# Fixed-Point Arithmetic in SHE Schemes

Anamaria Costache[1], Nigel P. Smart[1], **Srinivas Vivek**[1],

Adrian Waller[2]

[1]**University of Bristol**

[2]Thales UK Research & Technology

July 6, 2016

# Outline

- Motivation
- Encoding integers and fixed-point numbers
- Lower bounds on ring parameters
- Application to homomorphic image processing
- Conclusion

# Motivation

# Somewhat Homomorphic Encryption

- Huge improvement in efficiency of FHE schemes since 2009.
- Yet, practical FHE schemes seem out of reach.
- Goal: to obtain practical SHE schemes for a given class of functions.

# Somewhat Homomorphic Encryption

- Huge improvement in efficiency of FHE schemes since 2009.
- Yet, practical FHE schemes seem out of reach.
- Goal: to obtain practical SHE schemes for a given class of functions.

# Somewhat Homomorphic Encryption

- Huge improvement in efficiency of FHE schemes since 2009.
- Yet, practical FHE schemes seem out of reach.
- Goal: to obtain practical SHE schemes for a given class of functions.

# Problem 1

- Problem 1: how to encode/decode data types of an application into an SHE scheme?
- Application: emulate fixed point arithmetic.
- Target SHE schemes: ring-based SHE schemes
  - currently among the most efficient,
  - plaintext space $R = \mathbb{Z}[x]/(\Phi_m(X), p)$.
- Encoding considerably effects efficiency
  - working with binary circuits is usually inefficient in practice.

# Problem 1

- Problem 1: how to encode/decode data types of an application into an SHE scheme?
- Application: emulate fixed point arithmetic.
- Target SHE schemes: ring-based SHE schemes
  - currently among the most efficient,
  - plaintext space $R = \mathbb{Z}[x]/(\Phi_m(X), p)$.
- Encoding considerably effects efficiency
  - working with binary circuits is usually inefficient in practice.

# Problem 1

- Problem 1: how to encode/decode data types of an application into an SHE scheme?
- Application: emulate fixed point arithmetic.
- Target SHE schemes: ring-based SHE schemes
  - currently among the most efficient,
  - plaintext space $R = \mathbb{Z}[x]/(\Phi_m(X), p)$.
- Encoding considerably effects efficiency
  - working with binary circuits is usually inefficient in practice.

# Problem 1

- Problem 1: how to encode/decode data types of an application into an SHE scheme?
- Application: emulate fixed point arithmetic.
- Target SHE schemes: ring-based SHE schemes
  - currently among the most efficient,
  - plaintext space $R = \mathbb{Z}[x]/(\Phi_m(X), p)$.
- Encoding considerably effects efficiency
  - working with binary circuits is usually inefficient in practice.

# Problem 2

- The parameters of an SHE scheme depends upon
  - multiplicative depth,
  - plaintext modulus $p$ and degree of the ring $d$,
  - security level
- Problem 2: derive lower bounds on
  - plaintext modulus $p$
    - to ensure no wrap around the modulus $p$
  - degree of the ring $d$
    - to ensure no wrap around the modulus $q$ of $\mathcal{C}$

# Problem 2

- The parameters of an SHE scheme depends upon
  - multiplicative depth,
  - plaintext modulus $p$ and degree of the ring $d$,
  - security level
- Problem 2: derive lower bounds on
  - plaintext modulus $p$
    - to ensure no wrap around the modulus $p$
  - degree of the ring $d$
    - to ensure no wrap around the modulus $q$ of $\mathcal{C}_0$

# Problem 2

- The parameters of an SHE scheme depends upon
  - multiplicative depth,
  - plaintext modulus $p$ and degree of the ring $d$,
  - security level
- Problem 2: derive lower bounds on
  - plaintext modulus $p$
    - ⋆ to ensure no wrap around the modulus $p$,
  - degree of the ring $d$
    - ⋆ to ensure no wrap around the modulus $\Phi_m(X)$,

# Problem 2

- The parameters of an SHE scheme depends upon
  - multiplicative depth,
  - plaintext modulus $p$ and degree of the ring $d$,
  - security level
- Problem 2: derive lower bounds on
  - plaintext modulus $p$
    - ⋆ to ensure no wrap around the modulus $p$,
  - degree of the ring $d$
    - ⋆ to ensure no wrap around the modulus $\Phi_m(X)$,

# Problem 1: Encoding data types

# Encoding integers: Scalar encoding

- Integer encoded as the constant term.
- $p$ must be greater than the largest integer occurring.
- For a regular circuit of
  - multiplicative depth: $M$
  - No. of additions per multiplicative depth: $A$,
  - and for inputs in $[-L, \ldots, L]$

$$p > 2 \cdot 2^{A\left(2^{M+1}-2\right)} \cdot L^{2^M}.$$

# Encoding integers: Scalar encoding

- Integer encoded as the constant term.
- $p$ must be greater than the largest integer occurring.
- For a regular circuit of
  - multiplicative depth: $M$
  - No. of additions per multiplicative depth: $A$,
  - and for inputs in $[-L, \ldots, L]$

$$p > 2 \cdot 2^{A\left(2^{M+1}-2\right)} \cdot L^{2^M}.$$

# Encoding integers: Scalar encoding

- Integer encoded as the constant term.
- $p$ must be greater than the largest integer occurring.
- For a regular circuit of
  - multiplicative depth: $M$
  - No. of additions per multiplicative depth: $A$,
  - and for inputs in $[-L, \ldots, L]$

$$p > 2 \cdot 2^{A\left(2^{M+1}-2\right)} \cdot L^{2^M}.$$

# Encoding integers: Non-balanced base-$B$ encoding

- Integer encoded as a polynomial corresponding to base-$B$ expansion.
  - coefficients in $[0, .., B-1]$.
- *Example*: $19 \mapsto 2 \cdot X^2 + 1$, when base $B = 3$.
- Smaller-sized coefficients compared to scalar encoding.
- Every coefficient has the same sign.

# Encoding integers: Non-balanced base-$B$ encoding

- Integer encoded as a polynomial corresponding to base-$B$ expansion.
  - coefficients in $[0, .., B-1]$.
- *Example*: $19 \mapsto 2 \cdot X^2 + 1$, when base $B = 3$.
- Smaller-sized coefficients compared to scalar encoding.
- Every coefficient has the same sign.

# Encoding integers: Non-balanced base-$B$ encoding

- Integer encoded as a polynomial corresponding to base-$B$ expansion.
  - coefficients in $[0, .., B-1]$.
- *Example*: $19 \mapsto 2 \cdot X^2 + 1$, when base $B = 3$.
- Smaller-sized coefficients compared to scalar encoding.
- Every coefficient has the same sign.

# Encoding integers: Non-balanced base-$B$ encoding

- Integer encoded as a polynomial corresponding to base-$B$ expansion.
  - coefficients in $[0, .., B-1]$.
- *Example*: $19 \mapsto 2 \cdot X^2 + 1$, when base $B = 3$.
- Smaller-sized coefficients compared to scalar encoding.
- Every coefficient has the same sign.

# Encoding integers: Balanced base-$B$ encoding

- Integer encoded as a polynomial corresponding to base-$B$ expansion.
  - coefficients *instead* in $[-(B-1)/2, \ldots, (B-1)/2]$.
- *Example*: $19 \mapsto X^3 - X^2 - X - 1$, when bal. base $B = 3$.
- *Tradeoff* : size of coefficients and the degree of encodings.
- $B = 3$ turns out be **optimal**.
- Deriving bounds on the size of coefficients is more challenging.

# Encoding integers: Balanced base-$B$ encoding

- Integer encoded as a polynomial corresponding to base-$B$ expansion.
  - coefficients *instead* in $[-(B-1)/2, \ldots, (B-1)/2]$.
- *Example*: $19 \mapsto X^3 - X^2 - X - 1$, when bal. base $B = 3$.
- *Tradeoff*: size of coefficients and the degree of encodings.
- $B = 3$ turns out be **optimal**.
- Deriving bounds on the size of coefficients is more challenging.

# Encoding integers: Balanced base-$B$ encoding

- Integer encoded as a polynomial corresponding to base-$B$ expansion.
  - coefficients *instead* in $[-(B-1)/2, \ldots, (B-1)/2]$.
- *Example*: $19 \mapsto X^3 - X^2 - X - 1$, when bal. base $B = 3$.
- *Tradeoff*: size of coefficients and the degree of encodings.
- $B = 3$ turns out be **optimal**.
- Deriving bounds on the size of coefficients is more challenging.

# Encoding integers: Balanced base-$B$ encoding

- Integer encoded as a polynomial corresponding to base-$B$ expansion.
  - coefficients *instead* in $[-(B-1)/2, \ldots, (B-1)/2]$.
- *Example*: $19 \mapsto X^3 - X^2 - X - 1$, when bal. base $B = 3$.
- *Tradeoff*: size of coefficients and the degree of encodings.
- $B = 3$ turns out be **optimal**.
- Deriving bounds on the size of coefficients is more challenging.

# Encoding integers: Balanced base-$B$ encoding

- Integer encoded as a polynomial corresponding to base-$B$ expansion.
  - coefficients *instead* in $[-(B-1)/2, \ldots, (B-1)/2]$.
- *Example*: $19 \mapsto X^3 - X^2 - X - 1$, when bal. base $B = 3$.
- *Tradeoff*: size of coefficients and the degree of encodings.
- $B = 3$ turns out be **optimal**.
- Deriving bounds on the size of coefficients is more challenging.

# Encoding fixed-point numbers: Scaled integer encoding

- Fixed-point number encoded as an integer scaled down by a power of a base $B$.

$$y = y^+.y^- \mapsto (q(X), i), \qquad y = q(B) * B^{-i},$$

where the integer itself is represented by a balanced base-$B$ encoding.

- *Example*: $6.33\ldots = 19 \times 3^{-1} \mapsto (X^3 - X^2 + 1, 1)$, when bal. base $B = 3$.

# Encoding fixed-point numbers: Scaled integer encoding

- Fixed-point number encoded as an integer scaled down by a power of a base $B$.

$$y = y^+.y^- \mapsto (q(X), i), \qquad y = q(B) * B^{-i},$$

where the integer itself is represented by a balanced base-$B$ encoding.

- *Example*: $6.33\ldots = 19 \times 3^{-1} \mapsto (X^3 - X^2 + 1, 1)$, when bal. base $B = 3$.

# Encoding fixed-point numbers: Scaled integer encoding

- **Addition** $+$:

$$(q(X), i) + (q'(X), i') =: (Q, I).$$

$$(Q, I) = \begin{cases} (q + q' \cdot X^{i-i'}, i) & \text{if } i > i' \\ (q' + q \cdot X^{i'-i}, i') & \text{if } i' \geq i. \end{cases}$$

- **Multiplication** $\times$:

$$(q(X), i) \times (q'(X), i') = (q(X) \cdot q'(X), i + i').$$

# Encoding fixed-point numbers: Scaled integer encoding

- **Addition** $+$:

$$(q(X), i) + (q'(X), i') =: (Q, I).$$

$$(Q, I) = \begin{cases} (q + q' \cdot X^{i-i'}, i) & \text{if } i > i' \\ (q' + q \cdot X^{i'-i}, i') & \text{if } i' \geq i. \end{cases}$$

- **Multiplication** $\times$:

$$(q(X), i) \times (q'(X), i') = (q(X) \cdot q'(X), i + i').$$

# Encoding fixed-point numbers: Scaled integer encoding

- Let $\mathcal{R}_1 = \{(q, i) \mid q \in \mathbb{Z}[X]/\Phi_m(X), i \in \mathbb{Z}/\phi(m)\mathbb{Z}\}$.
- $\mathcal{R}_1$ is a ring w.r.t. $+$ and $\times$.
- *Disadvantage*: Keep track of the exponent $i$ in the clear for every ciphertext
  - not an issue if the evaluated circuit is public.
- *Bounds*: same as the balanced base integer representation

# Encoding fixed-point numbers: Scaled integer encoding

- Let $\mathcal{R}_1 = \{(q, i) \mid q \in \mathbb{Z}[X]/\Phi_m(X), i \in \mathbb{Z}/\phi(m)\mathbb{Z}\}$.
- $\mathcal{R}_1$ is a ring w.r.t. $+$ and $\times$.
- *Disadvantage*: Keep track of the exponent $i$ in the clear for every ciphertext
  - not an issue if the evaluated circuit is public.
- *Bounds*: same as the balanced base integer representation

# Encoding fixed-point numbers: Scaled integer encoding

- Let $\mathcal{R}_1 = \{(q, i) \mid q \in \mathbb{Z}[X]/\Phi_m(X), i \in \mathbb{Z}/\phi(m)\mathbb{Z}\}$.
- $\mathcal{R}_1$ is a ring w.r.t. $+$ and $\times$.
- *Disadvantage*: Keep track of the exponent $i$ in the clear for every ciphertext
  - not an issue if the evaluated circuit is public.
- *Bounds*: same as the balanced base integer representation

# Encoding fixed-point numbers: Scaled integer encoding

- Let $\mathcal{R}_1 = \{(q, i) \mid q \in \mathbb{Z}[X]/\Phi_m(X), i \in \mathbb{Z}/\phi(m)\mathbb{Z}\}$.
- $\mathcal{R}_1$ is a ring w.r.t. $+$ and $\times$.
- *Disadvantage*: Keep track of the exponent $i$ in the clear for every ciphertext
  - not an issue if the evaluated circuit is public.
- *Bounds*: same as the balanced base integer representation

# Encoding fixed-point numbers: Fractional encoding

- Proposed by [Dowlin et al. 2015].
- Closely related to Laurent polynomial representation.
- Suppose we are working in $\mathbb{Z}[X]/(X^n + 1)$.

$$y = y^+.y^- \mapsto \sum_{i \leq l^+} b_i \cdot X^i + \sum_{0 < i \leq l^-} b_{-i} \cdot X^{-i} \pmod{X^n + 1},$$

- Since $-X^n \equiv 1 \pmod{X^n + 1}$,

$$y = y^+.y^- \mapsto \sum_{i \leq l^+} b_i \cdot X^i - \sum_{0 < i \leq l^-} b_{-i} \cdot X^{n-i} \pmod{X^n + 1}.$$

# Encoding fixed-point numbers: Fractional encoding

- Proposed by [Dowlin et al. 2015].
- Closely related to Laurent polynomial representation.
- Suppose we are working in $\mathbb{Z}[X]/(X^n + 1)$.

$$y = y^+.y^- \mapsto \sum_{i \leq l^+} b_i \cdot X^i + \sum_{0 < i \leq l^-} b_{-i} \cdot X^{-i} \pmod{X^n + 1},$$

- Since $-X^n \equiv 1 \pmod{X^n + 1}$,

$$y = y^+.y^- \mapsto \sum_{i \leq l^+} b_i \cdot X^i - \sum_{0 < i \leq l^-} b_{-i} \cdot X^{n-i} \pmod{X^n + 1}.$$

# Encoding fixed-point numbers: Fractional encoding

- Proposed by [Dowlin et al. 2015].
- Closely related to Laurent polynomial representation.
- Suppose we are working in $\mathbb{Z}[X]/(X^n + 1)$.

$$y = y^+.y^- \mapsto \sum_{i \leq l^+} b_i \cdot X^i + \sum_{0 < i \leq l^-} b_{-i} \cdot X^{-i} \pmod{X^n + 1},$$

- Since $-X^n \equiv 1 \pmod{X^n + 1}$,

$$y = y^+.y^- \mapsto \sum_{i \leq l^+} b_i \cdot X^i - \sum_{0 < i \leq l^-} b_{-i} \cdot X^{n-i} \pmod{X^n + 1}.$$

# Encoding fixed-point numbers: Fractional encoding

- Proposed by [Dowlin et al. 2015].
- Closely related to Laurent polynomial representation.
- Suppose we are working in $\mathbb{Z}[X]/(X^n + 1)$.

$$y = y^+.y^- \mapsto \sum_{i \leq l^+} b_i \cdot X^i + \sum_{0 < i \leq l^-} b_{-i} \cdot X^{-i} \pmod{X^n + 1},$$

- Since $-X^n \equiv 1 \pmod{X^n + 1}$,

$$y = y^+.y^- \mapsto \sum_{i \leq l^+} b_i \cdot X^i - \sum_{0 < i \leq l^-} b_{-i} \cdot X^{n-i} \pmod{X^n + 1}.$$

# Encoding fixed-point numbers: Fractional encoding

- *Example*: in $\mathbb{Z}[X]/(X^8+1)$ and bal. base $B=3$,
  - $6.33\ldots \mapsto -X^7+X^2-X$.
- *Advantage*: "Dispenses away" the need for tracking the exponent
  - need to separate coefficients in integer and fractional parts.
- *Disadvantage*
  - works only in power-of-two cyclotomic rings,
  - no slots for SIMD operations.
- **Addition** and **Multiplication**: as in $\mathcal{R}_2 := \mathbb{Z}[X]/(X^n+1)$.
- *Bounds*: seems more complicated than the scaled integer case?

# Encoding fixed-point numbers: Fractional encoding

- *Example*: in $\mathbb{Z}[X]/(X^8+1)$ and bal. base $B=3$,
  - $6.33\ldots \mapsto -X^7+X^2-X$.
- *Advantage*: "Dispenses away" the need for tracking the exponent
  - need to separate coefficients in integer and fractional parts.
- *Disadvantage*
  - works only in power-of-two cyclotomic rings,
  - no slots for SIMD operations.
- **Addition** and **Multiplication**: as in $\mathcal{R}_2 := \mathbb{Z}[X]/(X^n+1)$.
- *Bounds*: seems more complicated than the scaled integer case?

# Encoding fixed-point numbers: Fractional encoding

- *Example*: in $\mathbb{Z}[X]/(X^8+1)$ and bal. base $B=3$,
  - $6.33\ldots \mapsto -X^7+X^2-X$.
- *Advantage*: "Dispenses away" the need for tracking the exponent
  - need to separate coefficients in integer and fractional parts.
- *Disadvantage*
  - works only in power-of-two cyclotomic rings,
  - no slots for SIMD operations.
- **Addition** and **Multiplication**: as in $\mathcal{R}_2 := \mathbb{Z}[X]/(X^n+1)$.
- *Bounds*: seems more complicated than the scaled integer case?

# Encoding fixed-point numbers: Fractional encoding

- *Example*: in $\mathbb{Z}[X]/(X^8+1)$ and bal. base $B=3$,
  - $6.33\ldots \mapsto -X^7+X^2-X$.
- *Advantage*: "Dispenses away" the need for tracking the exponent
  - need to separate coefficients in integer and fractional parts.
- *Disadvantage*
  - works only in power-of-two cyclotomic rings,
  - no slots for SIMD operations.
- **Addition** and **Multiplication**: as in $\mathcal{R}_2 := \mathbb{Z}[X]/(X^n+1)$.
- *Bounds*: seems more complicated than the scaled integer case?

# Encoding fixed-point numbers: Fractional encoding

- *Example*: in $\mathbb{Z}[X]/(X^8+1)$ and bal. base $B=3$,
  - $6.33\ldots \mapsto -X^7+X^2-X$.
- *Advantage*: "Dispenses away" the need for tracking the exponent
  - need to separate coefficients in integer and fractional parts.
- *Disadvantage*
  - works only in power-of-two cyclotomic rings,
  - no slots for SIMD operations.
- **Addition** and **Multiplication**: as in $\mathcal{R}_2 := \mathbb{Z}[X]/(X^n+1)$.
- *Bounds*: seems more complicated than the scaled integer case?

# Problem 2: bounding size of coefficients

# Upper bounds on coefficients

- Problem: compute max. size of coefficients in encodings given
  - circuit description,
  - bounds on input numbers
- Our contribution:
  - efficient computational procedure to bound arbitrary circuits
    - precisely determining max. values can be very expensive.
  - closed form upper bounds for regular circuits.

# Upper bounds on coefficients

- Problem: compute max. size of coefficients in encodings given
  - circuit description,
  - bounds on input numbers
- Our contribution:
  - efficient computational procedure to bound arbitrary circuits
    - ★ precisely determining max. values can be very expensive,
  - closed form upper bounds for regular circuits.

# Upper bounds on coefficients

- Two cases:
  - balanced base-3 integer encoding $\equiv$ scaled integer encoding,
  - fractional encoding (modulo $X^n + 1$) for fixed-point numbers.

# Equivalence: scaled integer vs. fractional encoding

- Ring $\mathcal{R}_1$ of scaled integer encodings
  - $\mathcal{R}_1 := \{(q, i) \mid q \in \mathbb{Z}[X]/(X^n + 1),\ i \in \mathbb{Z}/n\mathbb{Z})\}$.
- Ring $\mathcal{R}_2$ of fractional encodings
  - $\mathcal{R}_2 := \mathbb{Z}[X]/(X^n + 1)$.
- $\mathcal{R}_1$ is **isomorphic** to $\mathcal{R}_2$

$$\Psi : \begin{cases} \mathcal{R}_1 & \to & \mathcal{R}_2 \\ (q := q_I \cdot X^i + q_F, i) & \mapsto & q_I - q_F \cdot X^{n-i} \end{cases}$$

# Equivalence: scaled integer vs. fractional encoding

- Ring $\mathcal{R}_1$ of scaled integer encodings
  - $\mathcal{R}_1 := \{(q, i) \mid q \in \mathbb{Z}[X]/(X^n + 1),\ i \in \mathbb{Z}/n\mathbb{Z})\}$ .
- Ring $\mathcal{R}_2$ of fractional encodings
  - $\mathcal{R}_2 := \mathbb{Z}[X]/(X^n + 1)$.
- $\mathcal{R}_1$ is **isomorphic** to $\mathcal{R}_2$

$$\Psi : \begin{cases} \mathcal{R}_1 & \to & \mathcal{R}_2 \\ (q := q_I \cdot X^i + q_F, i) & \mapsto & q_I - q_F \cdot X^{n-i} \end{cases}$$

# Equivalence: scaled integer vs. fractional encoding

- Ring $\mathcal{R}_1$ of scaled integer encodings
  - $\mathcal{R}_1 := \{(q, i) \mid q \in \mathbb{Z}[X]/(X^n + 1),\ i \in \mathbb{Z}/n\mathbb{Z})\}$ .
- Ring $\mathcal{R}_2$ of fractional encodings
  - $\mathcal{R}_2 := \mathbb{Z}[X]/(X^n + 1)$.
- $\mathcal{R}_1$ is **isomorphic** to $\mathcal{R}_2$

$$\Psi : \begin{cases} \mathcal{R}_1 & \to & \mathcal{R}_2 \\ (q := q_I \cdot X^i + q_F, i) & \mapsto & q_I - q_F \cdot X^{n-i} \end{cases}$$

# Equivalence: scaled integer vs. fractional encoding

- *Example*: encoding 6.33
  - using balanced base-3 representation and $\mathbb{Z}[X]/(X^8+1)$,
  - scaled integer encoding ($\mathcal{R}_1$): $(X^3 - X^2 + 1, 1)$,
  - fractional encoding ($\mathcal{R}_2$): $-X^7 + X^2 - X$,

# Equivalence: scaled integer vs. fractional encoding

- *Example*: encoding 6.33
  - using balanced base-3 representation and $\mathbb{Z}[X]/(X^8+1)$,
  - scaled integer encoding ($\mathcal{R}_1$): $(X^3 - X^2 + 1, 1)$,
  - fractional encoding ($\mathcal{R}_2$): $-X^7 + X^2 - X$,

# Equivalence: scaled integer vs. fractional encoding

- *Example*: encoding 6.33
  - using balanced base-3 representation and $\mathbb{Z}[X]/(X^8+1)$,
  - scaled integer encoding ($\mathcal{R}_1$): $(X^3 - X^2 + 1, 1)$,
  - fractional encoding ($\mathcal{R}_2$): $-X^7 + X^2 - X$,

# Equivalence: scaled integer vs. fractional encoding

- *Example*: encoding 6.33
  - using balanced base-3 representation and $\mathbb{Z}[X]/(X^8+1)$,
  - scaled integer encoding ($\mathcal{R}_1$): $(X^3 - X^2 + 1, 1)$,
  - fractional encoding ($\mathcal{R}_2$): $-X^7 + X^2 - X$,

# Equivalence: scaled integer vs. fractional encoding

- *Example*: encoding 6.33
  - using balanced base-3 representation and $\mathbb{Z}[X]/(X^8+1)$,
  - scaled integer encoding ($\mathcal{R}_1$): $(X^3 - X^2 + 1, 1)$,
  - fractional encoding ($\mathcal{R}_2$): $-X^7 + X^2 - X$,

$$(X^3 - X^2 + 1) \cdot (-X^7) = -X^7 + X^2 - X \pmod{X^8 + 1}.$$

- *Note*: infinity norm of intermediate polynomials is identical for any circuit.
- Suffices to analyse only the integer case.

# Equivalence: scaled integer vs. fractional encoding

- *Example*: encoding 6.33
  - using balanced base-3 representation and $\mathbb{Z}[X]/(X^8+1)$,
  - scaled integer encoding ($\mathcal{R}_1$): $(X^3 - X^2 + 1, 1)$,
  - fractional encoding ($\mathcal{R}_2$): $-X^7 + X^2 - X$,

$$(X^3 - X^2 + 1) \cdot (-X^7) = -X^7 + X^2 - X \pmod{X^8+1}.$$

- *Note*: infinity norm of intermediate polynomials is identical for any circuit.
- Suffices to analyse only the integer case.

# Equivalence: scaled integer vs. fractional encoding

- *Example*: encoding 6.33
  - using balanced base-3 representation and $\mathbb{Z}[X]/(X^8+1)$,
  - scaled integer encoding ($\mathcal{R}_1$): $(X^3 - X^2 + 1, 1)$,
  - fractional encoding ($\mathcal{R}_2$): $-X^7 + X^2 - X$,

$$(X^3 - X^2 + 1) \cdot (-X^7) = -X^7 + X^2 - X \pmod{X^8+1}.$$

- *Note*: infinity norm of intermediate polynomials is identical for any circuit.
- Suffices to analyse only the integer case.

# Bounding integer-valued arithmetic circuits

- Suppose there are $t$ distinct input ranges $[-L_i, \ldots, L_i]$.
- $d_i$: degree of corresponding balanced base-3 encoding polynomials.
- Input encoding polynomials have infinity norm 1.
- Define

$$c_{[(d_1,e_1),\ldots,(d_t,e_t)]} = \left\| \prod_{i=1}^{t} (1 + x + x^2 + \ldots + x^{d_i})^{e_i} \right\|_\infty .$$

# Bounding integer-valued arithmetic circuits

- Suppose there are $t$ distinct input ranges $[-L_i, \ldots, L_i]$.
- $d_i$: degree of corresponding balanced base-3 encoding polynomials.
- Input encoding polynomials have infinity norm 1.
- Define

$$c_{[(d_1,e_1),\ldots,(d_t,e_t)]} = \left\| \prod_{i=1}^{t} (1 + x + x^2 + \ldots + x^{d_i})^{e_i} \right\|_\infty .$$

# Bounding integer-valued arithmetic circuits

- Suppose there are $t$ distinct input ranges $[-L_i, \ldots, L_i]$.
- $d_i$: degree of corresponding balanced base-3 encoding polynomials.
- Input encoding polynomials have infinity norm 1.
- Define

$$c_{[(d_1,e_1),\ldots,(d_t,e_t)]} = \left\| \prod_{i=1}^{t} (1 + x + x^2 + \ldots + x^{d_i})^{e_i} \right\|_\infty .$$

# Bounding integer-valued arithmetic circuits

- Suppose there are $t$ distinct input ranges $[-L_i, \ldots, L_i]$.
- $d_i$: degree of corresponding balanced base-3 encoding polynomials.
- Input encoding polynomials have infinity norm 1.
- Define

$$c_{[(d_1,e_1),\ldots,(d_t,e_t)]} = \left\| \prod_{i=1}^{t} (1 + x + x^2 + \ldots + x^{d_i})^{e_i} \right\|_\infty .$$

# Bounding integer-valued arithmetic circuits

- Upper bound is of the form

$$L_P = \sum_{e_1,\ldots,e_t} a_{[(d_1,e_1),\ldots,(d_t,e_t)]} \cdot c_{[(d_1,e_1),\ldots,(d_t,e_t)]},$$

# Bounding integer-valued arithmetic circuits

- Compute the bounds iteratively

$$L_P = \sum_{e_1,\ldots,e_t} a_{[(d_1,e_1),\ldots,(d_t,e_t)]} \cdot c_{[(d_1,e_1),\ldots,(d_t,e_t)]},$$

$$L_{P'} = \sum_{e'_1,\ldots,e'_t} a_{[(d_1,e'_1),\ldots,(d_t,e'_t)]} \cdot c_{[(d_1,e'_1),\ldots,(d_t,e'_t)]},$$

# Bounding integer-valued arithmetic circuits

- Compute the bounds iteratively

$$L_{P+P'} = L_P + L_{P'},$$

$$L_{P \cdot P'} = \sum_{e_1,\ldots,e_t,e'_1,\ldots,e'_t} \left( a_{[(d_1,e_1),\ldots,(d_t,e_t)]} \cdot a_{[(d_1,e'_1),\ldots,(d_t,e'_t)]} \right) \cdot \left( c_{[(d_1,e_1+e'_1),\ldots,(d_t,e_t+e'_t)]} \right)$$

# Bounding integer-valued arithmetic circuits

- Iteratively bounding

$$c_{[(d_1,e_1),\ldots,(d_t,e_t)]} = \left\| \prod_{i=1}^{t} (1 + x + x^2 + \ldots + x^{d_i})^{e_i} \right\|_\infty .$$

using the relation

$$c_{[(d_1,e_1),\ldots,(d_t,e_t)]} \leq (d_k \cdot e_k + 1) \cdot c_{d_k,e_k} \cdot c_{[(d_1,e'_1),\ldots,(d_t,e'_t)]},$$

where $e'_i = e_i$ except that $e'_k = 0$.

- Partial order: $d_i \cdot e_i \leq (d_{i+1} \cdot e_{i+1})$.

# Bounding integer-valued arithmetic circuits

- Iteratively bounding

$$c_{[(d_1,e_1),\ldots,(d_t,e_t)]} = \left\| \prod_{i=1}^{t} (1 + x + x^2 + \ldots + x^{d_i})^{e_i} \right\|_\infty .$$

  using the relation

$$c_{[(d_1,e_1),\ldots,(d_t,e_t)]} \leq (d_k \cdot e_k + 1) \cdot c_{d_k,e_k} \cdot c_{[(d_1,e'_1),\ldots,(d_t,e'_t)]},$$

  where $e'_i = e_i$ except that $e'_k = 0$.

- Partial order: $d_i \cdot e_i \leq (d_{i+1} \cdot e_{i+1})$.

# Bounding integer-valued arithmetic circuits

- Finally, bounding

$$c_{d,e} = \left\| (1 + x + x^2 + \ldots + x^d)^e \right\|_\infty .$$

- **Simple bounds**: $\frac{(d+1)^e}{d \cdot e + 1} \leq c_{d,e} \leq (d+1)^e$ .
- **Tighter bound** [Mattner & Roos 2008]: If $e \neq 2$ or $d \in \{1, 2, 3\}$, then

$$c_{d,e} < \sqrt{\frac{6}{\pi \cdot d \cdot e \cdot (d+2)}} \cdot (d+1)^e .$$

# Bounding integer-valued arithmetic circuits

- Finally, bounding

$$c_{d,e} = \left\| (1 + x + x^2 + \ldots + x^d)^e \right\|_\infty .$$

- **Simple bounds**: $\frac{(d+1)^e}{d \cdot e + 1} \leq c_{d,e} \leq (d+1)^e$ .
- **Tighter bound** [Mattner & Roos 2008]: If $e \neq 2$ or $d \in \{1, 2, 3\}$, then

$$c_{d,e} < \sqrt{\frac{6}{\pi \cdot d \cdot e \cdot (d+2)}} \cdot (d+1)^e .$$

# Bounding integer-valued arithmetic circuits

- Finally, bounding

$$c_{d,e} = \left\| (1 + x + x^2 + \ldots + x^d)^e \right\|_\infty .$$

- **Simple bounds**: $\frac{(d+1)^e}{d \cdot e + 1} \leq c_{d,e} \leq (d+1)^e$ .
- **Tighter bound** [Mattner & Roos 2008]: If $e \neq 2$ or $d \in \{1, 2, 3\}$, then

$$c_{d,e} < \sqrt{\frac{6}{\pi \cdot d \cdot e \cdot (d+2)}} \cdot (d+1)^e .$$

# Bounding integer-valued arithmetic circuits

- They also show that

$$\lim_{e\to\infty} \frac{\sqrt{e}\cdot c_{d,e}}{(d+1)^e} = \sqrt{\frac{6}{\pi\cdot d\cdot(d+2)}}.$$

# Bounding integer-valued arithmetic circuits

- For a regular circuit having
  - $M$ levels of multiplication,
  - $A$ additions per multiplicative level,
  - max. initial degree of encodings $d = \lceil \log(2 \cdot L + 1) / \log 3 \rceil - 1$,

$$B_{M,A} = c_{d,2^M} \cdot 2^{A\left(2^{M+1}-2\right)},$$

# Bounding integer-valued arithmetic circuits

- For a regular circuit having
  - $M$ levels of multiplication,
  - $A$ additions per multiplicative level,
  - max. initial degree of encodings $d = \lceil \log(2 \cdot L + 1) / \log 3 \rceil - 1$,

$$B_{M,A} < \sqrt{\frac{6}{\pi \cdot 2^M \cdot d(d+2)}} \cdot (d+1)^{2^M} \cdot 2^{A\left(2^{M+1}-2\right)}.$$

  - For an SHE scheme

$$p > 2 \cdot B_{M,A}, \qquad \deg(\mathcal{R}) > 2^M \cdot d.$$

# Bounding integer-valued arithmetic circuits

- For a regular circuit having
  - $M$ levels of multiplication,
  - $A$ additions per multiplicative level,
  - max. initial degree of encodings $d = \lceil \log(2 \cdot L + 1) / \log 3 \rceil - 1$,

$$B_{M,A} < \sqrt{\frac{6}{\pi \cdot 2^M \cdot d(d+2)}} \cdot (d+1)^{2^M} \cdot 2^{A\left(2^{M+1}-2\right)}.$$

  - For an SHE scheme

$$p > 2 \cdot B_{M,A}, \qquad \deg(\mathcal{R}) > 2^M \cdot d.$$

# Bounding integer-valued arithmetic circuits

- Concrete bounds for $p$ (in bits) and $\deg(\mathcal{R})$ for 20-bit inputs:

| $M$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $A = 0$ | 5 | 12 | 26 | 55 | 114 |
| $A = 1$ | 7 | 18 | 40 | 85 | 176 |
| $A = 2$ | 9 | 24 | 54 | 115 | 238 |
| $A = 3$ | 11 | 30 | 68 | 145 | 300 |
| $A = 4$ | 13 | 36 | 82 | 175 | 362 |
| $A = 5$ | 15 | 42 | 96 | 205 | 424 |
| $\deg(\mathcal{R})$ | 24 | 48 | 96 | 192 | 384 |

# Application: homomorphic image processing

# Homomorphic image processing

- FFT - Hadamard product - iFFT: standard image processing pipeline.
- We performed a homomorphic evaluation of this pipeline
  - matrix for Hadamard product was also encrypted,
  - encrypted inputs were complex numbers
    - FFT and HAD inputs: 8 bits precision,
    - precision of roots of unity is variable - result to be within 32 bits precision.
- The whole operation is non-linear
  - cannot apply additive homomorphic schemes.

# Homomorphic image processing

- FFT - Hadamard product - iFFT: standard image processing pipeline.
- We performed a homomorphic evaluation of this pipeline
  - matrix for Hadamard product was also encrypted,
  - encrypted inputs were complex numbers
    - FFT and HAD inputs: 8-bits precision,
    - precision of roots of unity is variable - result to be within 32-bits precision.
- The whole operation is non-linear
  - cannot apply additive homomorphic schemes.

# Homomorphic image processing

- FFT - Hadamard product - iFFT: standard image processing pipeline.
- We performed a homomorphic evaluation of this pipeline
  - matrix for Hadamard product was also encrypted,
  - encrypted inputs were complex numbers
    - ★ FFT and HAD inputs: 8-bits precision,
    - ★ precision of roots of unity is variable - result to be within 32-bits precision.
- The whole operation is non-linear
  - cannot apply additive homomorphic schemes.

# Homomorphic image processing

- FFT - Hadamard product - iFFT: standard image processing pipeline.
- We performed a homomorphic evaluation of this pipeline
  - matrix for Hadamard product was also encrypted,
  - encrypted inputs were complex numbers
    - ★ FFT and HAD inputs: 8-bits precision,
    - ★ precision of roots of unity is variable - result to be within 32-bits precision.
- The whole operation is non-linear
  - cannot apply additive homomorphic schemes.

# Homomorphic image processing

- We use mixed Fourier transform instead of FFT
  - tradeoff: *depth* vs. *number of scalar multiplications*,
  - greater depth means longer modulus chain in SHE scheme and hence slower.

# Homomorphic image processing

- Concrete parameters for the FFT-Hadamard-iFFT pipeline:

| | FFT $\mathcal{B} = 1$ | | $\mathcal{B} = \sqrt{n}$ | | NFT $\mathcal{B} = n$ | |
|---|---|---|---|---|---|---|
| | $\log_2 p$ | $\deg(\mathcal{R})$ | $\log_2 p$ | $\deg(\mathcal{R})$ | $\log_2 p$ | $\deg(\mathcal{R})$ |
| $n$ | $\geq$ | $\geq$ | $\geq$ | $\geq$ | $\geq$ | $\geq$ |
| 16 | 54 | 190 | 37 | 118 | 25 | 46 |
| 64 | 74 | 248 | 49 | 146 | 29 | 44 |
| 256 | 93 | 298 | 61 | 170 | 33 | 42 |
| 1024 | 112 | 340 | 72 | 190 | 37 | 40 |

# Homomorphic image processing

- Timing results (in sec) using HElib running on a 6-core cluster:

| $n$ | $\mathcal{B}$ | $\deg(\mathcal{R})$ | $\log_2 q$ | HElib Levels | Amortization Amount | CPU Time | Amortized Time |
|---|---|---|---|---|---|---|---|
| 16 | 1 | 32768 | 710 | 33 | 172 | 188 | 1.09 |
| 16 | 4 | 32768 | 451 | 19 | 277 | 147 | 0.53 |
| 16 | 16 | 16384 | 192 | 9 | 356 | 106 | 0.3 |
| 64 | 8 | 32768 | 622 | 30 | 224 | 1500 | 6.69 |
| 64 | 64 | 16384 | 192 | 10 | 372 | 1582 | 4.25 |
| 256 | 256 | 16384 | 278 | 11 | 390 | 34876 | 89.4 |

# Conclusion

# Conclusion

- We investigated the growth of coefficients and the degree for various encoding schemes.
- Proved the equivalence of scaled integer and fractional encodings for fixed-point numbers.
- Computed concrete bounds for regular circuits.
- Implemented homomorphic evaluation of FFT-Hadamard product-iFFT pipeline used in image processing.

# Conclusion

- We investigated the growth of coefficients and the degree for various encoding schemes.
- Proved the equivalence of scaled integer and fractional encodings for fixed-point numbers.
- Computed concrete bounds for regular circuits.
- Implemented homomorphic evaluation of FFT-Hadamard product-iFFT pipeline used in image processing.

# Conclusion

- We investigated the growth of coefficients and the degree for various encoding schemes.
- Proved the equivalence of scaled integer and fractional encodings for fixed-point numbers.
- Computed concrete bounds for regular circuits.
- Implemented homomorphic evaluation of FFT-Hadamard product-iFFT pipeline used in image processing.

# Conclusion

- We investigated the growth of coefficients and the degree for various encoding schemes.
- Proved the equivalence of scaled integer and fractional encodings for fixed-point numbers.
- Computed concrete bounds for regular circuits.
- Implemented homomorphic evaluation of FFT-Hadamard product-iFFT pipeline used in image processing.

# Future Work

- Improve the current bounds for coefficient growth.
- SIMD implementation of FFT-Hadamard product-iFFT pipeline for scaled integer encoding.

# Future Work

- Improve the current bounds for coefficient growth.
- SIMD implementation of FFT-Hadamard product-iFFT pipeline for scaled integer encoding.

Thank You!