Functional Bootstrapping for Packed Ciphertexts

Via Homomorphic LUT Evaluation

Seonhong Min, 30th May 2024
Seoul National University

Introduction

Fully Homomorphic Encryption

- **Fully Homomorphic Encryption**
  - Enables an unlimited number of computations over encrypted data.
- **Somewhat HE (SHE) can be constructed from (R)LWE**
  - Only supports a limited number of multiplications.
  - Not FHE.
- **Bootstrapping [Gen09]**
  - Homomorphic evaluation of decryption circuit.
  - The message remains the same, introduces a noise with fixed size.
  - The main bottleneck of homomorphic computation.

FV (Fan-Vercauteren) Scheme

- **Scheme description**
  - Base ring : $R = \mathbb{Z}[X]/\Phi_m(X)$
  - Secret key : $\mathsf{sk} \in R$, a ternary polynomial with small Hamming weight.
  - Message : $\mu(X) \in R_t = R/tR$ for plaintext modulus t.
  - Ciphertext : $(b, a) \in R_q^2 = (R/qR)^2$ for ciphertext modulus q.
    - Encrypt : $a \leftarrow \mathcal{U}(R_q), e \leftarrow \chi$, and set $b = -a \cdot \mathsf{sk} + \lfloor q/t \rceil \cdot \mu + e$.
    - Decrypt : $\lfloor t/q \cdot \left(b + a \cdot \mathsf{sk}\right) \rceil = \lfloor t/q \cdot (\lfloor q/t \rceil \cdot \mu + e) \rceil = \mu$.
    - Message in the MSB, noise in the LSB.

FV (Fan-Vercauteren) Scheme

- **SIMD arithmetic**
  - For a prime number $p \nmid m$, $R_p = \mathbb{Z}_p[X]/\Phi_m(X) \cong \prod_{i=1}^{k} \mathbb{Z}_p[X]/F_i(X)$
    - For d, the multiplicative order of p in group $\mathbb{Z}_m^\times$, $k = \phi(m)/d$.
    - Each $F_i(X)$ is a degree d (monic) irreducible polynomial.
  - We can perform SIMD arithmetic over $GF(p^d)^k$.
  - Usually, we encode only the constant term and use $\mathbb{Z}_p^k$ arithmetic.

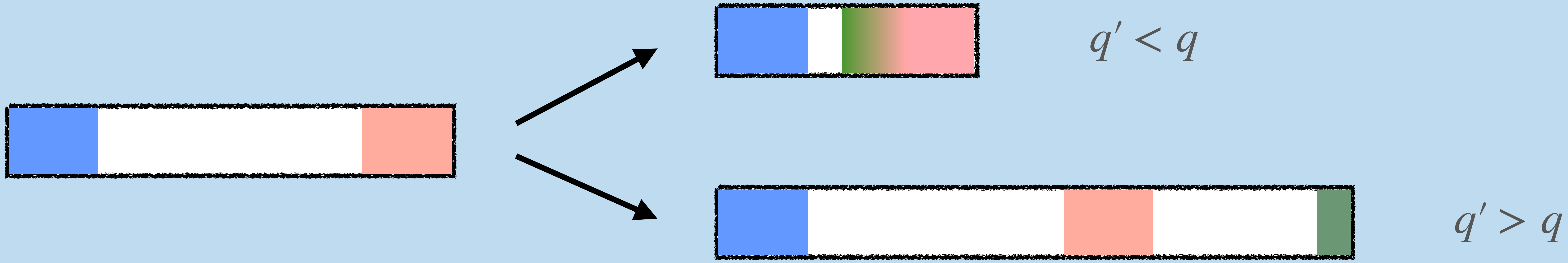
FV (Fan-Vercauteren) Scheme

- **SIMD arithmetic (2)**
  - Hensel's lifting lemma gives the relation $R_{p^s} \cong \prod_{i=1}^{k} \mathbb{Z}_{p^s}[X]/\tilde{F}_i(X)$.
  - We can use SIMD arithmetic over $\mathbb{Z}_{p^s}^k$.
- **Plaintext Change**
  - In FV context, $p \cdot \overrightarrow{m} \in \mathbb{Z}_{p^s}^k$ is equivalent to $m \in \mathbb{Z}_{p^{s-1}}^k$.

    ➡Just a simple change of plaintext modulus! (Change of interpretation…)
  - This operation is often referred as 'homomorphic division'.

FV (Fan-Vercauteren) Scheme

- **Scale-Invariant Scheme**
  - Since the message is stored in MSB, FV is invariant to (ciphertext) scaling.
  - Given an encryption $\mathsf{ct} = (c_0, c_1) \in R_q^2$ of message $\mu \in R_t$,
    - $(\lfloor q'/q \cdot c_0 \rceil, \lfloor q'/q \cdot c_1 \rceil) \in R_{q'}^2$ is still an encryption of μ,
    - As long as rounding error does not interfere the message part.



Bootstrapping of FV

Input : $\mathsf{ct} = (b, a) \in R_q^2$ **encrypting** $\mu(X) \in R_{p^s}$**.**

1. **ModSwitch (+ Dot Product, SubSum)**
   - Change the ciphertext modulus to p^r
     - i.e., generate $(b', a') = (\lfloor p^r/q \cdot b \rceil, \lfloor p^r/q \cdot a \rceil) \in R_{p^r}^2$
     - To make the decryption circuit as compact as possible.
   - Generate encryption of $[b' + a' \cdot \mathsf{sk}]_{p^r} = p^{r-s} \cdot \mu + e \in R_{p^r}$
     - Simply compute $(\lfloor q/p^r \rceil \cdot b', \lfloor q/p^r \rceil \cdot a') \in R_q^2$
   - Embed e into the ‘valid’ encoding space.
     - Note that e is totally random.
     - Therefore, the SIMD encoding of $\mathbb{Z}_{p^r}^k$ may not be valid.
     - Can be computed with automorphisms.

Bootstrapping of FV

2. **Coeffs2Slots**
   - Homomorphically move the coefficients of plaintext to the slots.
     - i.e., generate encryption of $p^{r-s} \cdot \overrightarrow{\mu} + \vec{e} \in \mathbb{Z}_{p^s}^k$, the coefficient vector of $p^{r-s} \cdot \mu(X) + e(X)$.
     - This can be performed with homomorphic matrix multiplication.
3. **DigitExtract**
   - Homomorphically remove the noise part e.
     - i.e., generate encryption of $\overrightarrow{\mu} \in \mathbb{Z}_{p^s}^k$.
     - Consists of a number of polynomial evaluations.
4. **Slots2Coeffs**
   - Homomorphically move the slots to the coefficients.
     - i.e., generate encryption of $\mu(X)$
     - Can be performed via a homomorphic matrix multiplication.

Bootstrapping of FV

	Functionality	Coefficients	Message
-	-	$\mu(X) \in R_{p^s}$	$\{m_i\}_{1\leq i\leq k} \in \mathbb{Z}_{p^s}^k$
ModSwitch	Switch the ciphertext modulus to p^r	$p^{r-s} \cdot \mu(X) + e(X) \in R_{p^r}$	?
Coeffs2Slots	Move the coefficients to slots	?	$\{p^{r-s} \cdot \mu_i + e_i\} \in \mathbb{Z}_{p^r}^k$
DigitExtract	Homomorphically remove the noise	?	$\{\mu_i\} \in \mathbb{Z}_{p^s}^k$
Slots2Coeffs	Move the slots to coefficients	$\mu(X) \in R_{p^s}$	$\{m_i\}_{1\leq i\leq k} \in \mathbb{Z}_{p^s}^k$

Digit Extraction

- **Given** $u_{r-1}u_{r-2}\ldots u_0 \in \mathbb{Z}_{p^r}$, **homomorphically compute** $u_{r-1}u_{r-2}\ldots u_{r-s} \in \mathbb{Z}_{p^s}$
  - There is no polynomial directly compute this.
  - We utilise homomorphic division to circumvent this problem.
  - There exists a series of 'Digit Extraction Polynomial' $\{G_i\}_{1\leq i}$.
    - $G_i(x) = [x]_p \pmod{p^i}$
    - i.e. Extracts the last digit of the given number.
  - Remove LSB iteratively, using digit extraction polynomials.

Digit Extraction

- **Input** : $u := u_{r-1}u_{r-2}\ldots u_0 \in \mathbb{Z}_{p^r}$
- **Output** : $u_{r-1}u_{r-2}\ldots u_{r-s} \in \mathbb{Z}_{p^r}$
  - $G_r(u) = 0\ldots 0u_0 \in \mathbb{Z}_{p^r}$.
  - $u - G_r(u) = u_{r-1}\ldots u_1 0 = p \cdot (u_{r-1}\ldots u_1)$.
  - $(u - G_r(u))/p = u_{r-1}\ldots u_1 \in \mathbb{Z}_{p^{r-1}}$

    ➡ Homomorphic division by p!
  - Repeat this procedure for $r - s$ times.
  - In practice, there exists a depth optimisation. (See [CH18], [GIKV22])

Our Work

Our Contribution

- **Homomorphic LUT evaluation from $\mathbb{Z}_{p^r}$ to $\mathbb{Z}_{p^s}$**
  - This is generally a hard task, since it may not be a polynomial function.
  - We devise a general evaluation method for arbitrary LUTs.
- **Functional bootstrapping for any RLWE encryptions.**
  - Similar to TFHE, it can bootstrap any RLWE ciphertext regardless the scheme.
  - In this work, we focus on FV and CKKS.

Functional Bootstrapping Pipeline

- **Usage of ‘slim mode’ bootstrapping**
  - In (normal) bootstrapping, digit extraction operates on coefficients.
  - Therefore, we use ‘slim mode’ ([HS18]), which operates on message.
    - Slots2Coeffs $\rightarrow$ModSwitch $\rightarrow$Coeffs2Slots $\rightarrow$DigitExtract
    - Adds the rounding noise to the message part instead of the coefficients.

Functional Bootstrapping Pipeline

	Functionality	FV	CKKS
Slots2Coeffs	Move the messages to coefficients	$m(X) \in R_t$	$\lfloor \Delta \cdot m(X) \rceil \in R$
ModSwitch	Switch the ciphertext modulus to p^r	$\left\lfloor \frac{p^r}{t} \right\rceil \cdot m(X) + e(X) \in R_{p^r}$	$\lfloor \Delta' \cdot m(X) \rceil \in R_{p^r}$
Coeffs2Slots	Move the coefficients to slots	$\left\{ \left\lfloor \frac{p^r}{t} \right\rceil \cdot m_i + e_i \right\}_{1 \le i \le k} \in \mathbb{Z}_{p^r}^k$	$\{\Delta' \cdot m_i\}_{1 \le i \le k} \in \mathbb{Z}_{p^r}^k$
EvalLUT	Evaluate LUT over the slots	$\{f(m_i)\}_{1 \le i \le k} \in \mathbb{Z}_{p^s}^k$	$\{f(m_i)\}_{1 \le i \le k} \in \mathbb{Z}_{p^s}^k$

Homomorphic LUT Evaluation ($\mathbb{Z}_{p^r}$ to $\mathbb{Z}_p$)

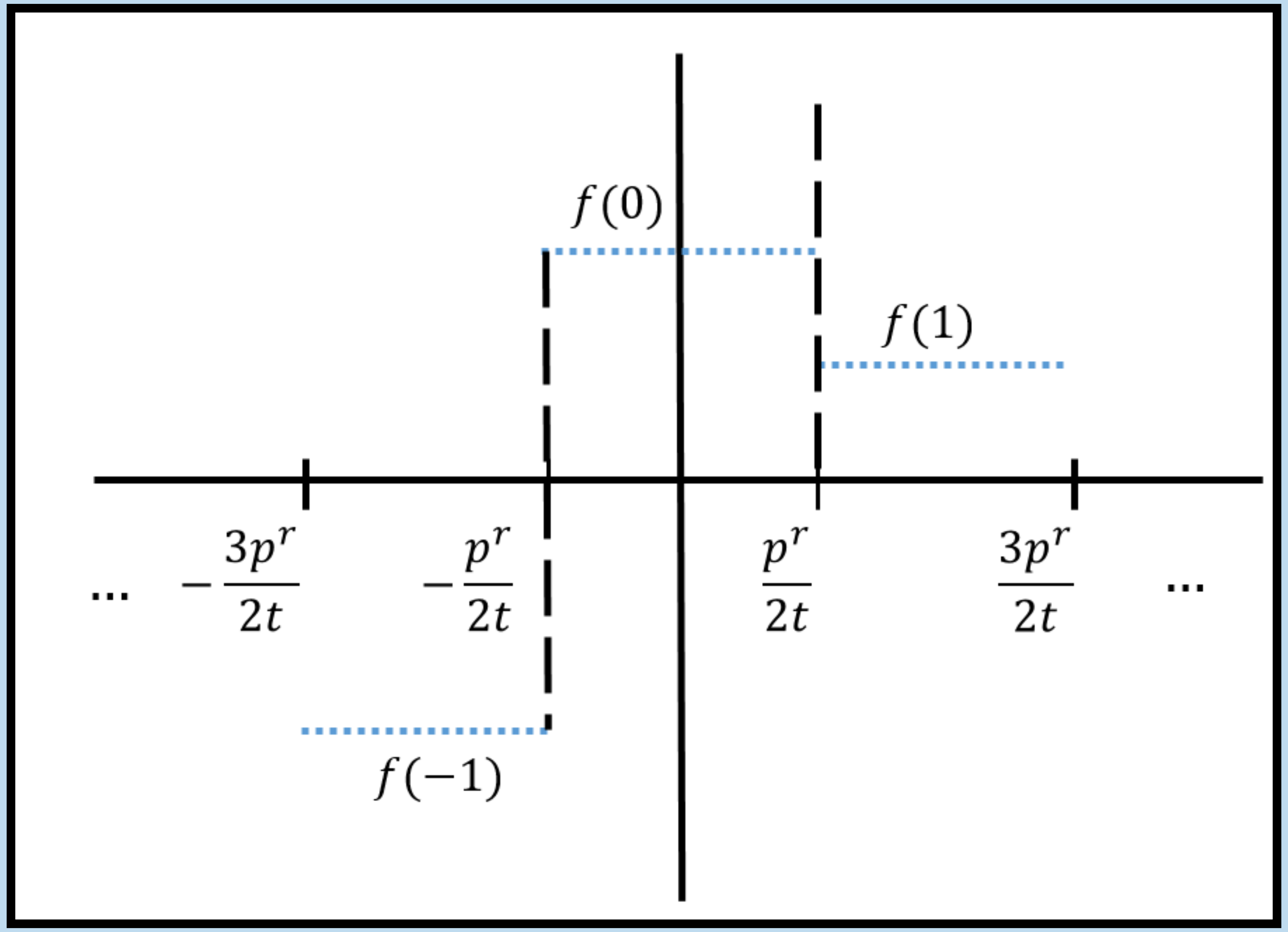
- **Given an LUT $F : \mathbb{Z}_{p^r} \to \mathbb{Z}_p$**
  - (Hopefully) there exists a polynomial p such that $p(x) = p^{r-1} \cdot F(x) \pmod{p^r}$.
  - Generally, there is no such polynomial p.
- **Our observation**
  - F can be written as a multivariate function of each digit of the input.
    - i.e., $F(u_{r-1} \dots u_0) = \tilde{F}(u_0, \dots, u_{r-1})$
  - Then, $\tilde{F}$ always has a polynomial representation over $\mathbb{Z}_p$.

Homomorphic LUT Evaluation ($\mathbb{Z}_{p^r}$ to $\mathbb{Z}_p$)

- **Our method**
  - Given LUT $F : \mathbb{Z}_{p^r} \to \mathbb{Z}_p$, find $\tilde{F} : \mathbb{Z}_p^r \to \mathbb{Z}_p$ such that $\tilde{F}(x_0, x_1, \ldots, x_{r-1}) = F(x_{r-1} \ldots x_0)$.
  - During **DigitExtract**, each digit is extracted.
    - More precisely, compute $[p^{r-i-1} \cdot u_r \ldots u_{i+1} u_i]_{p^{r-i}} = [u_i]_p$.
  - Then, evaluate $\tilde{F}$ using each digit.
- **Drawback**
  - (At most) $\tilde{F}$ is of degree $r(p-1)$, with p^r terms.
  - Computing such polynomial can be time-consuming.

Heaviside Function Evaluation

- **(Shifted) Heaviside Function**
  - The most basic form of step function
  - $\mathbf{1}_{x<B}(x) = \begin{cases} 0 & \text{if } x < B = b_{r-1}\ldots b_0 \\ 1 & \text{otherwise} \end{cases}$
- **Why Heaviside Function?**
  - LUT for FV-to-FV functional bootstrapping has a form of step function.
  - Heaviside function is the easiest form of the step function family.



Heaviside Function Evaluation

- **Recurrence Relation**
  - Define two Heaviside Functions over $\mathbb{Z}_{p^{r-1}}$
    - $\mathbf{1}_{x<B_1}(x) = \begin{cases} 0 & \text{if } x < B_1 := b_{r-1}\ldots(b_1+1) \\ 1 & \text{otherwise} \end{cases}$
    - $\mathbf{1}_{x<B_2}(x) = \begin{cases} 0 & \text{if } x < B_2 := b_{r-1}\ldots b_1 \\ 1 & \text{otherwise} \end{cases}$
  - Construct the following recurrence relation.
    - $\mathbf{1}_{x<B}(u_{r-1}\ldots u_1 u_0) = \mathbf{1}_{x<b_0}(u_0) \cdot \mathbf{1}_{x<B_1}(u_{r-1}\ldots u_1) + \mathbf{1}_{x\geq b_0}(u_1) \cdot \mathbf{1}_{x<B_2}(u_{r-1}\ldots u_1)$

Heaviside Function Evaluation

- **Recurrence Relation**
  - $\mathbf{1}_{x<B}(u_{r-1}\ldots u_1u_0) = \mathbf{1}_{x<b_0}(u_0)\cdot\mathbf{1}_{x<B_1}(u_{r-1}\ldots u_1) + \mathbf{1}_{x\geq b_0}(u_0)\cdot\mathbf{1}_{x<B_2}(u_{r-1}\ldots u_1)$
    - $\mathbf{1}_{x<b_0}$ and $\mathbf{1}_{x\geq b_0}$ has a univariate polynomial representation of u_0.
    - $\mathbf{1}_{x<B_1}, \mathbf{1}_{x<B_2}$ can be represented with two LUTs over $\mathbb{Z}_{p^{r-2}}$, using the relation.

      ➡ In fact, $\mathbf{1}_{x<B_1}$ and $\mathbf{1}_{x<B_2}$ can be represented with two identical LUTs.

      - $\mathbf{1}_{x<B_1}(u_{r-1}\ldots u_1) = \mathbf{1}_{x<(b_1+1)}\cdot\mathbf{1}_{x<B_3}(u_{r-1}\ldots u_2) + \mathbf{1}_{x\geq(b_1+1)}\cdot\mathbf{1}_{x<B_4}(u_{r-1}\ldots u_2)$
      - $\mathbf{1}_{x<B_2}(u_{r-1}\ldots u_1) = \mathbf{1}_{x<b_1}\cdot\mathbf{1}_{x<B_3}(u_{r-1}\ldots u_2) + \mathbf{1}_{x\geq b_1}\cdot\mathbf{1}_{x<B_4}(u_{r-1}\ldots u_2)$
    - It only requires $2+4+\ldots+2 = 4r-4$ univariate polynomial evaluations of degree $p-1$.

Heaviside Function Evaluation

- **Algorithm**
  - Input : Bound $B = b_{r-1} \ldots b_0 \in \mathbb{Z}_{p^r}$, (encrypted) messages $u_0, \ldots, u_{r-1} \in \mathbb{Z}_p$
  - Output : $\mathbf{1}_{x \geq b_{r-1} \ldots b_0}(u_{r-1} \ldots u_0)$

1. $\begin{array}{l} x_0 \leftarrow \mathbf{1}_{x \geq b_{r-1}+1}(u_{r-1}) \\ x_1 \leftarrow \mathbf{1}_{x \geq b_{r-1}}(u_{r-1}) \end{array}$

2. $\begin{array}{l} x_0 \leftarrow \mathbf{1}_{x < b_i+1}(u_i) \cdot x_0 + \mathbf{1}_{x \geq b_i+1}(u_i) \cdot x_1 \\ x_1 \leftarrow \mathbf{1}_{x < b_i}(u_i) \cdot x_0 + \mathbf{1}_{x \geq b_i}(u_i) \cdot x_1 \end{array}$ for $i = r - 2; i > 0; i -= 1$

3. Return $\mathbf{1}_{x < b_0}(u_0) \cdot x_0 + \mathbf{1}_{x \geq b_0}(u_0) \cdot x_1$

Step Function Evaluation

- **Step function is a linear combination of Heaviside functions.**
  - Given an LUT $F(x) = \begin{cases} \alpha_1 & \text{if } x < B_1 \\ \alpha_2 & \text{if } B_1 \le x < B_2 \\ \vdots & \\ \alpha_k & \text{if } B_{k-1} \le x \end{cases}$,

    We can write $F(x) = \alpha_1 + (\alpha_2 - \alpha_1) \cdot F_1(x) + \ldots + (\alpha_k - \alpha_{k-1}) \cdot F_{k-1}(x)$

    where $F_i(x) = \begin{cases} 0 & \text{if } x < B_i \\ 1 & \text{otherwise} \end{cases}$.
- **Remark : One can generalise the recurrence relation as long as $k \le p$.**

Homomorphic LUT Evaluation ($\mathbb{Z}_{p^r}$ to $\mathbb{Z}_{p^s}$)

- **Our method**
  - Given $F : \mathbb{Z}_{p^r} \to \mathbb{Z}_{p^s}$, define s LUTs $F_i : \mathbb{Z}_{p^r} \to \mathbb{Z}_p$ which outputs i-th digit of F.
    - i.e., $F_i(x) = \left[F(x)/p^i\right]_p \quad (0 \le i < s)$
  - Then, we have $F(x) = \sum_{i=0}^{s-1} \left[F_i(x)\right]_{p^r} \cdot p^i = \sum_{i=0}^{s-1} \left[F_i(x)\right]_{p^{r-i}}$.
  - Therefore, it remains to compute $\left[F_i(x)\right]_{p^{r-i}}$.

    ➡In other words, we need *homomorphic lifting*.

Homomorphic Lifting

- **Input : ct $= (b, a) \in R_q^2$, an encryption of $\overrightarrow{m} \in \mathbb{Z}_p^k$.**
  - Compute ct$' = (\lfloor 1/p^{i-1} \cdot b \rceil, \lfloor 1/p^{i-1} \cdot a \rceil) \in R_q^2$. (+SubSum)
    - ct$'$ is an encryption of $\overrightarrow{m} + p \cdot \vec{I}$ for some random $\vec{I} \in \mathbb{Z}_{p^{i-1}}^k$.
    - Evaluating G_i returns an encryption of $\overrightarrow{m} \in \mathbb{Z}_{p^i}$.
  - Why does it not need Coeffs2Slots/Slots2Coeffs as in bootstrapping?
    - This case, the message is stored in the LSB.
    - Conversely, the message is stored in the MSB when bootstrap.
    - When i is large enough (i.e., $||\vec{I}||_\infty \ll p^i$), depth consumption can be mitigated with Coeffs2Slots and Slots2Coeffs. (Use low-degree null polynomial from [MHWW24])

Comparison to TFHE-like schemes

	Ours	TFHE	Amortized TFHE (FHEW-like)	Amortized TFHE (FV/CKKS)	Amortized TFHE (Others)
Scheme	This work	[DM14], [CGGI16], [LMK+23]	[MS18], [GPvL23], [MKMS23]	[LW23], [LW24], [BCKS24]	[LW23], [OPP23]
Remaining Multiplicative Level	O	X	X	X	O
Large Plaintext Modulus	O	X	X	O	△
SIMD arithmetic	O	X	O	O	O

Asymptotic Bootstrapping Complexity

	Ephemeral Message Space	Time Complexity
Traditional Bootstrapping	$\Delta \cdot m + e$	$O(\log p^r + \log \|\|s\|\|_1)$
General Bootstrapping	$\Delta \cdot e_1 + e_2$	$O(\log(\|\|s\|\|_1))$
Functional Bootstrapping	$\Delta \cdot m + e$	$O(\log p^r + \log \|\|s\|\|_1)$

Classification of Existing Works

	BGV/FV	CKKS	FHEW-like
Traditional Bootstrapping	[HS14], [CH18], [GIKV22]	[CHK+18], [CCS19], [HK20], [LLL+21]…	-
General Bootstrapping	[KSS24], [MHWW24]	[KPK+22]	[ADE+21]
Functional Bootstrapping	Our work	[BCKS24]	[DM14], [CGGI16], [LMK+23]
Others	[LW23], [LW24]	-	[MS18], [LW23], [MKMS23], [OPP23]…

Thank you for listening!