High-Precision Bootstrapping for Approximate Homomorphic Encryption by Error Variance Minimization

Yongwoo Lee[1], Joon-Woo Lee[2], Young-Sik Kim[3], Yongjune Kim[4], Jong-Seon No[2], and HyungChul Kang[1]

[1,2]Samsung Advanced Institute of Technology

[2]Seoul National University

[3]Chosun University

[4]DGIST

Eurocrypt 2022, Jun 1 2022

Objectives of the Paper

1. Optimal polynomials for approximate HE in SNR aspect:
   Polynomial approximation for CKKS
   - High-precision CKKS bootstrapping (> 90-bit)
   - Depth equivalent to prior arts for up to 40-bit accuracy
2. Lazy-BSGS algorithm:
   Efficient homomorphic evaluation of polynomials
   - About x2 faster than original BSGS algorithm
   - BSGS algorithm-specific polynomial approximation

Table: Comparison of previous and the proposed method

	proposed	**prior arts**
approximation	direct	indirect
evaluation	single high-degree poly.	composition of polys
precision	> 90-bit	< 40-bit
depth	11-12	11-12
measure of noise	SNR	minimax

Objectives of the Paper

1. Optimal polynomials for approximate HE in SNR aspect:
   Polynomial approximation for CKKS
   - High-precision CKKS bootstrapping (> 90-bit)
   - Depth equivalent to prior arts for up to 40-bit accuracy
2. Lazy-BSGS algorithm:
   Efficient homomorphic evaluation of polynomials
   - About x2 faster than original BSGS algorithm
   - BSGS algorithm-specific polynomial approximation

Table: Comparison of previous and the proposed method

	proposed	**prior arts**
approximation	direct	indirect
evaluation	single high-degree poly.	composition of polys
precision	> 90-bit	< 40-bit
depth	11-12	11-12
measure of noise	SNR	minimax

Objectives of the Paper

1. Optimal polynomials for approximate HE in SNR aspect:
   Polynomial approximation for CKKS
   - High-precision CKKS bootstrapping (> 90-bit)
   - Depth equivalent to prior arts for up to 40-bit accuracy
2. Lazy-BSGS algorithm:
   Efficient homomorphic evaluation of polynomials
   - About x2 faster than original BSGS algorithm
   - BSGS algorithm-specific polynomial approximation

Table: Comparison of previous and the proposed method

	proposed	**prior arts**
approximation	direct	indirect
evaluation	single high-degree poly.	composition of polys
precision	> 90-bit	< 40-bit
depth	11-12	11-12
measure of noise	SNR	minimax

Outline

Outline

High-Precision Approximate HE

- High-precision HE is an **interesting** topic
  - Best-known precision is 40-bit [LLL+21, BMTH21]: less than double precision
  - **Bootstrapping is bottleneck** of CKKS accuracy
- Li-Micciancio **attack and accuracy**
  - Passive key recovery attack for CKKS is proposed by Li and Micciancio [LM21]
  - *Noise flooding technique* is a countermeasure:
    - Adding 30-40 bits of error after decryption
  - **High-precision** approximate HE makes everything simple

High-Precision Approximate HE

- High-precision HE is an **interesting** topic
  - Best-known precision is 40-bit [LLL+21, BMTH21]: less than double precision
  - **Bootstrapping is bottleneck** of CKKS accuracy
- Li-Micciancio **attack and accuracy**
  - Passive key recovery attack for CKKS is proposed by Li and Micciancio [LM21]
  - *Noise flooding technique* is a countermeasure:
    - Adding 30-40 bits of error after decryption
  - **High-precision** approximate HE makes everything simple

Approximate Homomorphic Encryption

- CKKS scheme is an **approximate** HE for efficient evaluation of real (or complex) numbers
- Approximate arithmetic: the message contains error

$$\langle \mathsf{ct}, \mathsf{sk} \rangle = \Delta m + e \pmod{Q}$$
$$\mathsf{ct} = (b, a), \mathsf{sk} = (1, s) \in \mathcal{R}_Q^2$$

- Polynomial approximation
  - **Addition and multiplications** are supported in CKKS scheme
  - Thus, **polynomials** can be evaluated
  - Non-arithmetic operations (comparison, modular reduction, ...) require polynomial approximation

Approximate Homomorphic Encryption

- CKKS scheme is an **approximate** HE for efficient evaluation of real (or complex) numbers
- Approximate arithmetic: the message contains error

$$\langle \mathsf{ct}, \mathsf{sk} \rangle = \Delta m + e \pmod{Q}$$
$$\mathsf{ct} = (b, a), \mathsf{sk} = (1, s) \in \mathcal{R}_Q^2$$

- Polynomial approximation
  - **Addition and multiplications** are supported in CKKS scheme
  - Thus, **polynomials** can be evaluated
  - Non-arithmetic operations (comparison, modular reduction, ...) require polynomial approximation

Approximate Homomorphic Encryption

- CKKS scheme is an **approximate** HE for efficient evaluation of real (or complex) numbers
- Approximate arithmetic: the message contains error

$$\langle \mathsf{ct}, \mathsf{sk} \rangle = \Delta m + e \pmod{Q}$$
$$\mathsf{ct} = (b, a), \mathsf{sk} = (1, s) \in \mathcal{R}_Q^2$$

- Polynomial approximation
  - **Addition and multiplications** are supported in CKKS scheme
  - Thus, **polynomials** can be evaluated
  - Non-arithmetic operations (comparison, modular reduction, ...) require polynomial approximation

Approximate Homomorphic Encryption

- CKKS scheme is an **approximate** HE for efficient evaluation of real (or complex) numbers
- Approximate arithmetic: the message contains error

$$\langle \mathsf{ct}, \mathsf{sk} \rangle = \Delta m + e \pmod{Q}$$
$$\mathsf{ct} = (b, a), \mathsf{sk} = (1, s) \in \mathcal{R}_Q^2$$

- Polynomial approximation
  - **Addition and multiplications** are supported in CKKS scheme
  - Thus, **polynomials** can be evaluated
  - Non-arithmetic operations (comparison, modular reduction, ...) require polynomial approximation

Ciphertext-Ciphertext Multiplications and Rescaling

- Ciphertext-ciphertext multiplication and relinearization
  - Ciphertext-ciphertext multiplications are expensive due to **relinearization**
  - **Relinearization:** key switching from $(1, s, s^2)$ to $(1, s)$
  - Key switching requires several NTTs
- Rescaling
  - CKKS plaintext is scaled by scaling factor
  - Rescaling reduces *message scale* and ciphertext *modulus*

$$Enc(\Delta' \cdot m) \pmod{Q} \rightarrow Enc(\frac{\Delta'}{\Delta} \cdot m) \pmod{\frac{Q}{\Delta}}$$

  - Rescaling is done after multiplications
  - Rescaling introduces rounding error

Ciphertext-Ciphertext Multiplications and Rescaling

- Ciphertext-ciphertext multiplication and relinearization
  - Ciphertext-ciphertext multiplications are expensive due to **relinearization**
  - **Relinearization:** key switching from $(1, s, s^2)$ to $(1, s)$
  - Key switching requires several NTTs
- Rescaling
  - CKKS plaintext is scaled by scaling factor
  - Rescaling reduces *message scale* and ciphertext *modulus*

$$Enc(\Delta' \cdot m) \pmod{Q} \rightarrow Enc(\frac{\Delta'}{\Delta} \cdot m) \pmod{\frac{Q}{\Delta}}$$

  - Rescaling is done after multiplications
  - Rescaling introduces rounding error

Depth, Level, and Bootstrapping

- Depth
  - Depth of a circuit: the maximal length of a path from an input to output gate
  - We care about **multiplicative depth**
  - E.g., degree d **polynomial** has depth $\log d$
- Level
  - Level of a ciphertext: **maximum depth** that the ciphertext can perform
- Bootstrapping
  - Homomorphic evaluation of decryption circuit
  - Refresh level of ciphertext
  - **Less depth** for bootstrapping $\rightarrow$ **more levels** after a bootstrapping

Depth, Level, and Bootstrapping

- Depth
  - Depth of a circuit: the maximal length of a path from an input to output gate
  - We care about **multiplicative depth**
  - E.g., degree d **polynomial** has depth $\log d$
- Level
  - Level of a ciphertext: **maximum depth** that the ciphertext can perform
- Bootstrapping
  - Homomorphic evaluation of decryption circuit
  - Refresh level of ciphertext
  - **Less depth** for bootstrapping $\rightarrow$ **more levels** after a bootstrapping

Depth, Level, and Bootstrapping

- Depth
  - Depth of a circuit: the maximal length of a path from an input to output gate
  - We care about **multiplicative depth**
  - E.g., degree d **polynomial** has depth $\log d$
- Level
  - Level of a ciphertext: **maximum depth** that the ciphertext can perform
- Bootstrapping
  - Homomorphic evaluation of decryption circuit
  - Refresh level of ciphertext
  - **Less depth** for bootstrapping $\rightarrow$ **more levels** after a bootstrapping

Depth, Level, and Bootstrapping

- Depth
  - Depth of a circuit: the maximal length of a path from an input to output gate
  - We care about **multiplicative depth**
  - E.g., degree d **polynomial** has depth $\log d$
- Level
  - Level of a ciphertext: **maximum depth** that the ciphertext can perform
- Bootstrapping
  - Homomorphic evaluation of decryption circuit
  - Refresh level of ciphertext
  - **Less depth** for bootstrapping $\rightarrow$ **more levels** after a bootstrapping

CKKS Bootstrapping

- ModRaise: change coefficient modulus from q to Q $(Q \gg q)$

$$\begin{aligned}\langle \mathsf{ct}, \mathsf{sk} \rangle &\approx \Delta m \pmod{q} \\ &\approx \Delta m + qI \\ &\approx \Delta m + qI \pmod{Q}\end{aligned}$$

- CoeffToSlot: homomorphic encoding - linear transformation
- EvalMod: approximate polynomial for modular reduction by q

$$f_{\mathsf{mod}}(\Delta m + qI) \approx \Delta m$$

- SlotToCoeff: homomorphic decoding - linear transformation

$$m(\zeta_j) = \sum_{i=0}^{N-1} m_i \zeta_j^i$$

CKKS Bootstrapping

- ModRaise: change coefficient modulus from q to Q $(Q \gg q)$

$$\begin{aligned}\langle \mathsf{ct}, \mathsf{sk} \rangle &\approx \Delta m \pmod{q} \\ &\approx \Delta m + qI \\ &\approx \Delta m + qI \pmod{Q}\end{aligned}$$

- CoeffToSlot: homomorphic encoding - linear transformation
- EvalMod: approximate polynomial for modular reduction by q

$$f_{\mathsf{mod}}(\Delta m + qI) \approx \Delta m$$

- SlotToCoeff: homomorphic decoding - linear transformation

$$m(\zeta_j) = \sum_{i=0}^{N-1} m_i \zeta_j^i$$

CKKS Bootstrapping

- ModRaise: change coefficient modulus from q to Q $(Q \gg q)$

$$\begin{aligned}\langle \mathsf{ct}, \mathsf{sk} \rangle &\approx \Delta m \pmod{q} \\ &\approx \Delta m + qI \\ &\approx \Delta m + qI \pmod{Q}\end{aligned}$$

- CoeffToSlot: homomorphic encoding - linear transformation
- EvalMod: approximate polynomial for modular reduction by q

$$f_{\mathsf{mod}}(\Delta m + qI) \approx \Delta m$$

- SlotToCoeff: homomorphic decoding - linear transformation

$$m(\zeta_j) = \sum_{i=0}^{N-1} m_i \zeta_j^i$$

CKKS Bootstrapping

- ModRaise: change coefficient modulus from q to Q $(Q \gg q)$

$$\begin{aligned}\langle \mathsf{ct}, \mathsf{sk} \rangle &\approx \Delta m \pmod{q} \\ &\approx \Delta m + qI \\ &\approx \Delta m + qI \pmod{Q}\end{aligned}$$

- CoeffToSlot: homomorphic encoding - linear transformation
- EvalMod: approximate polynomial for modular reduction by q

$$f_{\mathsf{mod}}(\Delta m + qI) \approx \Delta m$$

- SlotToCoeff: homomorphic decoding - linear transformation

$$m(\zeta_j) = \sum_{i=0}^{N-1} m_i \zeta_j^i$$

SNR Perspective of the CKKS Scheme

- The Signal-to-Noise Ratio (SNR) is a widely used **measure of signal quality**
- Defined as the ratio of the signal power to the noise power as

$$\mathrm{SNR} = \frac{P_S}{P_N} = \frac{E[S^2]}{E[N^2]}$$

- To increase the power of message, the larger scaling factor can be multiplied to the message $\rightarrow$ less levels
- So, we will minimize **noise power**, i.e., error variance

SNR Perspective of the CKKS Scheme

- The Signal-to-Noise Ratio (SNR) is a widely used **measure of signal quality**
- Defined as the ratio of the signal power to the noise power as

$$\mathrm{SNR} = \frac{P_S}{P_N} = \frac{E[S^2]}{E[N^2]}$$

- To increase the power of message, the larger scaling factor can be multiplied to the message $\rightarrow$ less levels
- So, we will minimize **noise power**, i.e., error variance

SNR Perspective of the CKKS Scheme

- The Signal-to-Noise Ratio (SNR) is a widely used **measure of signal quality**
- Defined as the ratio of the signal power to the noise power as

$$\mathrm{SNR} = \frac{P_S}{P_N} = \frac{E[S^2]}{E[N^2]}$$

- To increase the power of message, the larger scaling factor can be multiplied to the message $\rightarrow$ less levels
- So, we will minimize **noise power**, i.e., error variance

SNR Perspective of the CKKS Scheme

- The Signal-to-Noise Ratio (SNR) is a widely used **measure of signal quality**
- Defined as the ratio of the signal power to the noise power as

$$\mathrm{SNR} = \frac{P_S}{P_N} = \frac{E[S^2]}{E[N^2]}$$

- To increase the power of message, the larger scaling factor can be multiplied to the message $\rightarrow$ less levels
- So, we will minimize **noise power**, i.e., error variance

Outline

What Should We Consider?

- Polynomial basis is noisy
  - **Basis error** $\approx$ rounding error
  - Basis error is amplified by **large coefficients**
- Depth is valuable
  - **Penalty** for composition of small-degree polynomials
- Final error is **not approximation error** (in bootstrapping)
  - SlotToCoeff is done after polynomial evaluation
  - **Linear combination** of independent errors
  - We need *measure of error* considering SlotToCoeff

What Should We Consider?

- Polynomial basis is noisy
  - **Basis error** $\approx$ rounding error
  - Basis error is amplified by **large coefficients**
- Depth is valuable
  - **Penalty** for composition of small-degree polynomials
- Final error is **not approximation error** (in bootstrapping)
  - SlotToCoeff is done after polynomial evaluation
  - **Linear combination** of independent errors
  - We need *measure of error* considering SlotToCoeff

What Should We Consider?

- Polynomial basis is noisy
  - **Basis error** $\approx$ rounding error
  - Basis error is amplified by **large coefficients**
- Depth is valuable
  - **Penalty** for composition of small-degree polynomials
- Final error is **not approximation error** (in bootstrapping)
  - SlotToCoeff is done after polynomial evaluation
  - **Linear combination** of independent errors
  - We need *measure of error* considering SlotToCoeff

What Should We Consider?

- Polynomial basis is noisy
  - **Basis error** $\approx$ rounding error
  - Basis error is amplified by **large coefficients**
- Depth is valuable
  - **Penalty** for composition of small-degree polynomials
- Final error is **not approximation error** (in bootstrapping)
  - SlotToCoeff is done after polynomial evaluation
  - **Linear combination** of independent errors
  - We need *measure of error* considering SlotToCoeff

What Should We Consider?

- Polynomial basis is noisy
  - **Basis error** $\approx$ rounding error
  - Basis error is amplified by **large coefficients**
- Depth is valuable
  - **Penalty** for composition of small-degree polynomials
- Final error is **not approximation error** (in bootstrapping)
  - SlotToCoeff is done after polynomial evaluation
  - **Linear combination** of independent errors
  - We need *measure of error* considering SlotToCoeff

What Should We Consider?

- Polynomial basis is noisy
  - **Basis error** $\approx$ rounding error
  - Basis error is amplified by **large coefficients**
- Depth is valuable
  - **Penalty** for composition of small-degree polynomials
- Final error is **not approximation error** (in bootstrapping)
  - SlotToCoeff is done after polynomial evaluation
  - **Linear combination** of independent errors
  - We need *measure of error* considering SlotToCoeff

What Should We Consider?

- Polynomial basis is noisy
  - **Basis error** $\approx$ rounding error
  - Basis error is amplified by **large coefficients**
- Depth is valuable
  - **Penalty** for composition of small-degree polynomials
- Final error is **not approximation error** (in bootstrapping)
  - SlotToCoeff is done after polynomial evaluation
  - **Linear combination** of independent errors
  - We need *measure of error* considering SlotToCoeff

Noisy Polynomial Basis and Approximation for CKKS

- In the polynomial approximation for the CKKS scheme, **the magnitude of coefficients** should be small as well as the approx. error

Approximate arithmetic and noisy polynomial basis

- $f(x) = \sum c_i \phi_i(x)$: approximate polynomial of f_{mod}
- In the CKKS scheme, there exists error in polynomial basis, and thus we have

$$\sum c_i(\phi_i(x) + e_{\mathsf{basis},i}) = f(x) + \sum c_i \cdot e_{\mathsf{basis},i}.$$

- **Actual error:** $(f(x) - f_{\mathsf{mod}}(x)) + \sum c_i \cdot e_{\mathsf{basis},i}$
- As $e_{\mathsf{basis},i}$ is small, it is acceptable in general
- When $f(x) \ll |\boldsymbol{c}|$, such as polynomial for bootstrapping, the error is dominant

Error Variance-Minimizing Approximate Polynomial

- Magnitude of its *coefficients should be small*
- Using the generalized least square method, the optimal coefficient vector $\boldsymbol{c}^*$ of the approximate polynomial is obtained as

$$\boldsymbol{c}^* = \mathrm{argmin}_{\boldsymbol{c}} \left(Var[e_{\mathsf{aprx}}] + \sum w_i \boldsymbol{c}_i^2 \right),$$

  where $e_{\mathsf{aprx}} = f_{\mathsf{mod}} - f$
- Constants w_i are determined by the basis error

Input of Modulus Reduction

- Distribution of $\Delta m + q \cdot I$ is **not uniform!**
- **Reduce the error where the probability is high** $\rightarrow$ Variance
- I follows the Irwin-Hall distribution

Input distribution of f_{mod}

Figure: Input distribution of $f_{\mathsf{mod}}(\cdot)$: I follows Irwin-Hall, and distribution of m is unknown

Input of Modulus Reduction

- Distribution of $\Delta m + q \cdot I$ is **not uniform!**
- **Reduce the error where the probability is high** $\rightarrow$ Variance
- I follows the Irwin-Hall distribution

Input distribution of f_{mod}

Figure: Input distribution of $f_{\mathsf{mod}}(\cdot)$: I follows Irwin-Hall, and distribution of m is unknown

Input of Modulus Reduction

- Distribution of $\Delta m + q \cdot I$ is **not uniform!**
- **Reduce the error where the probability is high** $\rightarrow$ Variance
- I follows the Irwin-Hall distribution

Input distribution of f_{mod}

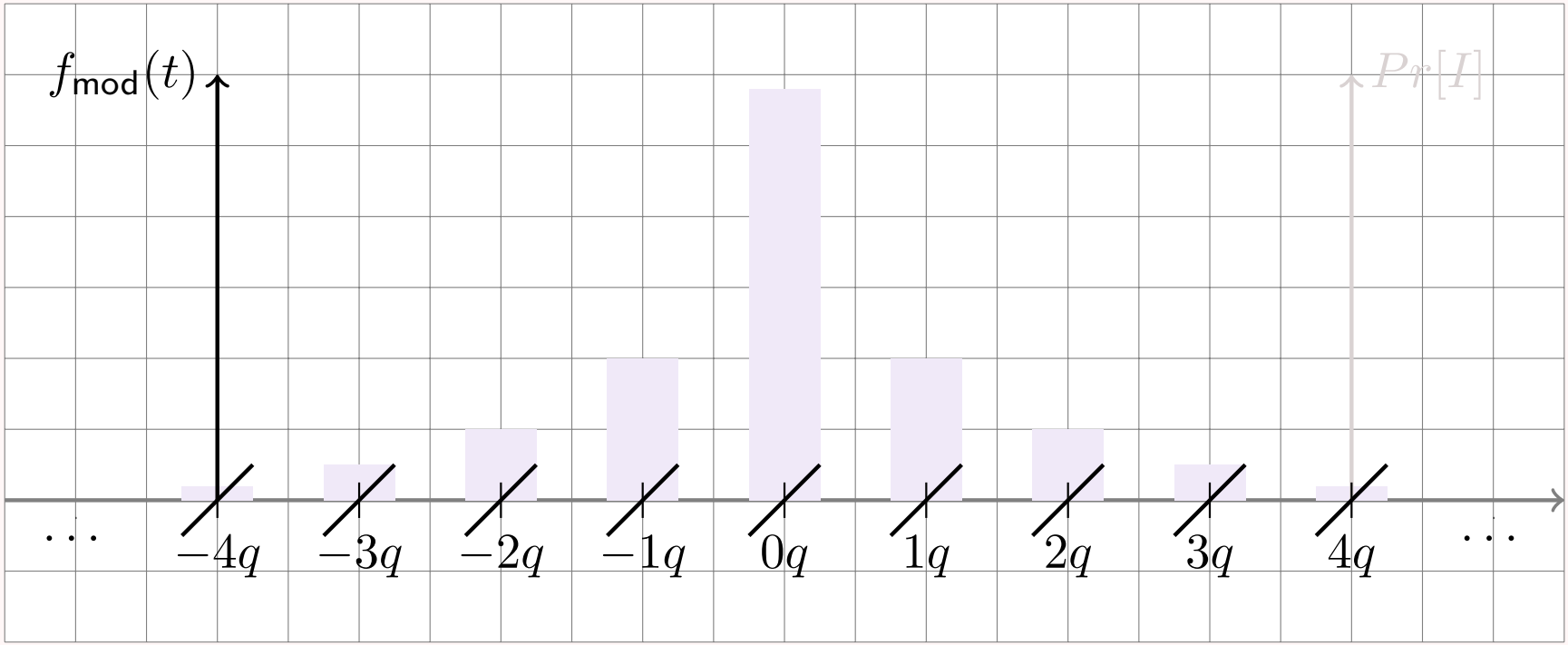


Figure: Input distribution of $f_{\mathsf{mod}}(\cdot)$: I follows Irwin-Hall, and distribution of m is unknown

Input of Modulus Reduction

- Distribution of $\Delta m + q \cdot I$ is **not uniform!**
- **Reduce the error where the probability is high** $\rightarrow$ Variance
- I follows the Irwin-Hall distribution

Input distribution of f_{mod}

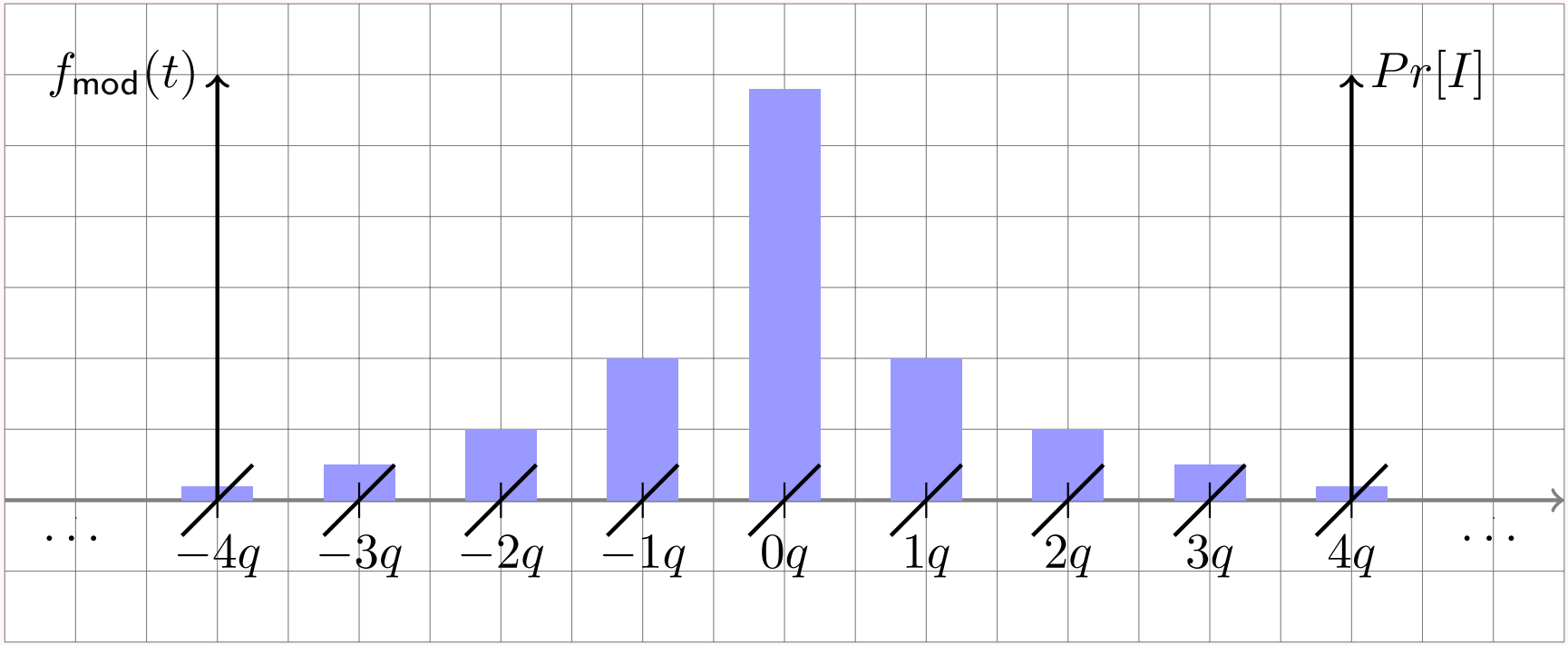


Figure: Input distribution of $f_{\mathsf{mod}}(\cdot)$: I follows Irwin-Hall, and distribution of m is unknown

Error Optimality of Proposed Method

- SlotToCoeff is the homomorphic decoding: $m(X)$ is given as $(m(\zeta_0), m(\zeta_1), \ldots, m(\zeta_{N/2-1}))$
- The error in the j-th slot after SlotToCoeff is

$$e_{\mathsf{boot},j}(\zeta_j) = \sum_{i=0}^{N-1} e_{\mathsf{mod},i} \cdot \zeta_j^i,$$

which is the sum of N independent random variables, where $|\zeta_j| = 1$

Error Optimality of Proposed Method (Contd.)

[Proposed] Variance-minimizing approximation

$$Var[e_{\mathsf{boot},j} = \sum e_{\mathsf{mod},i} \cdot \zeta_j^i] = Var[e_{\mathsf{mod},0} \cdot \zeta_j^0] + \cdots + Var[e_{\mathsf{mod},N-1} \cdot \zeta_j^{(N-1)}]$$

[Previous] Minimax approximation and bootstrapping error

$$\max\left(\left|\sum_{i=0}^{N-1} e_{\mathsf{aprx}}(t_i) \cdot \zeta_j^i\right|\right) \leq \left\|\zeta_j^0 \cdot e_{\mathsf{aprx}}\right\|_{\mathsf{sup}} + \cdots + \left\|\zeta_j^{N-1} \cdot e_{\mathsf{aprx}}\right\|_{\mathsf{sup}}$$

Depth Optimality of Direct Approximation

- Direct approximation finds an polynomial among all elements of P_{deg}
- Approximation of trigonometric function [LLL+21, BMTH21], the search space of the approximation algorithm is much more limited
  - For example, double angle formula twice and approximate polynomial for cosine and arcsine of degree deg_1 and deg_2, respectively
  - Then the search space is

$$\left\{ f_2 \circ g \circ f_1 | f_1 \in P_{\mathsf{deg}_1}, f_2 \in P_{\mathsf{deg}_2}, \text{ and } g(x) = (x^2 - 1)^2 - 1 \right\}$$

- **Direct** approximation **has less depth** than indirect approximation

$$\lceil \log(\mathsf{deg}_1 + 1) \rceil + 2 + \lceil \log(\mathsf{deg}_2 + 1) \rceil \geq \lceil \log(4\mathsf{deg}_1\mathsf{deg}_2 + 1) \rceil$$

Depth Optimality of Direct Approximation

- Direct approximation finds an polynomial among all elements of P_{deg}
- Approximation of trigonometric function [LLL+21, BMTH21], the search space of the approximation algorithm is much more limited
  - For example, double angle formula twice and approximate polynomial for cosine and arcsine of degree deg_1 and deg_2, respectively
  - Then the search space is

$$\left\{ f_2 \circ g \circ f_1 | f_1 \in P_{\mathsf{deg}_1}, f_2 \in P_{\mathsf{deg}_2}, \text{ and } g(x) = (x^2 - 1)^2 - 1 \right\}$$

- **Direct** approximation **has less depth** than indirect approximation

$$\lceil \log(\mathsf{deg}_1 + 1) \rceil + 2 + \lceil \log(\mathsf{deg}_2 + 1) \rceil \geq \lceil \log(4\mathsf{deg}_1\mathsf{deg}_2 + 1) \rceil$$

Depth Optimality of Direct Approximation

- Direct approximation finds an polynomial among all elements of P_{deg}
- Approximation of trigonometric function [LLL+21, BMTH21], the search space of the approximation algorithm is much more limited
  - For example, double angle formula twice and approximate polynomial for cosine and arcsine of degree deg_1 and deg_2, respectively
  - Then the search space is

$$\left\{ f_2 \circ g \circ f_1 | f_1 \in P_{\mathsf{deg}_1}, f_2 \in P_{\mathsf{deg}_2}, \text{ and } g(x) = (x^2 - 1)^2 - 1 \right\}$$

- **Direct** approximation **has less depth** than indirect approximation

$$\lceil \log(\mathsf{deg}_1 + 1) \rceil + 2 + \lceil \log(\mathsf{deg}_2 + 1) \rceil \geq \lceil \log(4\mathsf{deg}_1\mathsf{deg}_2 + 1) \rceil$$

Simple Analytic Solution

$$\arg\min_{\boldsymbol{c}} \left(Var[e_{\mathsf{aprx}}] + \sum w_i \boldsymbol{c}_i^2 \right)$$

$$\nabla_{\boldsymbol{c}} \left(Var[e_{\mathsf{aprx}}] + \sum w_i \boldsymbol{c}_i^2 \right) = 0$$

$$(\mathbf{T} + \boldsymbol{w}\mathbf{I}) \cdot \boldsymbol{c} = \boldsymbol{y}$$

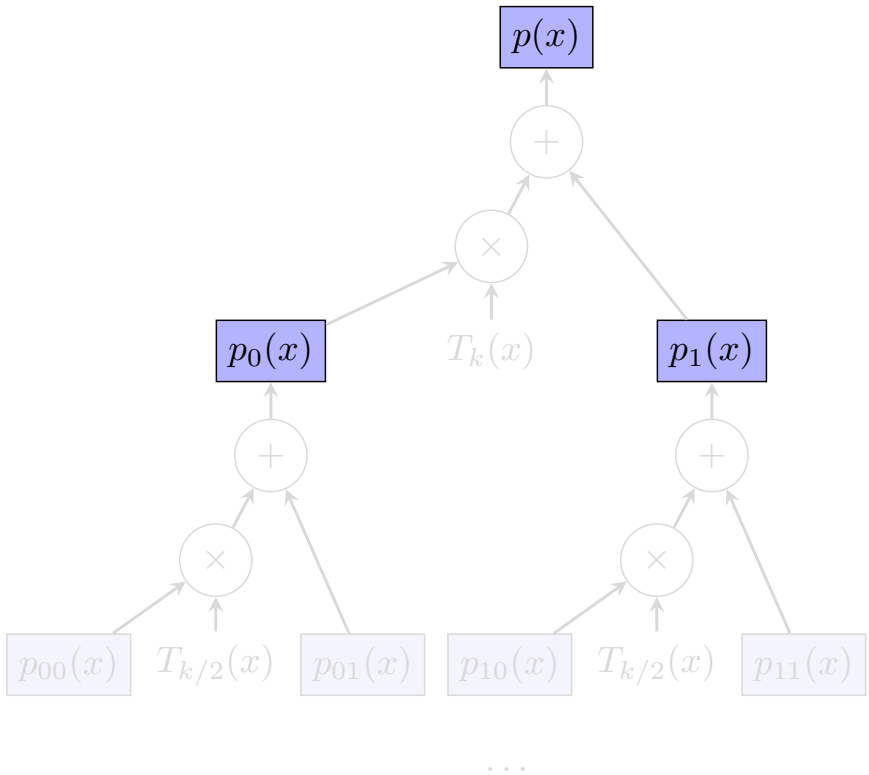
$$\mathbf{T} = \begin{bmatrix} E[\phi_1 \cdot \phi_1] & E[\phi_1 \cdot \phi_3] & \cdots & E[\phi_1 \cdot \phi_n] \\ E[\phi_3 \cdot \phi_1] & E[\phi_3 \cdot \phi_3] & \cdots & \vdots \\ \vdots & & \ddots & \vdots \\ E[\phi_n \cdot \phi_1] & E[\phi_n \cdot \phi_3] & \cdots & E[\phi_n \cdot \phi_n] \end{bmatrix}, \text{ and } \boldsymbol{y} = \begin{bmatrix} E[f_{\mathsf{mod}} \cdot \phi_1] \\ E[f_{\mathsf{mod}} \cdot \phi_3] \\ \vdots \\ E[f_{\mathsf{mod}} \cdot \phi_n] \end{bmatrix}$$

Outline

Baby-Step Giant-Step (BSGS) Algorithm

- Naive evaluation of degree d polynomial requires $O(d)$ multiplication
- BSGS algorithm reduces the complexity to $O(\sqrt{d})$
- In BSGS, a polynomial is **recursively** divided into smaller-degree polynomials

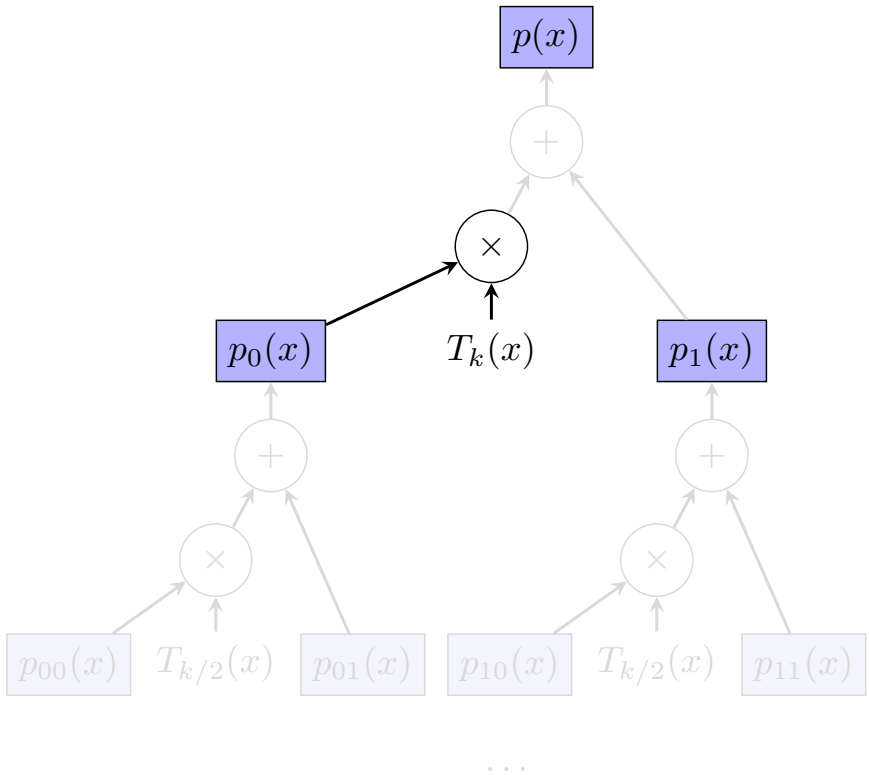
$$p(x) = p_0(x) \cdot T_k(x) + p_1(x)$$



Baby-Step Giant-Step (BSGS) Algorithm

- Naive evaluation of degree d polynomial requires $O(d)$ multiplication
- BSGS algorithm reduces the complexity to $O(\sqrt{d})$
- In BSGS, a polynomial is **recursively** divided into smaller-degree polynomials

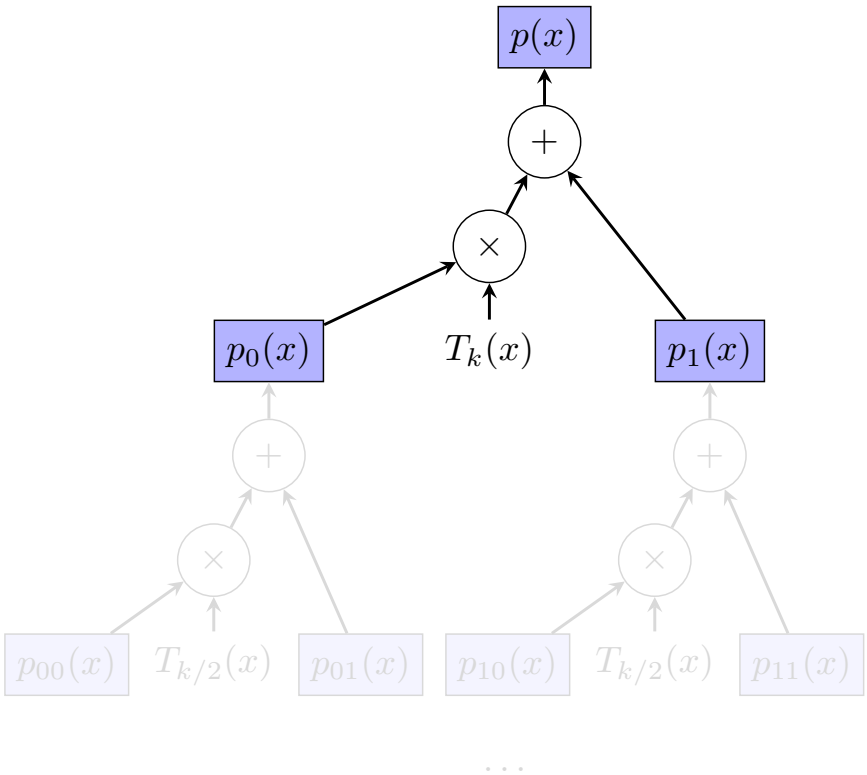
$$p(x) = p_0(x) \cdot T_k(x) + p_1(x)$$



Baby-Step Giant-Step (BSGS) Algorithm

- Naive evaluation of degree d polynomial requires $O(d)$ multiplication
- BSGS algorithm reduces the complexity to $O(\sqrt{d})$
- In BSGS, a polynomial is **recursively** divided into smaller-degree polynomials

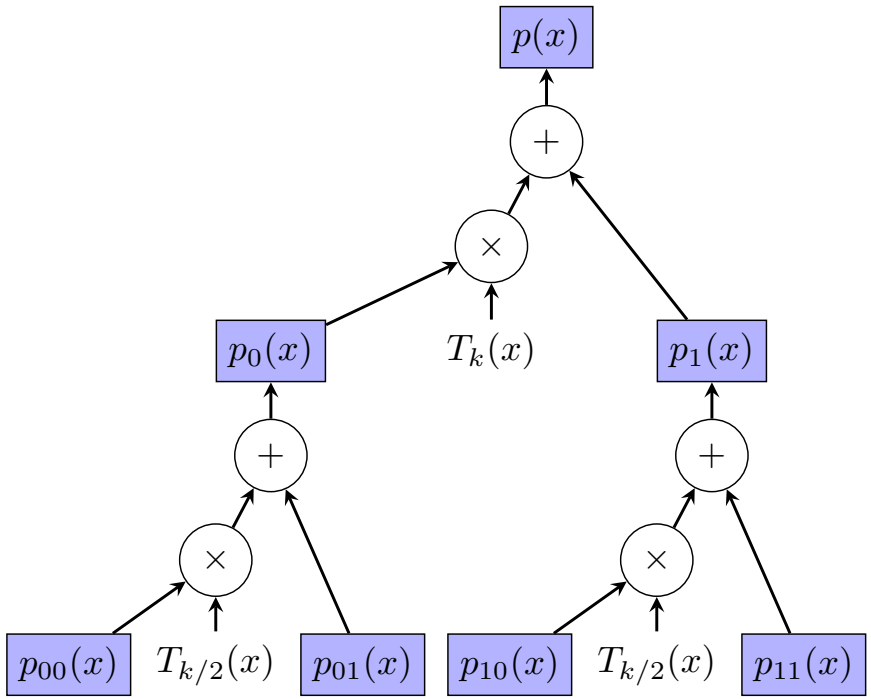
$$p(x) = p_0(x) \cdot T_k(x) + p_1(x)$$



Baby-Step Giant-Step (BSGS) Algorithm

- Naive evaluation of degree d polynomial requires $O(d)$ multiplication
- BSGS algorithm reduces the complexity to $O(\sqrt{d})$
- In BSGS, a polynomial is **recursively** divided into smaller-degree polynomials

$$p(x) = p_0(x) \cdot T_k(x) + p_1(x)$$



Building Blocks of BSGS and Lazy Operations

- Ciphertext-ciphertext multiplication for polynomial basis
- Plaintext-ciphertext multiplications for coefficients: without relinearization
- Addition after multiplying coefficients: without relinearization

Building Blocks of BSGS and Lazy Operations

- Ciphertext-ciphertext multiplication for polynomial basis
- Plaintext-ciphertext multiplications for coefficients: without relinearization
- Addition after multiplying coefficients: without relinearization

Lazy-BSGS Algorithm (simplified)

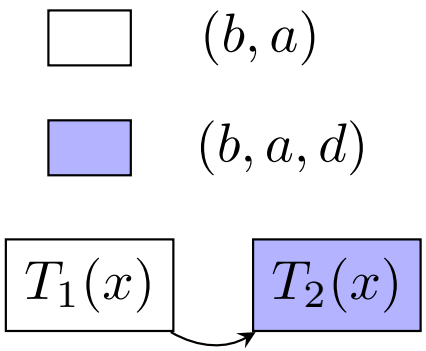
(b, a)

(b, a, d)

$T_1(x)$

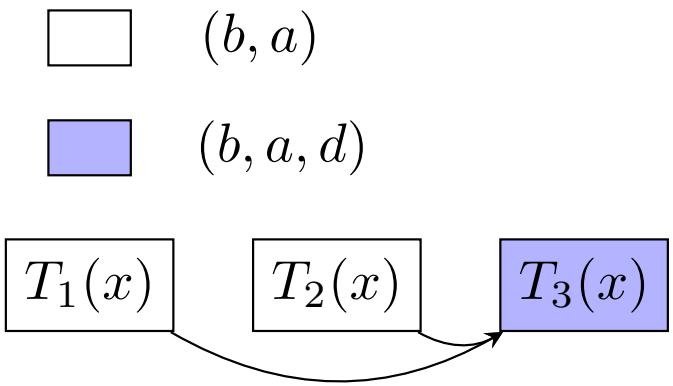
$p(x)$

Lazy-BSGS Algorithm (simplified)



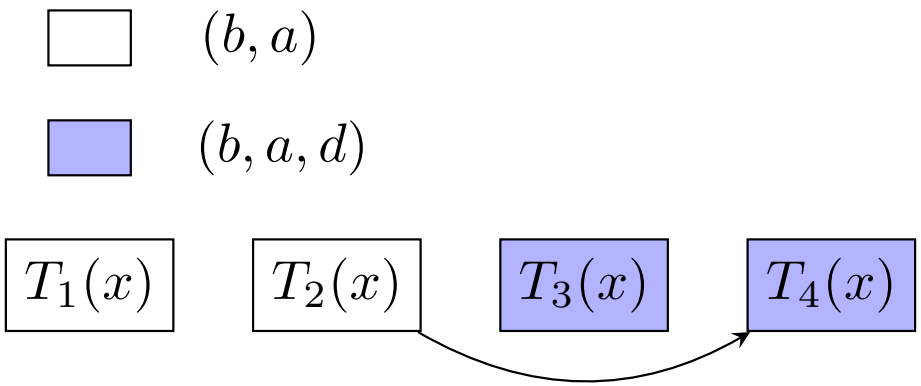
$p(x)$

Lazy-BSGS Algorithm (simplified)



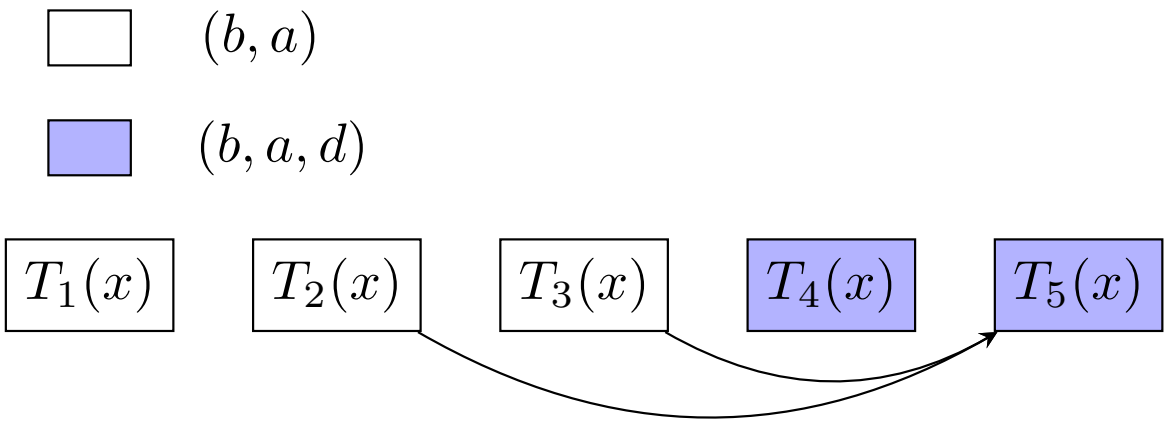
$p(x)$

Lazy-BSGS Algorithm (simplified)



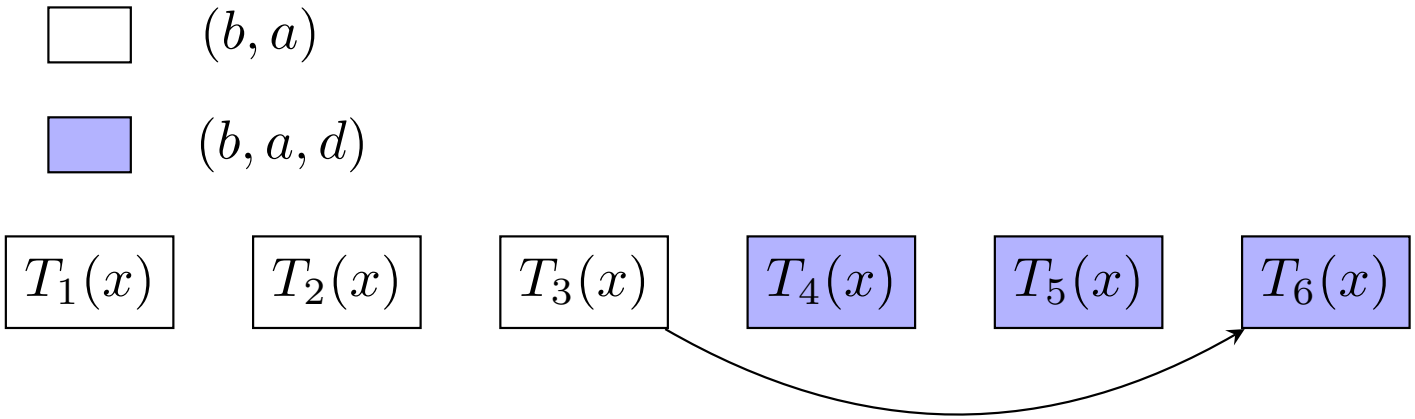
$p(x)$

Lazy-BSGS Algorithm (simplified)



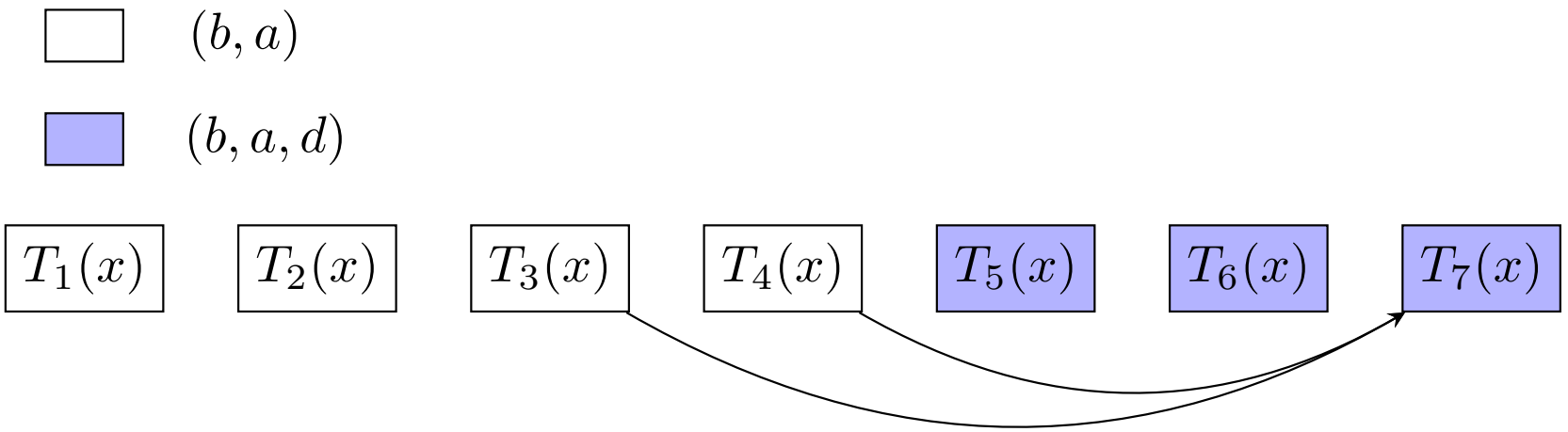
$p(x)$

Lazy-BSGS Algorithm (simplified)



$p(x)$

Lazy-BSGS Algorithm (simplified)



$p(x)$

Lazy-BSGS Algorithm (simplified)

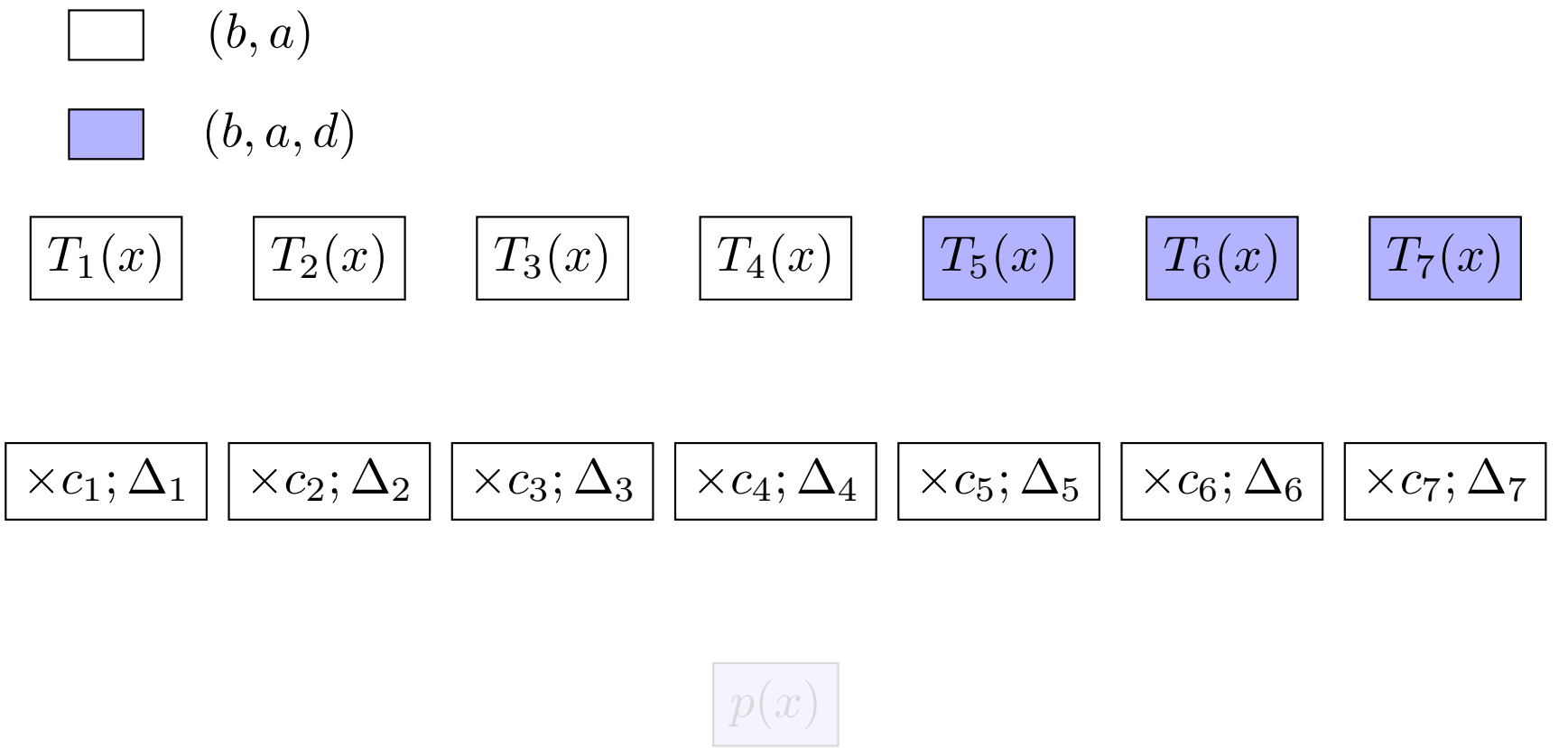
(b, a)

(b, a, d)

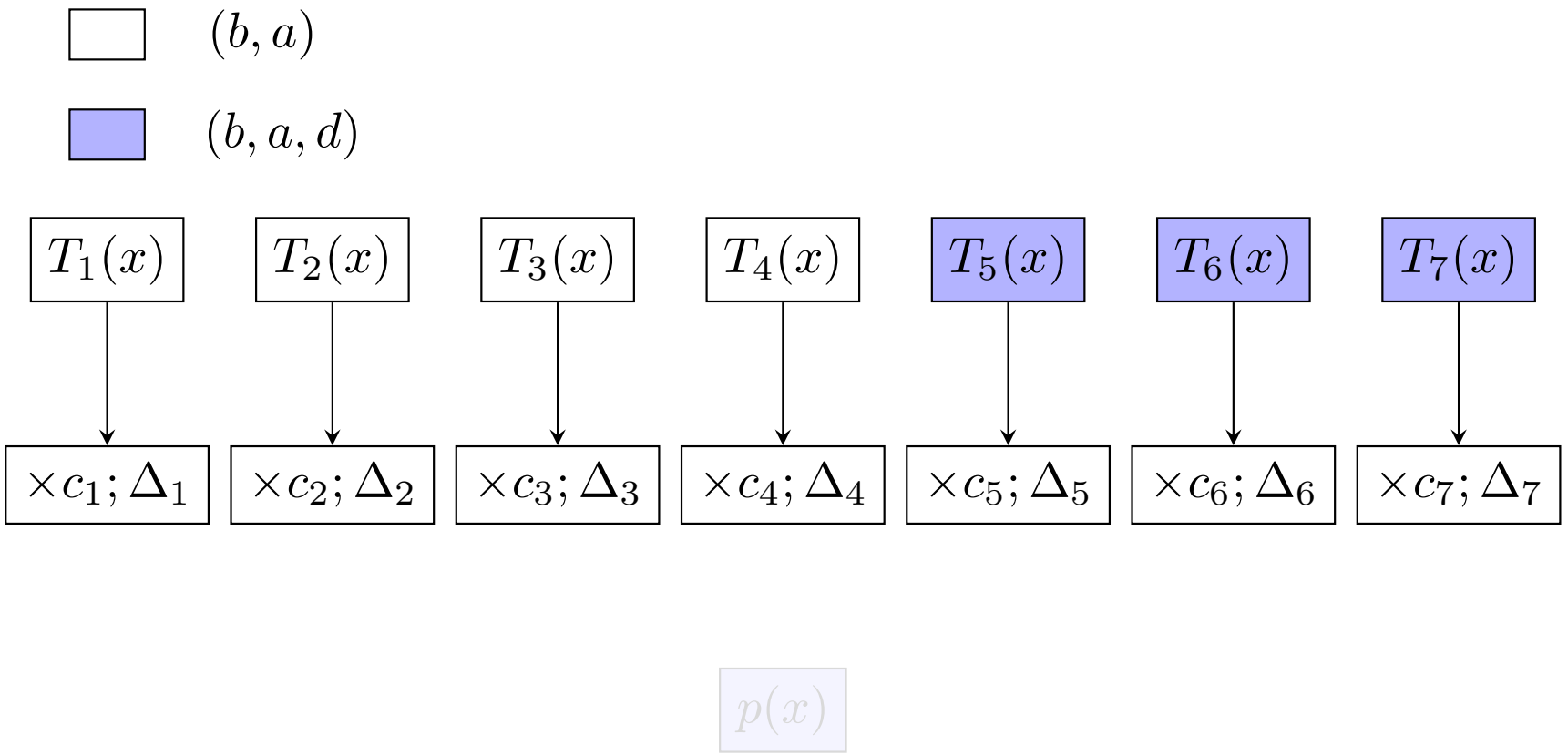
$T_1(x)$ $T_2(x)$ $T_3(x)$ $T_4(x)$ $T_5(x)$ $T_6(x)$ $T_7(x)$

$p(x)$

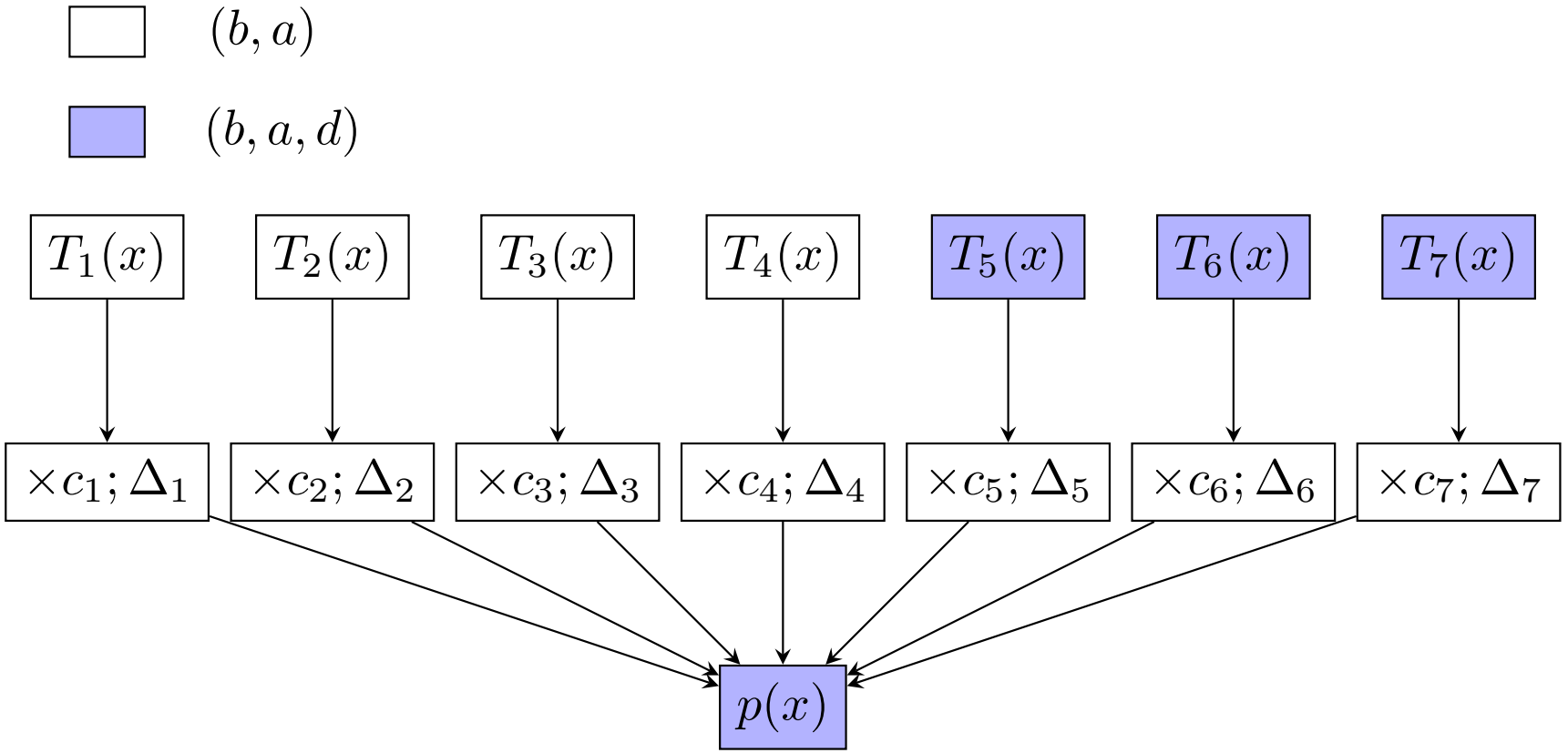
Lazy-BSGS Algorithm (simplified)



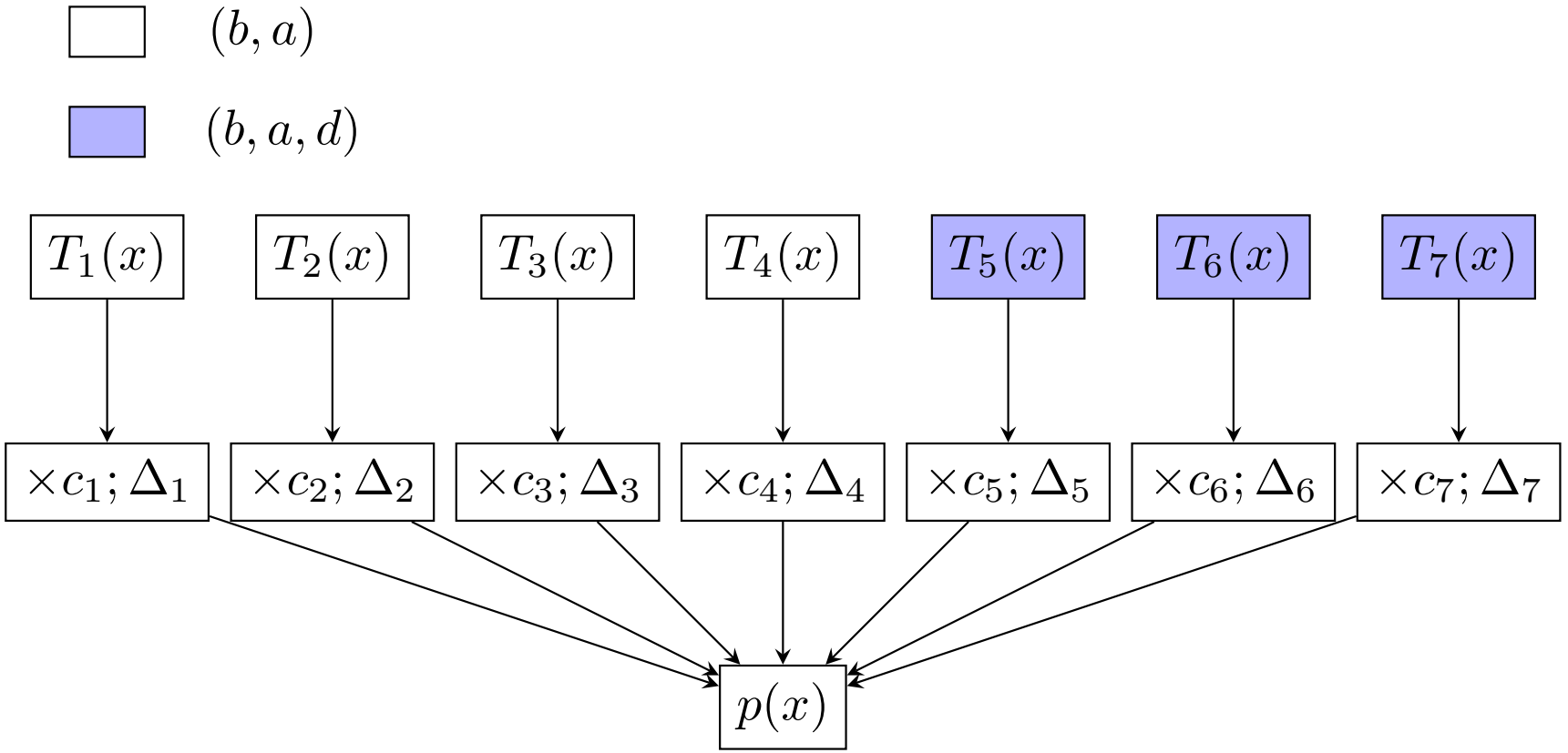
Lazy-BSGS Algorithm (simplified)



Lazy-BSGS Algorithm (simplified)



Lazy-BSGS Algorithm (simplified)



Comparison of Number of Multiplications

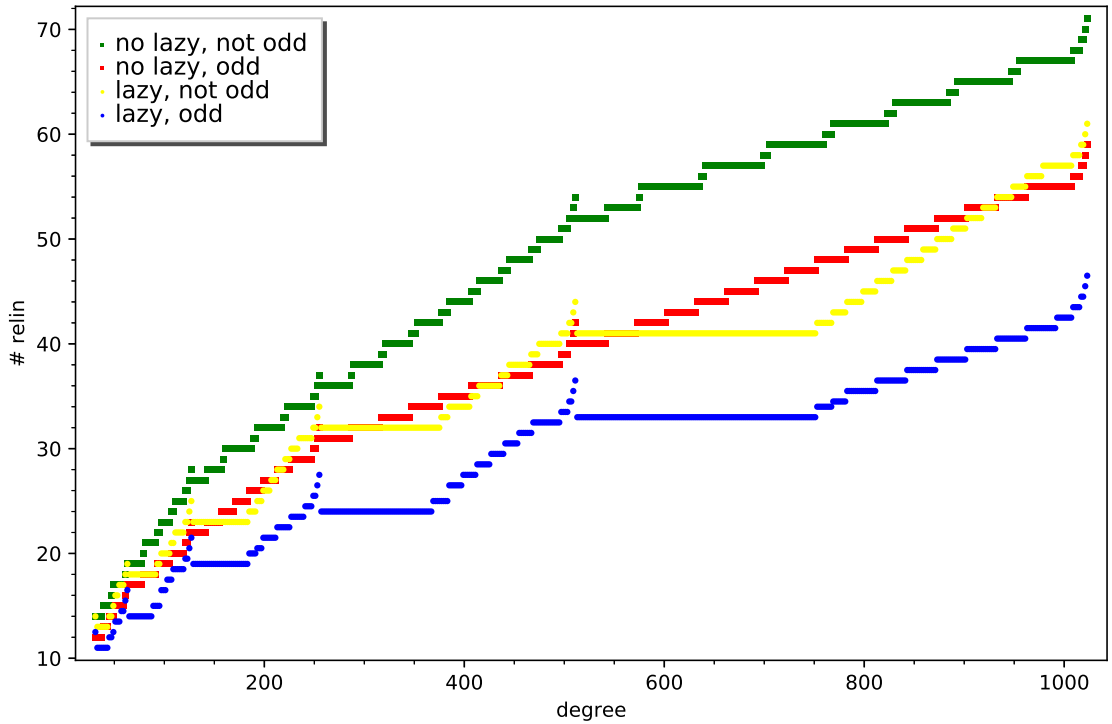


Figure: Number of relinearizations for the variants of BSGS algorithms.

Outline

Implementation Results

algorithm	EvalMod			depth	#relin	$Var[e_{\text{boot}}]$	bit prec.	runtime(s)
	cos	double	$\sin^{-1}$					
[LLL+21]	68	2	5	12	24	$2^{-64.5}$	32.6	451.5
[BMTH21]	62	2	7	11	24	$2^{-62.6}$	31.6	22.8
	62	2	3	10	22	$2^{-44.4}$	22.4	25.3
proposed	f_{mod}: 711			**10**	33	$2^{-62.1}$	31.4	28.3
proposed (high prec)	f_{mod}: 1625			**11**	46	$2^{-185.4}$	93.03	-
						$2^{-199.0}$	100.11	-

Table: Experimental result over `Lattigo` and `HEAAN`

Conclusions

- Two contributions for accurate and fast bootstrapping of CKKS scheme
  1. A method to find the optimal approximate polynomial of modular reduction for bootstrapping and its analytical solution by minimizing error variances
  2. More efficient algorithm to homomorphically evaluate a high-degree polynomial
- We can reserve more levels after bootstrapping or we can use the remaining level to speed up bootstrapping
- We discussed that the proposed method achieves the CKKS bootstrapping with very high accuracy, so we can directly apply the noise flooding technique to the CKKS scheme for IND-CPA$^{\mathrm{D}}$ security.

Thank you!
Questions?