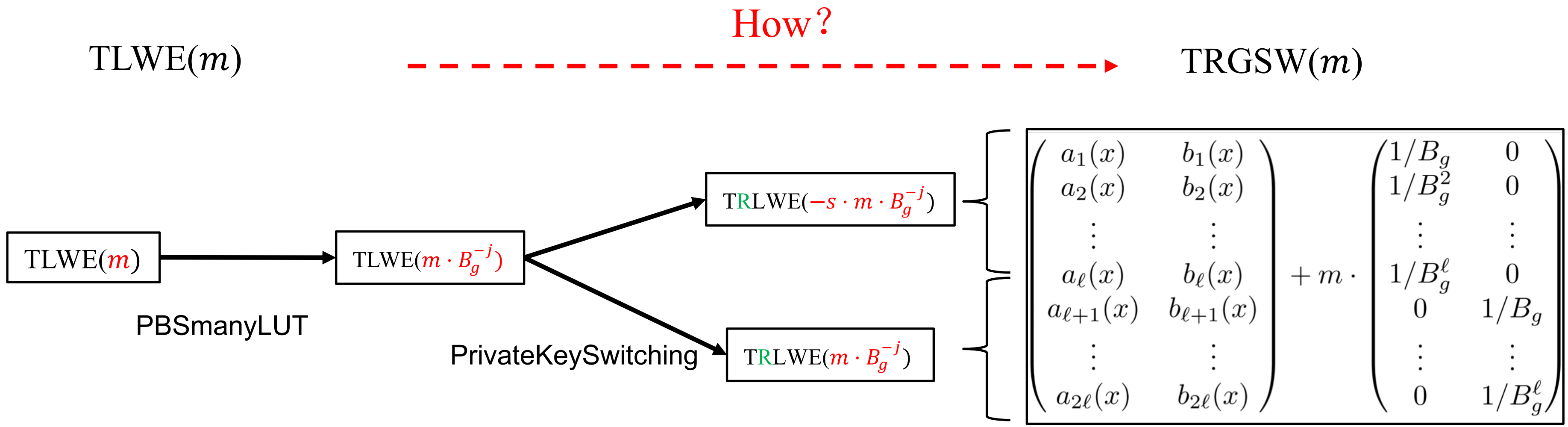
TLWE(m) - - - - - - - - - - → TRGSW(m)

TRLWE($-s \cdot m \cdot B_g^{-j}$)

TRLWE($m \cdot B_g^{-j}$)

$$\begin{pmatrix} a_1(x) & b_1(x) \\ a_2(x) & b_2(x) \\ \vdots & \vdots \\ a_\ell(x) & b_\ell(x) \\ a_{\ell+1}(x) & b_{\ell+1}(x) \\ \vdots & \vdots \\ a_{2\ell}(x) & b_{2\ell}(x) \end{pmatrix} + m \cdot \begin{pmatrix} 1/B_g & 0 \\ 1/B_g^2 & 0 \\ \vdots & \vdots \\ 1/B_g^\ell & 0 \\ 0 & 1/B_g \\ \vdots & \vdots \\ 0 & 1/B_g^\ell \end{pmatrix}$$

Circuit Bootstrapping(TLWE-to-TRGSW)

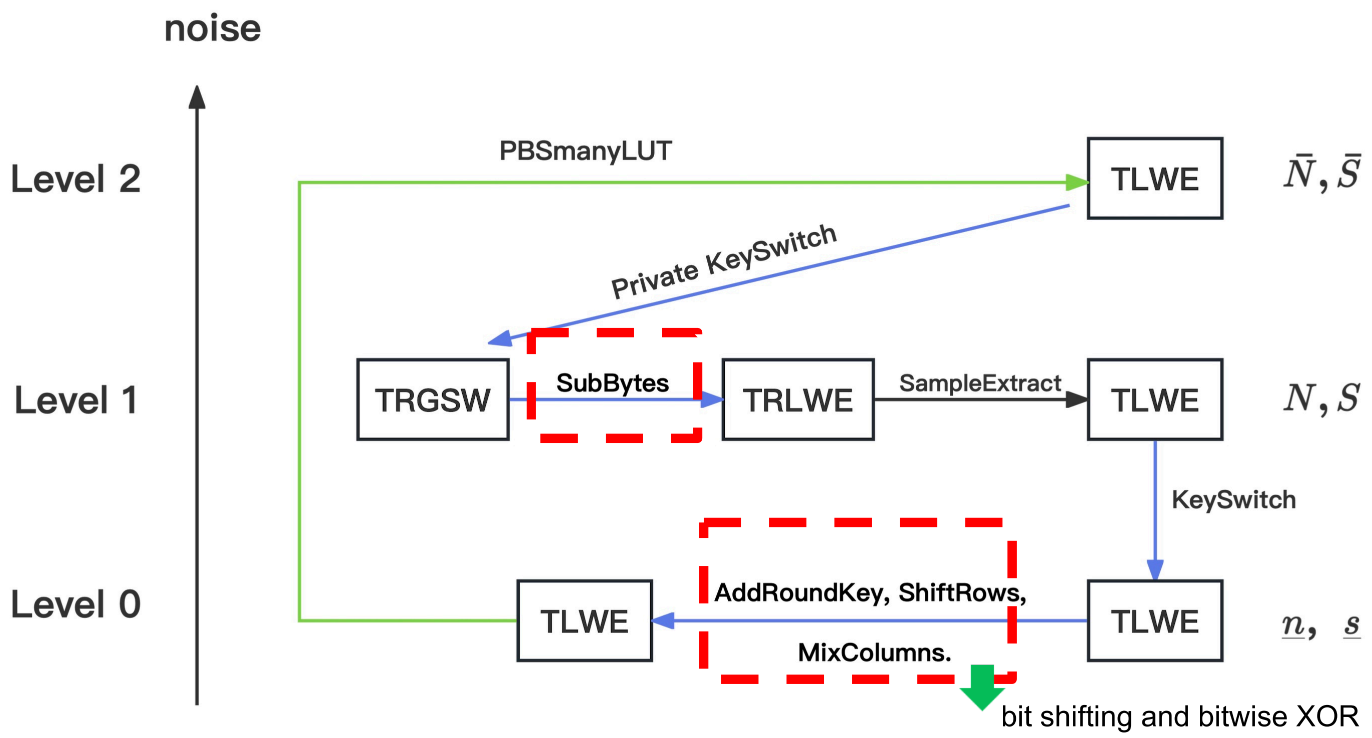


A new test polynomial that satisfies negacyclic property for PBSmanyLUT [CLOT21]:

$$P(X) = \sum_{i=0}^{\frac{N}{2^\rho \cdot 2}-1} \sum_{j=0}^{2^\rho - 1} (-1) \cdot \frac{1}{2\mathfrak{B}^j} X^{2^\rho \cdot i + j} + \sum_{i=\frac{N}{2^\rho \cdot 2}}^{\frac{N}{2^\rho}-1} \sum_{j=0}^{2^\rho - 1} \frac{1}{2\mathfrak{B}^j} X^{2^\rho \cdot i + j}$$

Fregata: Faster Homomorphic Evaluation of AES via TFHE

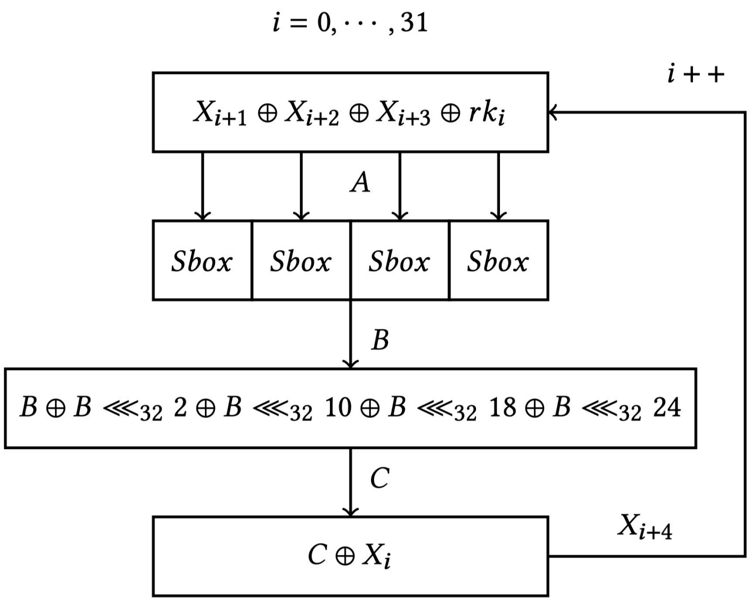
Message Encoding: $\{0,1\} \rightarrow \{0,\frac{1}{2}\}$ over the Torus



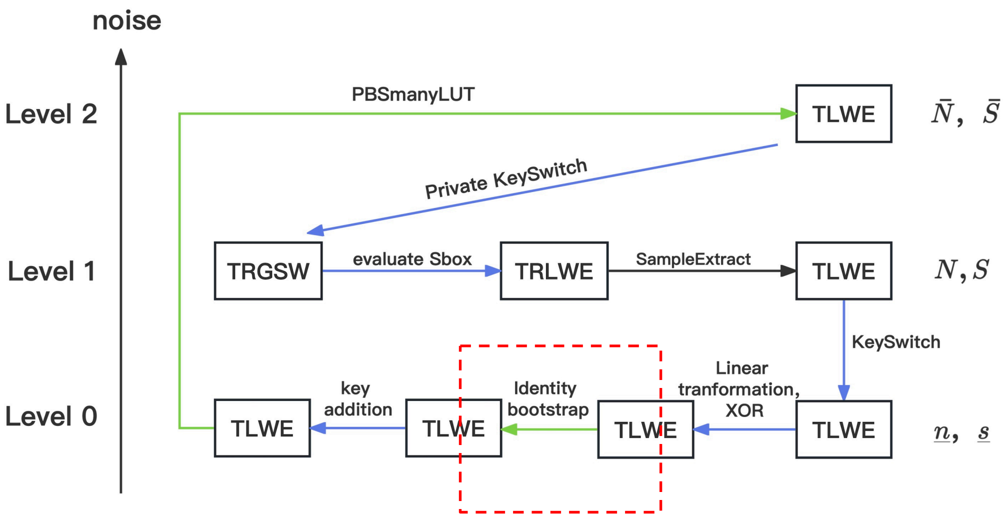
ShiftRows, AddRoundKey and MixColumns can be performed at Level 0, while SubBytes would be performed in Level 1

Scalability: Homomorphic Evaluation of SM4

SM4 is a Chinese block cipher standard used for protecting wireless networks and was released publicly in 2006. Now it has become the international standard of ISO/IEC.The structure of SM4 is similar to the AES algorithm, but it uses generalized Feistel structure. And its encryption computation requires up to 32 rounds.



SM4 cipher



Homomorphic Evaluation of SM4 via Fregata

Recent Improvement (FHE.org2024)

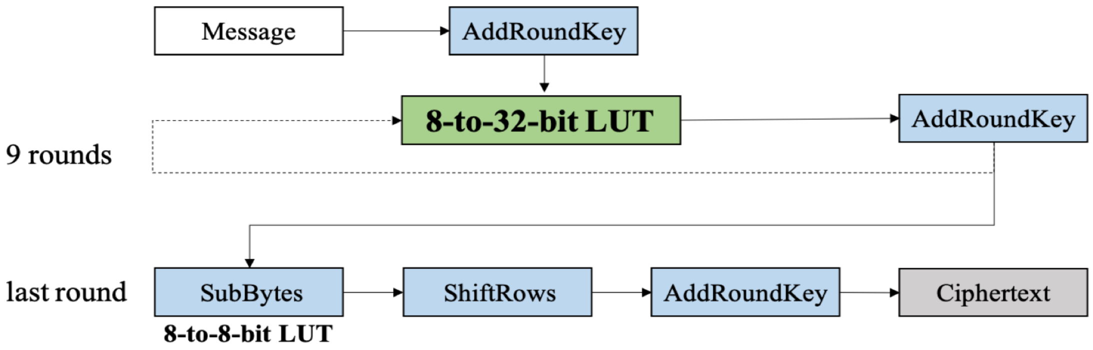
Implementation Methods of AES

(1)Using four basic functions

SubBytes, RowShifts, MixColumns and AddRoundKey

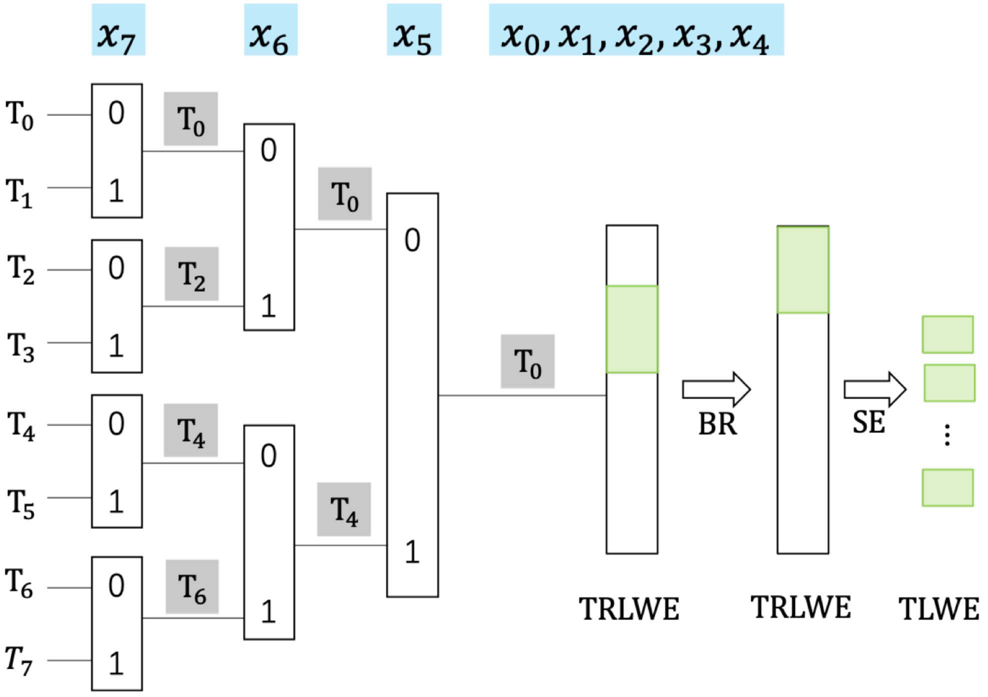
(2)Using LUT-based implementation

Merge SubBytes, RowShifts and MixColumns three functions into 8-to-32-bit LUT, as follows. We present faster evaluation of AES using this implementation based on leveled TFHE.



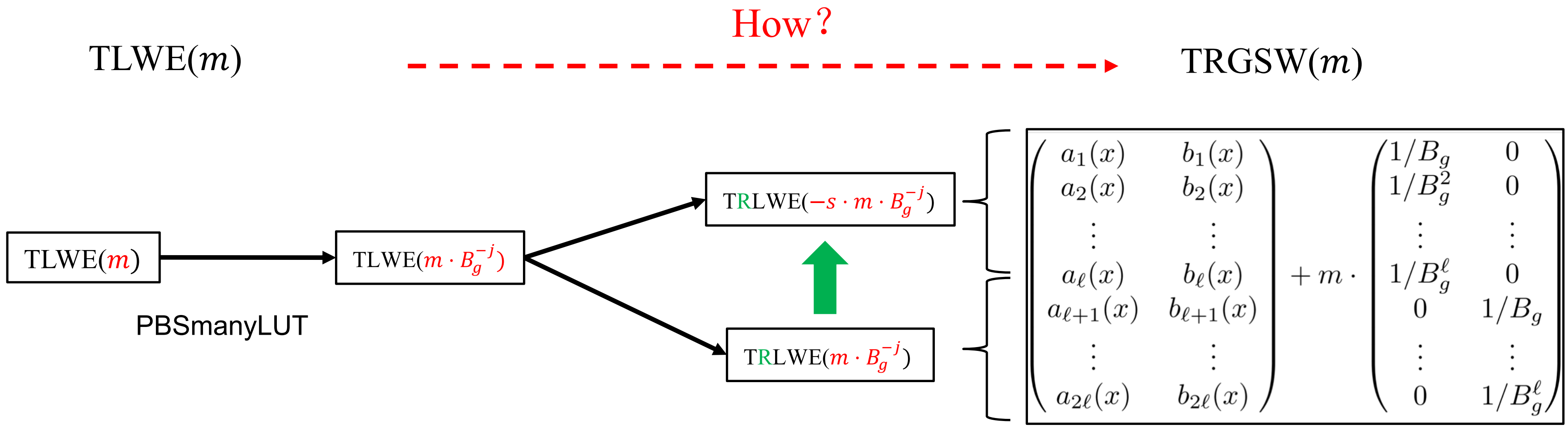
Efficient 8-to-32-bit lookup table

x_0	$\cdots$	x_7	f_0	$\cdots$	f_{31}
0	$\cdots$	0	$\sigma_{0,0}$	$\cdots$	$\sigma_{0,31}$
1	$\cdots$	0	$\sigma_{1,0}$	$\cdots$	$\sigma_{1,31}$
0	$\cdots$	0	$\sigma_{2,0}$	$\cdots$	$\sigma_{2,31}$
1	$\cdots$	0	$\sigma_{3,0}$	$\cdots$	$\sigma_{3,31}$
$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
0	$\cdots$	1	$\sigma_{252,0}$	$\cdots$	$\sigma_{252,31}$
1	$\cdots$	1	$\sigma_{253,0}$	$\cdots$	$\sigma_{253,31}$
0	$\cdots$	1	$\sigma_{254,0}$	$\cdots$	$\sigma_{254,31}$
1	$\cdots$	1	$\sigma_{255,0}$	$\cdots$	$\sigma_{255,31}$



Lookup table using CMUX and mixed packing

Efficient Circuit Bootstrapping(TLWE-to-TRGSW)



Accelerate the Second Step

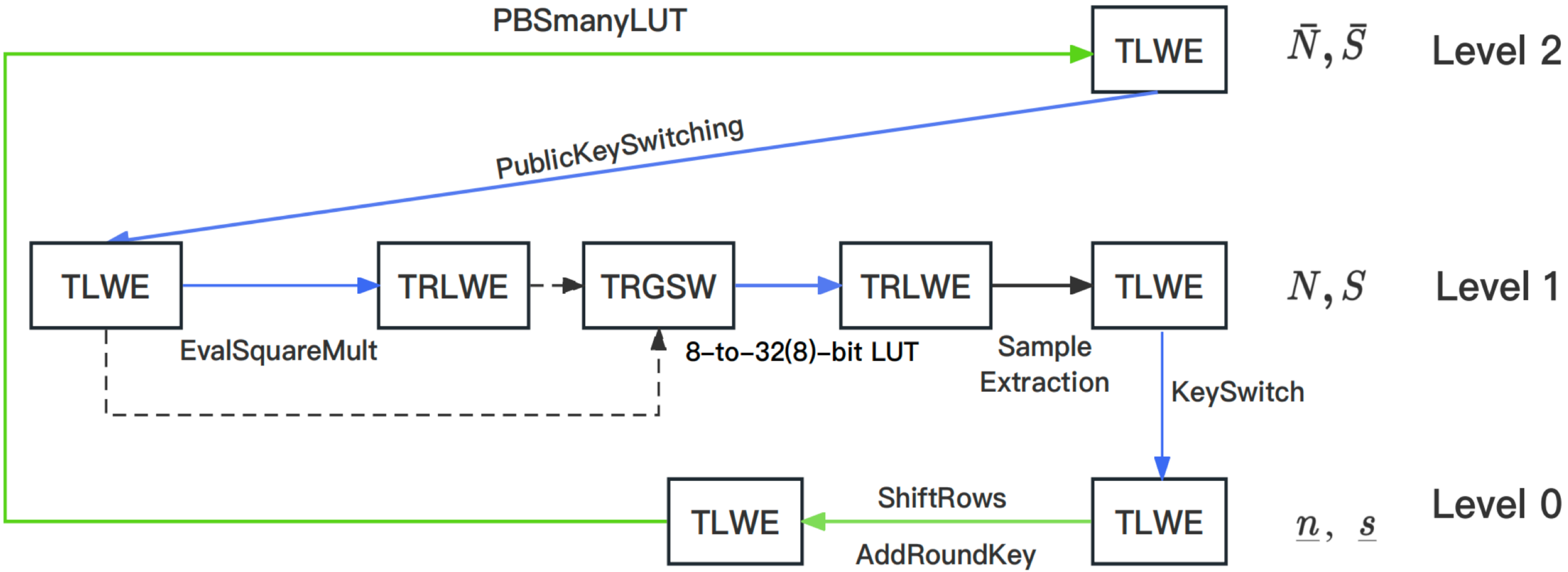
(1)The second ℓ rows of TRGSW can be constructed by **PublicKeySwitch:**

$$\mathrm{TLWE}(m) \rightarrow \mathrm{TRLWE}\left(\mathrm{m} \cdot \frac{1}{B^{-j}}\right), \mathrm{j} = 1, \cdots, \ell$$

(2)The first ℓ rows of TRGSW can be constructed by **EvalSquareMult** [KLD+23] **:**

$$\mathrm{TRLWE}\left(\mathrm{m} \cdot \frac{1}{B^{-j}}\right) \rightarrow \mathrm{TRLWE}\left(-\mathrm{s} \cdot \mathrm{m} \cdot \frac{1}{B^{-j}}\right), \mathrm{j} = 1, \cdots, \ell$$ (a KeySwitch)

New evaluation framework



Efficient AES evaluation framework based on leveled TFHE

Performance

Experimental environment

a single core of Intel(R) Core(TM) i5-11500 CPU @ 2.70GHz and 32 GB RAM, running the Ubuntu 20.04 operating system. We used a public available FHE library **TFHEpp**.

Implementation result

Scheme	Evaluation mode	Latency	Amortized
BGV	Leveled[GHS12]	4 mins	2 s
	Bootstrapping[GHS12]	18 mins	6 s
CKKS	Bootstrapping[ADE+23]	31mins	56.7 ms*
TFHE	Functional bootstrapping[SMK22]	4.2 mins	4.2 mins*
	Functional bootstrapping[TCBS23]	270 s	270 s
	Functional bootstrapping[BPR23]	211 s	211 s
	Ours(Leveled)	86s(46 s)	86s(46 s)

Table: Comparison of AES-128 evaluation latency based on different schemes

References

- [CGGI20] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène.TFHE: fast fully homomorphic encryption over the torus. J. Cryptol., 33(1): 34–91, 2020.
- [CLOT21] Ilaria Chillotti, Damien Ligier, Jean-Baptiste Orfila, and Samuel Tap. Improved programmable bootstrapping with larger precision and efficient arithmetic circuits for TFHE. In Advances in Cryptology-ASIACRYPT2021, Springer, 2021.
- [GHS12] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In Advances in Cryptology-CRYPTO2012, Springer, 2012.
- [CLT14] Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Scale-invariant fully homomorphic encryption over the integers. In Public-Key Cryptography - PKC 2014, pages 311–328. Springer, 2014.
- [GBA22] Antonio Guimarães, Edson Borin, and Diego F. Aranha. MOSFHET: optimized software for FHE over the torus. IACR Cryptol. ePrint Arch., page 515, 2022.
- [SMK22] Roy Stracovsky, Rasoul Akhavan Mahdavi, and Florian Kerschbaum. Faster evaluation of aes using tfhe. Poster Session, FHE. Org-2022, 2022.
- [TCBS23] Daphné Trama, Pierre-Emmanuel Clet, Aymen Boudguiga, and Renaud Sirdey. At last!A homomorphic AES evaluation in less than 30 seconds by means of TFHE.
- [KLD+23] Andrey Kim, Yongwoo Lee, Maxim Deryabin, Jieun Eom, and Rakyong Choi. LFHE: fully homomorphic encryption with bootstrapping key size less than a megabyte. IACR Cryptol. ePrint Arch., page 767, 2023.
- [BPR23] Nicolas Bon, David Pointcheval, and Matthieu Rivain. Optimized homomorphic evaluation of boolean functions. Cryptology ePrint Archive, Paper 2023/1589, 2023. https://eprint.iacr.org/2023/1589.
- [ADE+23] Ehud Aharoni, Nir Drucker, Gilad Ezov, Eyal Kushnir, Hayim Shaul, and Omri Soceanu. E2E near-standard and practical authenticated transciphering. IACR Cryptol. ePrint Arch., page1040, 2023.

Thank you !

GitHub home page:

https://github.com/WeiBenqiang/Fregata