Sparse-secret Ring-LWE in FHE: Is It Really Needed?

Ilia Iliashenko (joint work with Hao Chen, Kim Laine, Yongsoo Song)

Lattice Coding & Crypto Meeting, Royal Holloway

20 Nov

Learning with Errors (LWE)

$$\boldsymbol{b} = \boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{e}$$

$\boldsymbol{A} \in \mathbb{Z}_q^{n\times \mathrm{n}}$ is uniformly random, $\boldsymbol{s} \in \mathbb{Z}_q^n$ and $\boldsymbol{e} \in \mathbb{Z}_q^n$ is small.

Decision: distinguish between $(\boldsymbol{A}, \boldsymbol{b})$ and uniformly random $(\boldsymbol{M}, \boldsymbol{v})$.

Search: find $\boldsymbol{s}$.

Sample s and e coefficient-wise

$$\boldsymbol{b} = \boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{e}$$

$$\boldsymbol{s_0} \quad \boldsymbol{s_1} \quad \boldsymbol{\ldots} \quad \boldsymbol{s_{n-1}}$$

Uniformly random U_2 over $\{0,1\}^n$.
Uniformly random U_3 over $\{-1,0,1\}^n$.
Uniformly random U_q over $\mathbb{Z}_q^n$.
Discrete Gaussian $\mathcal{D}_q$ over $\mathbb{Z}_q^n$.

Hardness of LWE

$$\boldsymbol{b} = \boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{e}$$

$\boldsymbol{s_0}$	$\boldsymbol{s_1}$	…	$\boldsymbol{s_{n-1}}$

Uniformly random U_2 over $\{0,1\}^n$.
Uniformly random U_3 over $\{-1,0,1\}^n$.
Uniformly random U_q over $\mathbb{Z}_q^n$.
Discrete Gaussian $\mathcal{D}_q$ over $\mathbb{Z}_q^n$.

$\boldsymbol{s} \leftarrow U_q$, or U_2, or $\mathcal{D}_q$
$\boldsymbol{e} \leftarrow \mathcal{D}_q$ with $\sigma \in \Omega(\sqrt{n})$

$\longrightarrow$ LWE is as hard as classical lattice problems (GapSVP, DGS)

Sparse-secret LWE

$$\boldsymbol{b} = \boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{e}$$

$\boldsymbol{s_0}$	$\boldsymbol{s_1}$	…	$\boldsymbol{s_{n-1}}$

Uniformly random U_2 over $\{0,1\}^n$.
Uniformly random U_3 over $\{-1,0,1\}^n$.
Uniformly random U_q over $\mathbb{Z}_q^n$.
Discrete Gaussian $\mathcal{D}_q$ over $\mathbb{Z}_q^n$.

$\boldsymbol{s} \leftarrow U_3(h)$: $wt(\boldsymbol{s}) = h$
$\boldsymbol{e} \leftarrow \mathcal{D}_q$ $\longrightarrow$???

Ring-LWE

$$\boldsymbol{b} = \begin{bmatrix} \boldsymbol{a_0} & -\boldsymbol{a_{n-1}} & \cdots \\ \boldsymbol{a_1} & \boldsymbol{a_0} & \\ \boldsymbol{a_2} & \boldsymbol{a_1} & \\ \cdots & \cdots & \\ \boldsymbol{a_{n-1}} & \boldsymbol{a_{n-2}} & \end{bmatrix} \cdot \boldsymbol{s} + \boldsymbol{e}$$

Ring-LWE

$$b \quad = \quad a \quad \cdot \quad s \quad + \quad e$$

$a, b, s, e \in R_q = \mathbb{Z}[X]/(q, X^n + 1)$ (n must be a power of two)

Hardness of Ring-LWE

$$b \quad = \quad a \quad \cdot \quad s \quad + \quad e$$

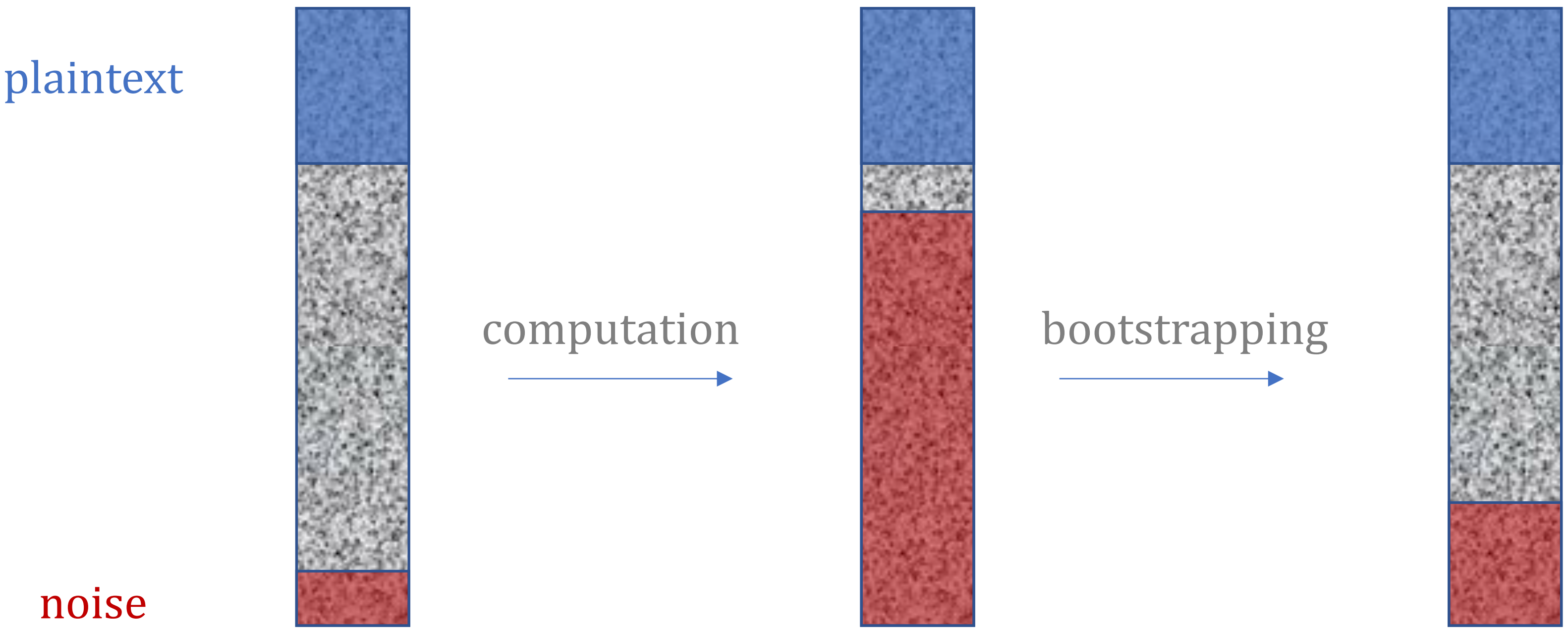
$a, b, s, e \in R_q = \mathbb{Z}[X]/(q, X^n + 1)$ (n must be a power of two)

$s \leftarrow U_q$ or $\mathcal{D}_q$ $\longrightarrow$ Ring-LWE is at least as hard as SIVP

Attacks on sparse-secret LWE

Albrecht, Eurocrypt'17
Albrecht et al., Asiacrypt '17
Cheon et al., IEEE Access'19
Curtis and Player, WAHC'19
Cheon and Son, WAHC'19
...

Efficient FHE schemes need sparse secrets for bootstrapping



Bootstrapping performs decryption homomorphically.

Efficient FHE schemes need sparse secrets for bootstrapping

Multiplicative depth of bootstrapping depends on $wt(s)$:

- FV: $\log\big(wt(s)\big) + \log(\log\big(wt(s)\big) + \log t)$
- BGV: $\log\big(wt(s)\big) + \log t$

Reference: Chen and Han, Eurocrypt'18

TFHE bootstrapping <u>does not</u> have this dependency.

Approximate HE

$$ct(m_1) \star ct(m_2) = ct(\simeq m_1 \odot m_2)$$

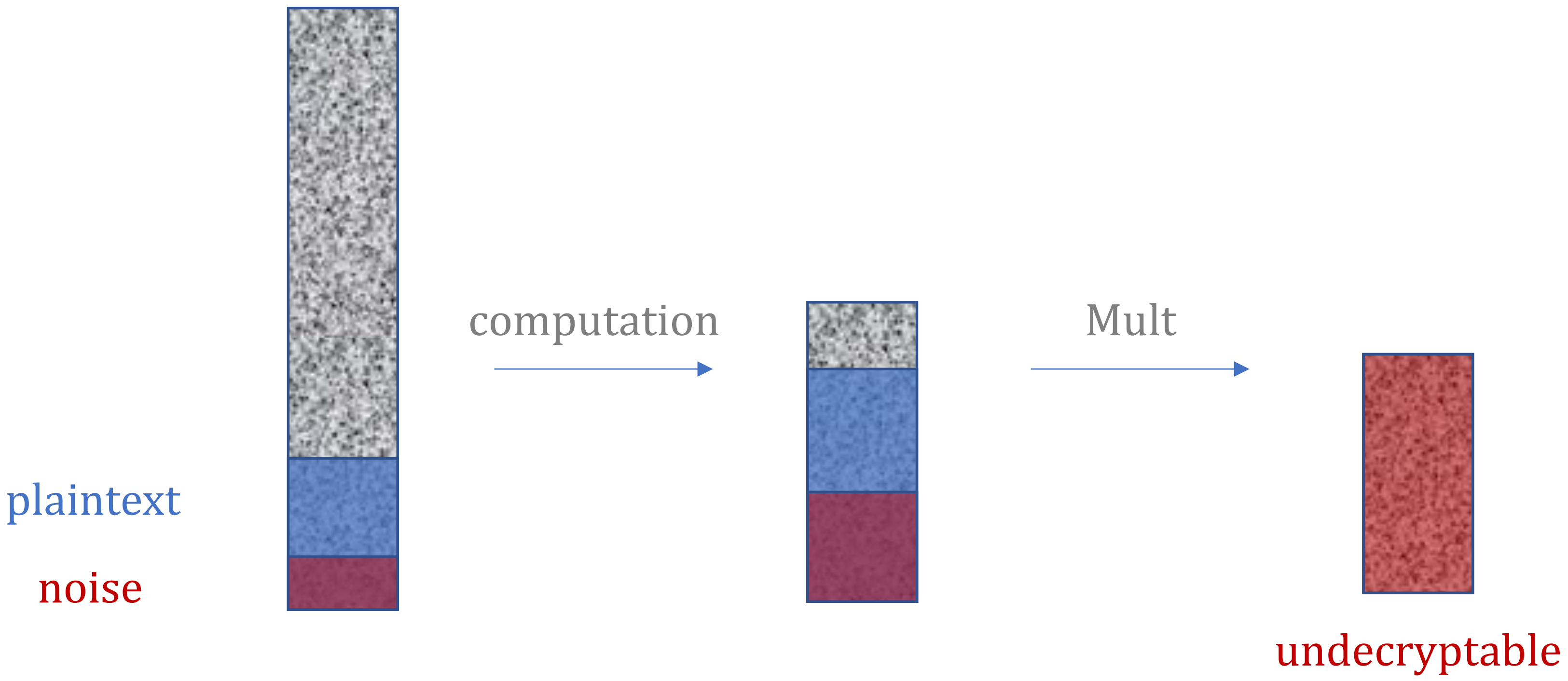
Approximate HE (HEAAN/CKKS)

Idea: consider ciphertext noise as a part of a message.

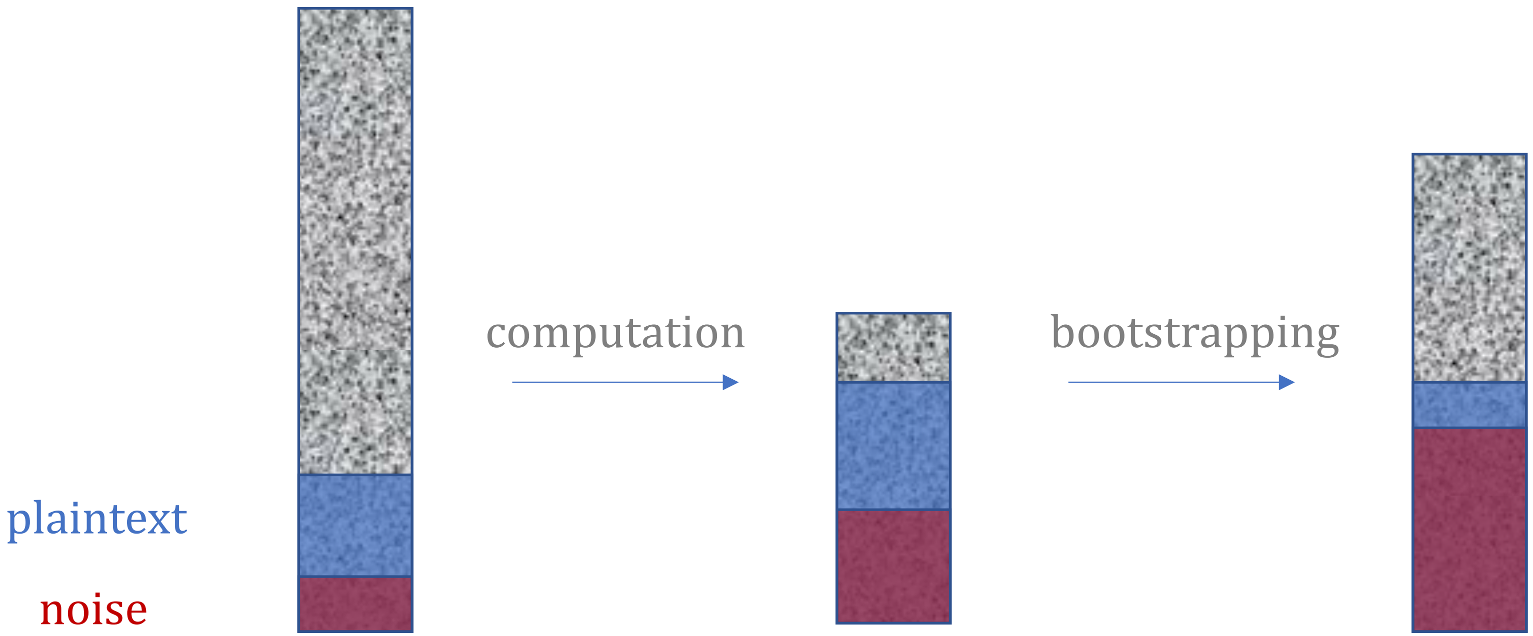
$$\mathrm{Decrypt}(ct) = m + e \simeq m.$$

Reference: Cheon et al., Asiacrypt'17

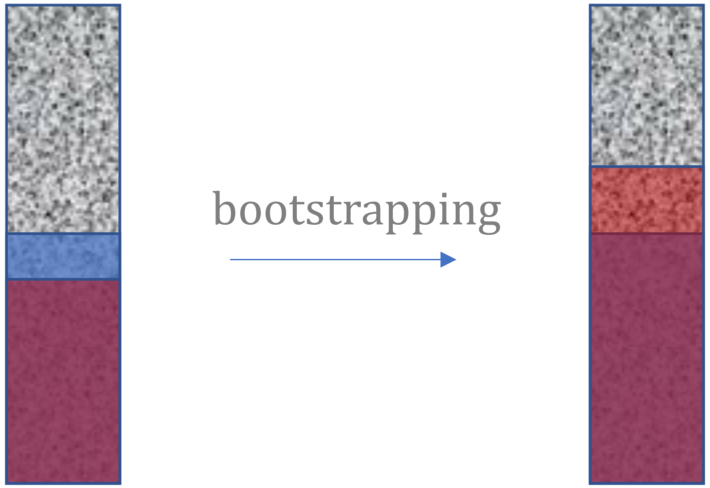
HEAAN bootstrapping



HEAAN bootstrapping

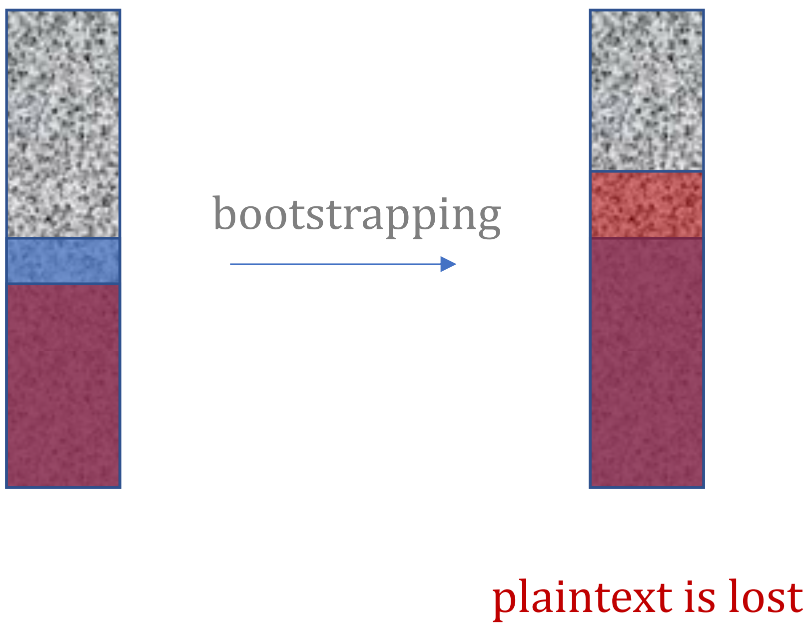


HEAAN “bootstrapping”



plaintext is lost

HEAAN “bootstrapping”



Correctness of Homomorphic Encryption

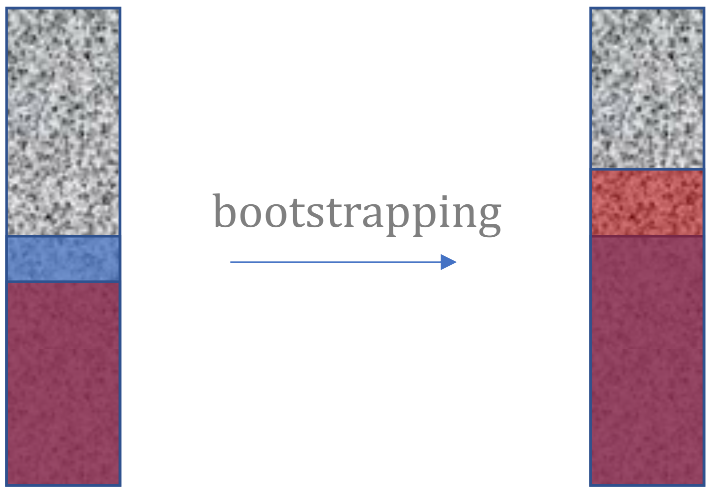
HE scheme E is correct for a circuit C if for any plaintexts $\pi_1, \dots, \pi_k$ it holds:
If $ct = \text{Evaluate}_E(C, \text{Enc}(\pi_1), \dots, \text{Enc}(\pi_k))$,
then $\text{Dec}_E(ct) = C(\pi_1, \dots, \pi_k)$.

Bootstrappable Encryption Scheme

Let C_E be the set of circuits that E can compactly and correctly evaluate. We say that E is bootstrappable with the respect to gate Γ if

$$Dec_E(\Gamma) \subseteq C_E.$$

HEAAN “bootstrapping”



plaintext is lost

~~**Correctness of Homomorphic Encryption**~~

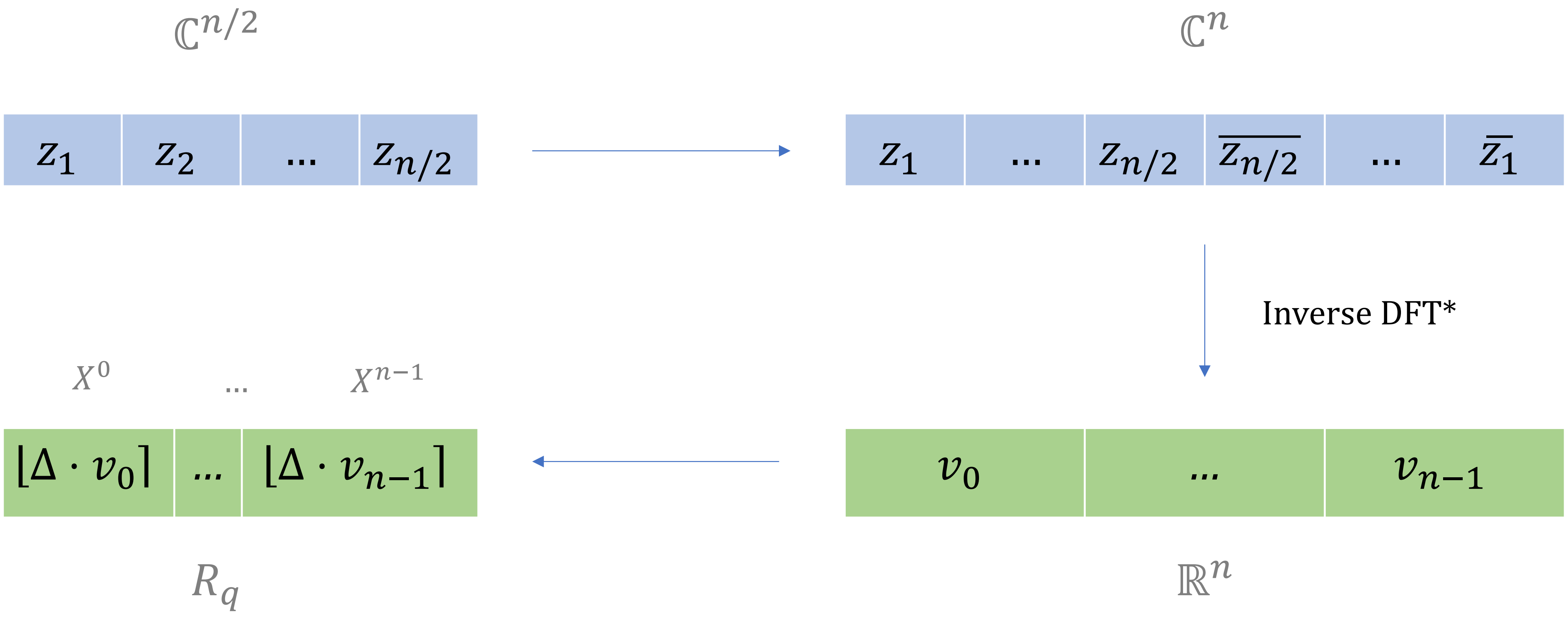
~~HE scheme E is correct for a circuit C if for any plaintexts $\pi_1, \ldots, \pi_k$ it holds:~~
~~If $ct = \text{Evaluate}_E(C, \text{Enc}(\pi_1), \ldots, \text{Enc}(\pi_k))$,~~
~~then $\text{Dec}_E(ct) = C(\pi_1, \ldots, \pi_k)$.~~

~~**Bootstrappable Encryption Scheme**~~

~~Let C_E be the set of circuits that E can compactly and correctly evaluate. We say that E is bootstrappable with the respect to gate Γ if~~

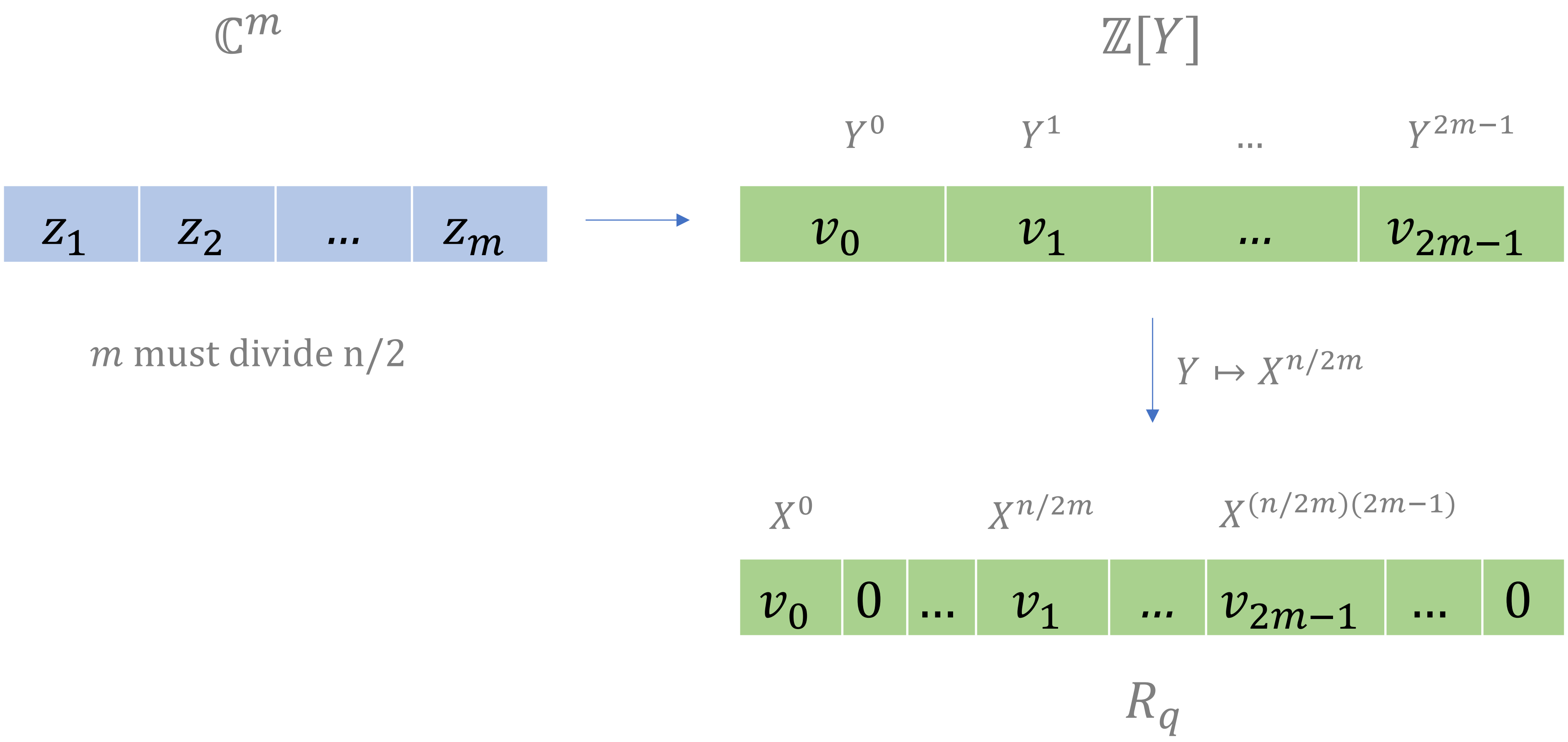
$$Dec_E(\Gamma) \subseteq C_E.$$

HEAAN works with complex vectors



*with primitive roots of unity

How to encode less than $n/2$ values?



Decoding

X^0		X^{n-1}
a_0	…	a_{n-1}

$\xrightarrow{\text{computation}}$

X^0		X^{n-1}
a'_0	…	a'_{n-1}

$\big\downarrow$ DFT*

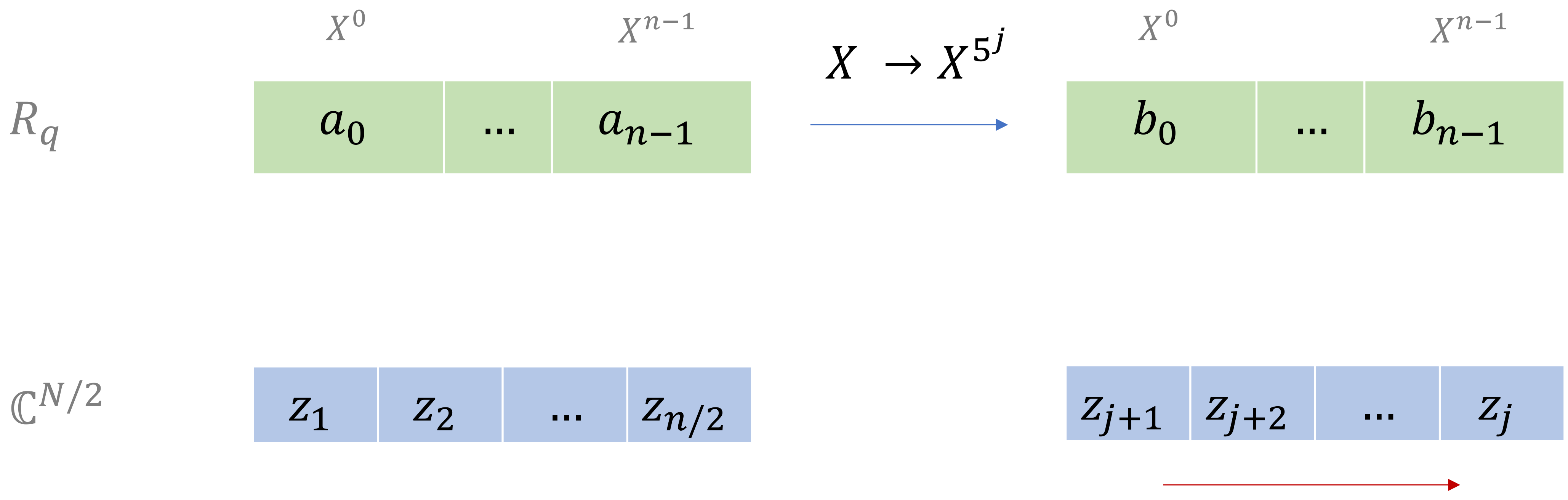
$\Delta' \cdot \boldsymbol{z}_{\mathbf{1}} + e_1$	…	$\Delta' \cdot \boldsymbol{z}_{\boldsymbol{n/2}} + e_{n/2}$

$\xleftarrow{1/\Delta'}$

$\approx z_1$	…	$\approx z_{n/2}$

*with primitive roots of unity

Rotation of encoded vectors



Rotation of encoded vectors

	X^0		X^{n-1}		X^0		X^{n-1}
R_q	a_0	…	a_{n-1}	$\xrightarrow{X \to X^{5^m}}$	b_0	…	b_{n-1}

$\mathbb{C}^m$	z_1	z_2	…	z_m		z_1	z_2	…	z_m

Rotations by km slots are automorphisms of R fixing $R' = \mathbb{Z}\left[X^{\frac{n}{2m}}\right]/(q, X^n + 1)$, $R' \subset R$.

Key generation, encryption and decryption

Key generation

$$-\left(\overset{\mathcal{D}_q}{\downarrow \atop e} + \overset{U_3(h)}{\downarrow \atop s} \cdot \overset{U_q}{\downarrow \atop a}\right) = b$$

secret key s

public key (a, b)

Key generation, encryption and decryption

Key generation

$$-\,(\overset{\mathcal{D}_q}{e} + \underset{\text{secret key}}{\overset{U_3(h)}{s}} \cdot \overset{U_q}{a}) = b$$

public key

Encryption

Given a public key pk and an encoding $m \in R_q$ compute

$$\underset{c_0}{m + \overset{U_3}{u} \cdot pk_b + \overset{\mathcal{D}_q}{e_0}} \qquad \underset{c_1}{u \cdot pk_a + \overset{\mathcal{D}_q}{e_1}}$$

Key generation, encryption and decryption

Key generation

$$-\,(\overset{\mathcal{D}_q}{e} + \overset{U_3(h)}{s} \cdot \overset{U_q}{a}) = b$$

s: secret key

a, b: public key

Encryption

Given a public key pk and an encoding $m \in R_q$ compute

$$c_0 = m + \overset{U_3}{u} \cdot pk_b + \overset{\mathcal{D}_q}{e_0} \qquad c_1 = u \cdot pk_a + \overset{\mathcal{D}_q}{e_1}$$

Decryption

Given a secret key s and a ciphertext $ct = (c_0, c_1)$ compute

$$[ct(s)]_q = c_0 + c_1 \cdot s \bmod q = m + \underbrace{e}_{\text{noise}}$$

Rescaling

Let Δ divide q.

	R_q		$R_{q/\Delta}$
	c_0, c_1	$\longrightarrow$	$\left\lfloor \frac{c_0}{\Delta} \right\rceil, \left\lfloor \frac{c_1}{\Delta} \right\rceil$
$\mathbb{C}^{n/2}$	$\Delta^2 \cdot z_1$ $\Delta^2 \cdot z_2$ … $\Delta^2 \cdot z_{n/2}$	$\longrightarrow$	$\Delta \cdot z_1$ $\Delta \cdot z_2$ … $\Delta \cdot z_{n/2}$

HEAAN bootstrapping

	Ciphertext	Plaintext	Cleartext vector
Input	$ct \in R_q^2$	$m\left(X^{\frac{n}{2m}}\right) = [ct(s)]_q$	$\boldsymbol{z_0}$ | **...** | $\boldsymbol{z_{m-1}}$
Output	$ct' \in R_{q'}^2, q' > q$	$\simeq m(X^{\frac{n}{2m}})$	$\boldsymbol{z_0}$ | **...** | $\boldsymbol{z_{m-1}}$

CKKS bootstrapping

	Ciphertext	Plaintext	Cleartext vector		
Input	$ct \in R_q^2$	$m\left(X^{\frac{n}{2m}}\right) = ct(s) - I(X) \cdot q$	$\boldsymbol{z_0}$	**...**	$\boldsymbol{z_{m-1}}$
Output	$ct' \in R_{q'}^2, q' > q$	$\simeq m(X^{\frac{n}{2m}})$	$\boldsymbol{z_0}$	**...**	$\boldsymbol{z_{m-1}}$

CKKS bootstrapping

	Ciphertext	Plaintext	Cleartext vector
Input	$ct \in R_q^2$	$m\left(X^{\frac{n}{2m}}\right) = ct(s) - I(X) \cdot q$	$\boldsymbol{z_0}$ ... $\boldsymbol{z_{m-1}}$
ModRaise	$ct \in R_{Q_0}^2, Q_0 > q$	$\left[m\left(X^{\frac{n}{2m}}\right) + I(X) \cdot q\right]_{Q_0}$	
Output	$ct' \in R_{q'}^2, q' > q$	$\simeq m(X^{\frac{n}{2m}})$	$\boldsymbol{z_0}$ ... $\boldsymbol{z_{m-1}}$

CKKS bootstrapping

	Ciphertext	Plaintext	Cleartext vector
Input	$ct \in R_q^2$	$m\left(X^{\frac{n}{2m}}\right) = ct(s) - I(X) \cdot q$	z_0 | ... | z_{m-1}
ModRaise	$ct \in R_{Q_0}^2, Q_0 > q$	$\left[m\left(X^{\frac{n}{2m}}\right) + I(X) \cdot q\right]_{Q_0}$	
SubSum	$ct_1 \in R_{Q_1}^2$	$\simeq \left[m\left(X^{\frac{n}{2m}}\right) + I\left(X^{\frac{n}{2m}}\right) \cdot q\right]_{Q_1}$	
Output	$ct' \in R_{q'}^2, q' > q$	$\simeq m(X^{\frac{n}{2m}})$	z_0 | ... | z_{m-1}

CKKS bootstrapping

	Ciphertext	Plaintext	Cleartext vector
Input	$ct \in R_q^2$	$m\left(X^{\frac{n}{2m}}\right) = ct(s) - I(X) \cdot q$	z_0 ... z_{m-1}
ModRaise	$ct \in R_{Q_0}^2, Q_0 > q$	$\left[m\left(X^{\frac{n}{2m}}\right) + I(X) \cdot q\right]_{Q_0}$	
SubSum	$ct_1 \in R_{Q_1}^2$	$\simeq \left[t(X^{\frac{n}{2m}})\right]_{Q_1}$	
Output	$ct' \in R_{q'}^2, q' > q$	$\simeq m(X^{\frac{n}{2m}})$	z_0 ... z_{m-1}

CKKS bootstrapping

	Ciphertext	Plaintext	Cleartext vector
Input	$ct \in R_q^2$	$m\left(X^{\frac{n}{2m}}\right) = ct(s) - I(X) \cdot q$	$\mathbf{z_0}$ | … | $\mathbf{z_{m-1}}$
ModRaise	$ct \in R_{Q_0}^2, Q_0 > q$	$\left[m\left(X^{\frac{n}{2m}}\right) + I(X) \cdot q\right]_{Q_0}$	
SubSum	$ct_1 \in R_{Q_1}^2$	$\simeq \left[t(X^{\frac{n}{2m}})\right]_{Q_1}$	
CoefToSlot (inverse DFT)	$ct_2 \in R_{Q_2}^2$		$\mathbf{t_0}$ | … | $\mathbf{t_{2m-1}}$
Output	$ct' \in R_{q'}^2, q' > q$	$\simeq m(X^{\frac{n}{2m}})$	$\mathbf{z_0}$ | … | $\mathbf{z_{m-1}}$

CKKS bootstrapping

	Ciphertext	Plaintext	Cleartext vector
Input	$ct \in R_q^2$	$m\left(X^{\frac{n}{2m}}\right) = ct(s) - I(X) \cdot q$	$\boldsymbol{z_0}$ | ... | $\boldsymbol{z_{m-1}}$
ModRaise	$ct \in R_{Q_0}^2, Q_0 > q$	$\left[m\left(X^{\frac{n}{2m}}\right) + I(X) \cdot q\right]_{Q_0}$	
SubSum	$ct_1 \in R_{Q_1}^2$	$\simeq \left[t(X^{\frac{n}{2m}})\right]_{Q_1}$	
CoefToSlot (inverse DFT)	$ct_2 \in R_{Q_2}^2$		$\boldsymbol{t_0}$ | ... | $\boldsymbol{t_{2m-1}}$
Mod q	$ct_3 \in R_{Q_3}^2$		$\boldsymbol{m_0}$ | ... | $\boldsymbol{m_{2m-1}}$
Output	$ct' \in R_{q'}^2, q' > q$	$\simeq m(X^{\frac{n}{2m}})$	$\boldsymbol{z_0}$ | ... | $\boldsymbol{z_{m-1}}$

CKKS bootstrapping

	Ciphertext	Plaintext	Cleartext vector
Input	$ct \in R_q^2$	$m\left(X^{\frac{n}{2m}}\right) = ct(s) - I(X) \cdot q$	z_0 | ... | z_{m-1}
ModRaise	$ct \in R_{Q_0}^2, Q_0 > q$	$\left[m\left(X^{\frac{n}{2m}}\right) + I(X) \cdot q\right]_{Q_0}$	
SubSum	$ct_1 \in R_{Q_1}^2$	$\simeq \left[t(X^{\frac{n}{2m}})\right]_{Q_1}$	
CoefToSlot (inverse DFT)	$ct_2 \in R_{Q_2}^2$		t_0 | ... | t_{2m-1}
Mod q	$ct_3 \in R_{Q_3}^2$		m_0 | ... | m_{2m-1}
SlotToCoef (DFT)	$ct_4 \in R_{Q_4}^2$	$\simeq \left[m(X^{\frac{n}{2m}})\right]_{Q_4}$	
Output	$ct' \in R_{q'}^2, q' > q$	$\simeq m(X^{\frac{n}{2m}})$	z_0 | ... | z_{m-1}

CKKS bootstrapping

	Ciphertext	Plaintext	Cleartext vector
Input	$ct \in R_q^2$	$m\left(X^{\frac{n}{2m}}\right) = ct(s) - I(X) \cdot q$	$\boldsymbol{z_0}$ … $\boldsymbol{z_{m-1}}$
ModRaise	$ct \in R_{Q_0}^2, Q_0 > q$	$\left[m\left(X^{\frac{n}{2m}}\right) + I(X) \cdot q\right]_{Q_0}$	
SubSum	$ct_1 \in R_{Q_1}^2$	$\simeq \left[t(X^{\frac{n}{2m}})\right]_{Q_1}$	
CoefToSlot (inverse DFT)	$ct_2 \in R_{Q_2}^2$		$\boldsymbol{t_0}$ … $\boldsymbol{t_{2m-1}}$
Mod q	$ct_3 \in R_{Q_3}^2$		$\boldsymbol{m_0}$ … $\boldsymbol{m_{2m-1}}$
SlotToCoef (DFT)	$ct_4 \in R_{Q_4}^2$	$\simeq \left[m(X^{\frac{n}{2m}})\right]_{Q_4}$	
Output	$ct' \in R_{q'}^2, q' = Q_4$	$\simeq m(X^{\frac{n}{2m}})$	$\boldsymbol{z_0}$ … $\boldsymbol{z_{m-1}}$

SubSum

SubSum computes $\mathrm{Tr}: R \to R'$, where $[R' : \mathbb{Z}] = 2m$.

$$ct \longrightarrow \sum_{i=0}^{\frac{n}{2m}-1} \mathrm{Rot}(ct, im)$$

$$\left[m\left(X^{\frac{n}{2m}}\right) + I(X) \cdot q\right]_{Q_0} \longrightarrow \simeq \left[m\left(X^{\frac{n}{2m}}\right) + I\left(X^{\frac{n}{2m}}\right) \cdot q\right]_{Q_1}$$

z_0	...	z_{m-1}

$\longrightarrow$

$\frac{n}{2m} z_0$	...	$\frac{n}{2m} z_{m-1}$

CoefToSlot and SlotToCoef

CoefToSlot = Encoding
done homomorphically

SlotToCoef = Decoding
done homomorphically

CoefToSlot and SlotToCoef

CoefToSlot = Encoding
done homomorphically

SlotToCoef = Decoding
done homomorphically

$\mathbf{\Sigma}$ is the canonical embedding matrix (DFT with $4m$-th primitive roots of unity)

$\mathbf{z} \mapsto \mathbf{t} = \mathbf{\Sigma}^{-1} \cdot \mathbf{z}$

$\mathbf{t} \mapsto \mathbf{z} = \mathbf{\Sigma} \cdot \mathbf{t}$

CoefToSlot and SlotToCoef

CoefToSlot = Encoding
done homomorphically

SlotToCoef = Decoding
done homomorphically

$\mathbf{\Sigma}$ is the canonical embedding matrix (DFT with $4m$-th primitive roots of unity)

$$\mathbf{z} \mapsto \mathbf{t} = \mathbf{\Sigma}^{-1} \cdot \mathbf{z} = \boldsymbol{L}_1 \cdot \ldots \cdot \boldsymbol{L}_l \cdot \mathbf{z}$$

L_i's are sparser than M.

$$\mathbf{t} \mapsto \mathbf{z} = \mathbf{\Sigma} \cdot \mathbf{t} = \boldsymbol{L'}_1 \cdot \ldots \cdot \boldsymbol{L'}_{l'} \cdot \mathbf{t}$$

The columns of $\boldsymbol{L}_i$'s need to be encoded into the plaintext space.

CoefToSlot and SlotToCoef

CoefToSlot = Encoding
done homomorphically

SlotToCoef = Decoding
done homomorphically

$\boldsymbol{\Sigma}$ is the canonical embedding matrix (DFT with $4m$-th primitive roots of unity)

$$\mathbf{z} \mapsto \mathbf{t} = \boldsymbol{\Sigma}^{-1} \cdot \mathbf{z} = \boldsymbol{L}_1 \cdot \ldots \cdot \boldsymbol{L}_l \cdot \mathbf{z}$$

L_i's are sparser than M.

$$\mathbf{t} \mapsto \mathbf{z} = \boldsymbol{\Sigma} \cdot \mathbf{t} = \boldsymbol{L'}_1 \cdot \ldots \cdot \boldsymbol{L'}_{l'} \cdot \mathbf{t}$$

The columns of $\boldsymbol{L}_i$'s need to be encoded into the plaintext space.

CoefToSlot $ct_1 \in R_{Q_1}^2$

SlotToCoef $ct_3 \in R_{Q_3}^2$

Since $Q_1 > Q_3$, homomorphic operations in CoefToSlot are heavier than those of SlotToCoeff.
Thus, use more FFT in CoefToSlot ($l > l'$).

Mod q

$$[ct(s)]_{Q_0} = m\left(X^{\frac{n}{2m}}\right) + I(X) \cdot q,$$

Mod q

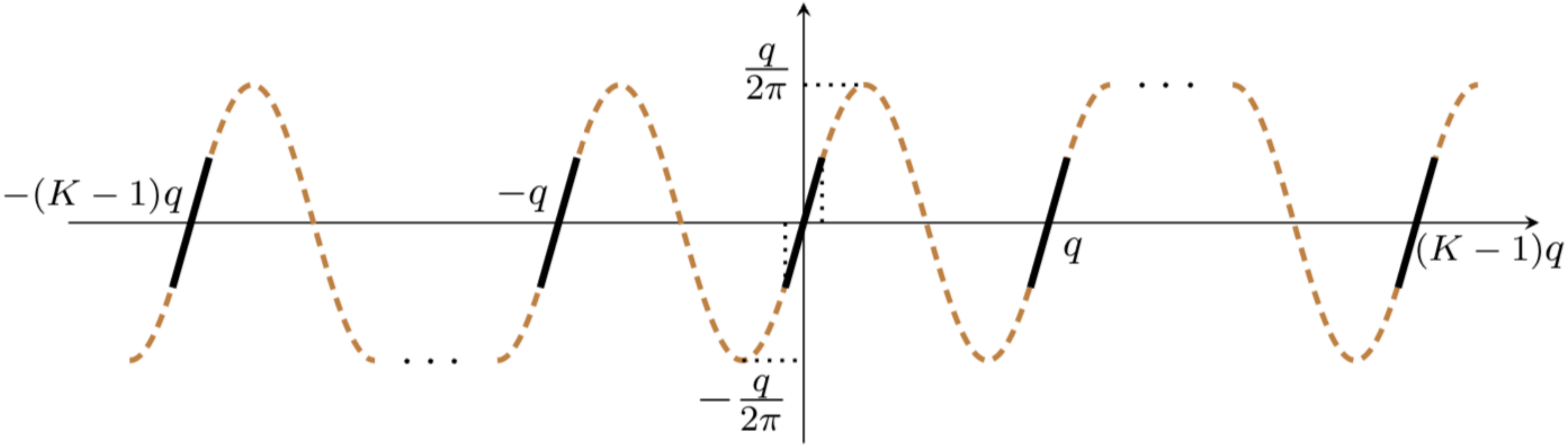
$$[ct(s)]_{Q_0} = m\left(X^{\frac{n}{2m}}\right) + I(X) \cdot q, \qquad |I(X)|_\infty < K$$

Mod q

$$[ct(s)]_{Q_0} = m\left(X^{\frac{n}{2m}}\right) + I(X) \cdot q, \qquad |I(X)|_\infty < K \leq 1 + wt(s)/2$$

Mod q

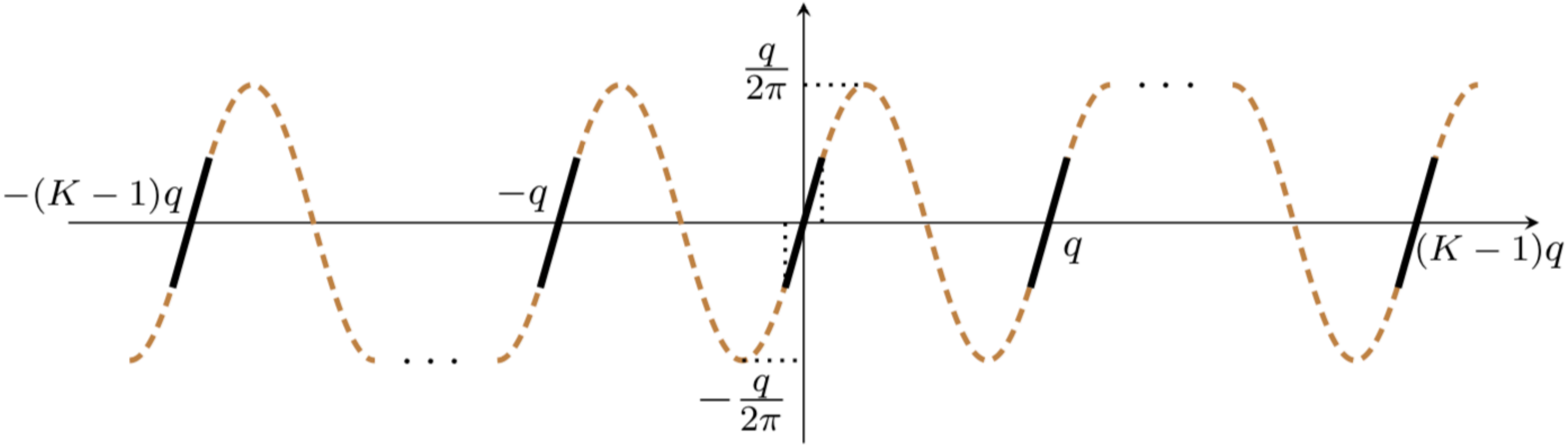
$$[ct(s)]_{Q_0} = m\left(X^{\frac{n}{2m}}\right) + I(X) \cdot q, \qquad |I(X)|_\infty < K \leq 1 + wt(s)/2$$



$$[x]_q \simeq \frac{q}{2\pi}\sin\left(\frac{2\pi x}{q}\right), \qquad x \in (-Kq, Kq)$$

Mod q

$$[ct(s)]_{Q_0} = m\left(X^{\frac{n}{2m}}\right) + I(X)\cdot q, \qquad |I(X)|_\infty < K \le 1 + wt(s)/2$$



$$[x]_q \simeq \frac{q}{2\pi}\sin\left(\frac{2\pi x}{q}\right) = \frac{q}{2\pi}\cos\left(\frac{2\pi x}{q} - \frac{\pi}{2}\right), \qquad x \in (-Kq, Kq)$$

Sine should be approximated by a polynomial

Previous works:

- Cheon et al., Eurocrypt'18: Taylor + double-angle formula for sine
- Chen et al., Eurocrypt'19: Chebyshev
- Han-Ki, eprint'19: Hermite + Chebyshev nodes + double-angle formula for cosine

Sine should be approximated by a polynomial

Previous works:

- Cheon et al., Eurocrypt'18: Taylor + double-angle formula for sine
- Chen et al., Eurocrypt'19: Chebyshev
- Han-Ki, eprint'19: Hermite + Chebyshev nodes + double-angle formula for cosine

The above results assume that

- the secret key s is sparse, $wt(s) = 64$,
- and, thus, $K \leq 12$ with high probability.

Sine should be approximated by a polynomial

Previous works:

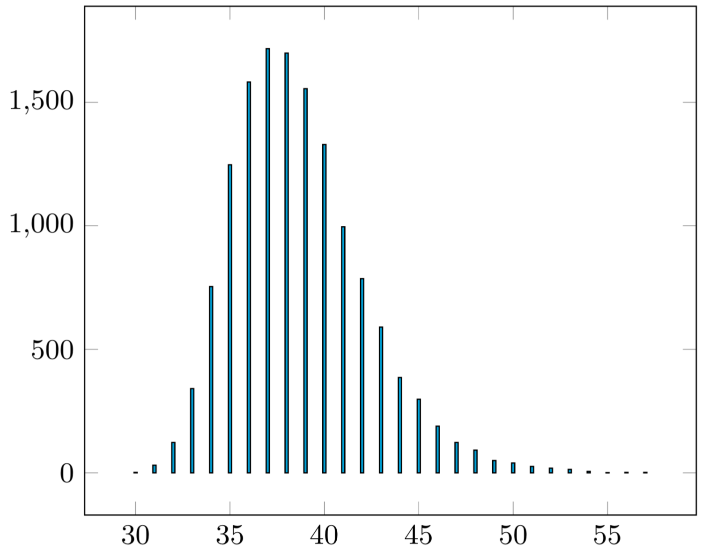
• Cheon et al., Eurocrypt'18:	Taylor + double-angle formula for sine
• Chen et al., Eurocrypt'19:	Chebyshev
• Han-Ki, eprint'19:	Hermite + Chebyshev nodes + double-angle formula for cosine

The above results assume that

- the secret key s is sparse, $wt(s) = 64$,
- and, thus, $K \leq 12$ with high probability.

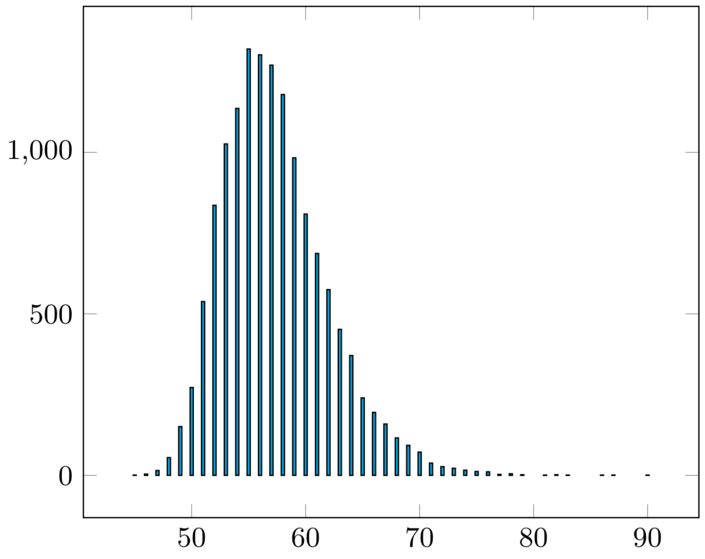
What happens when secret keys are dense?

Distribution of K when secret keys are dense



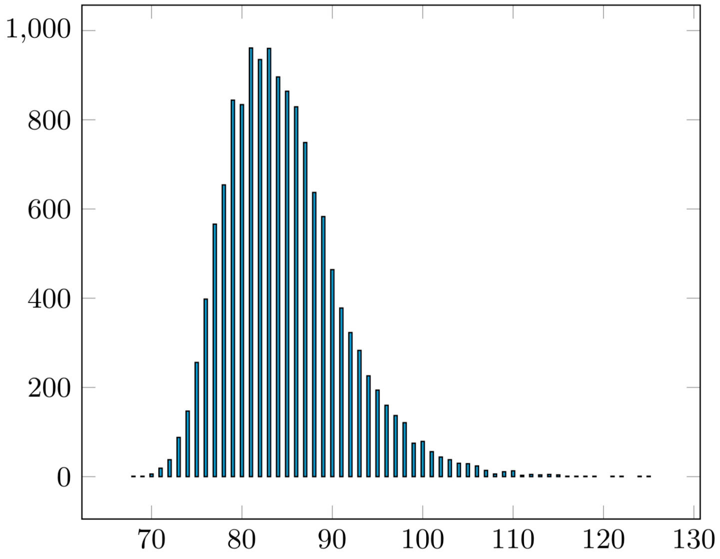
$n = 2048$

$\max K = 57$



$n = 4096$

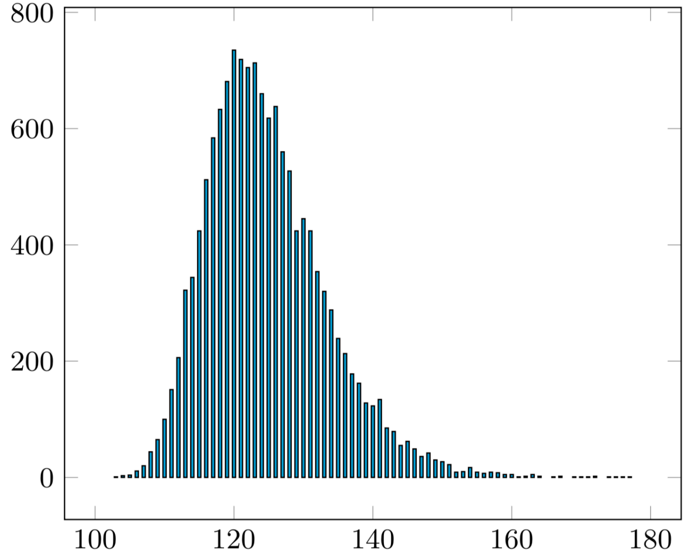
$\max K = 90$



$n = 8192$

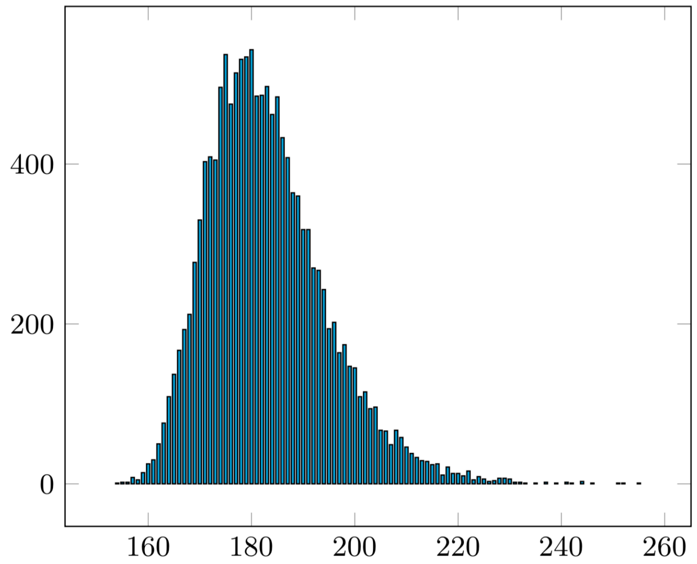
$\max K = 125$

Distribution of K when secret keys are dense



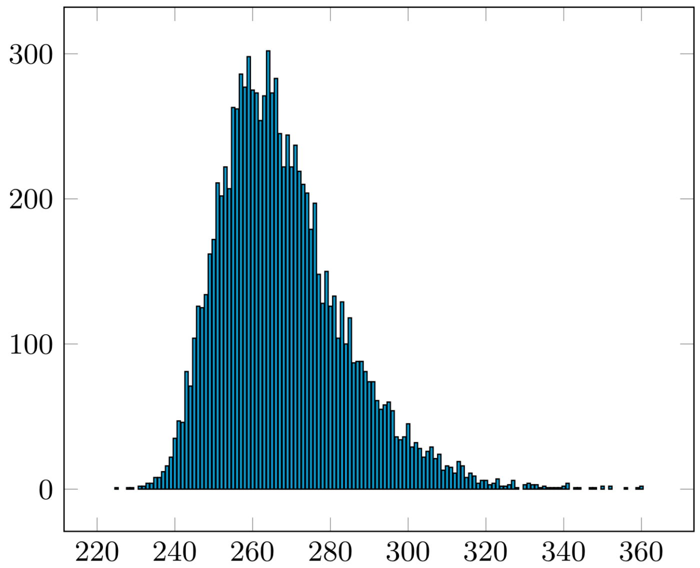
$n = 16384$

$\max K = 177$



$n = 32768$

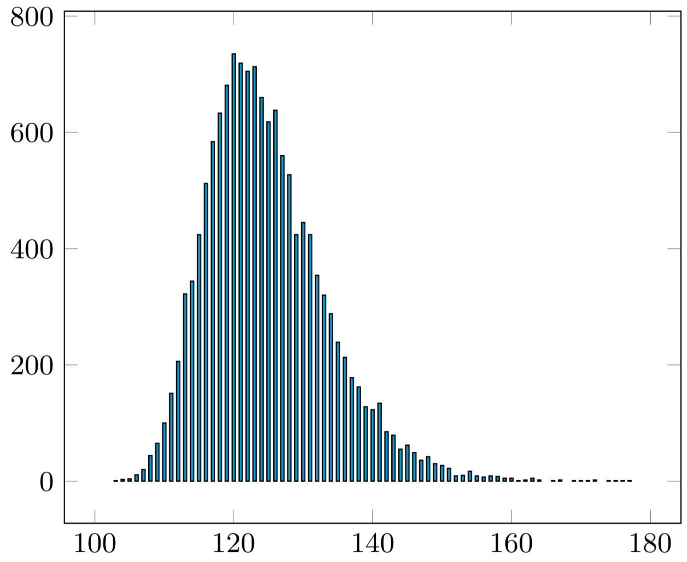
$\max K = 255$



$n = 65536$

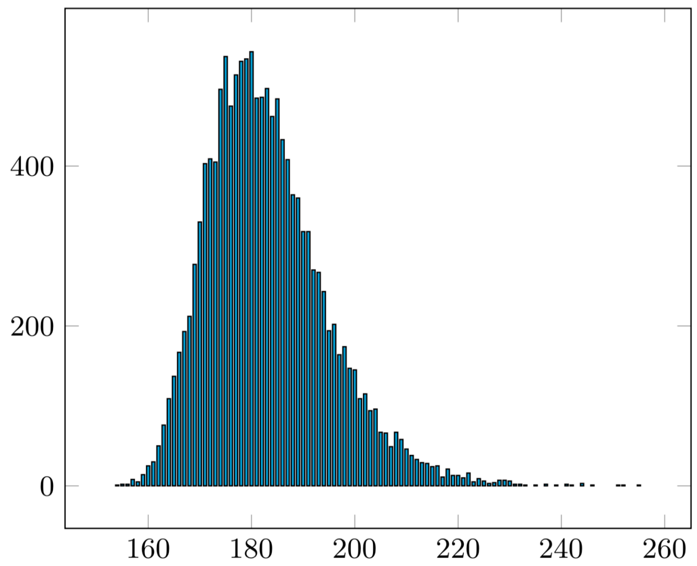
$\max K = 360$

Distribution of K when secret keys are dense



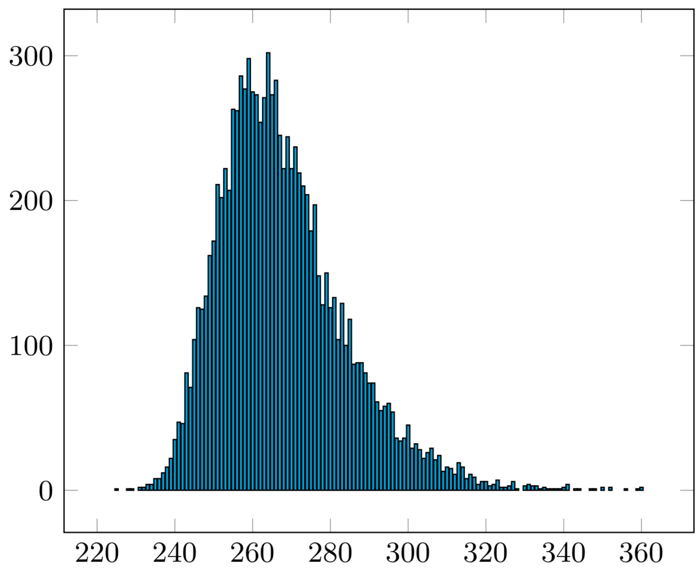
$n = 16384$

$\max K = 177$



$n = 32768$

$\max K = 255$



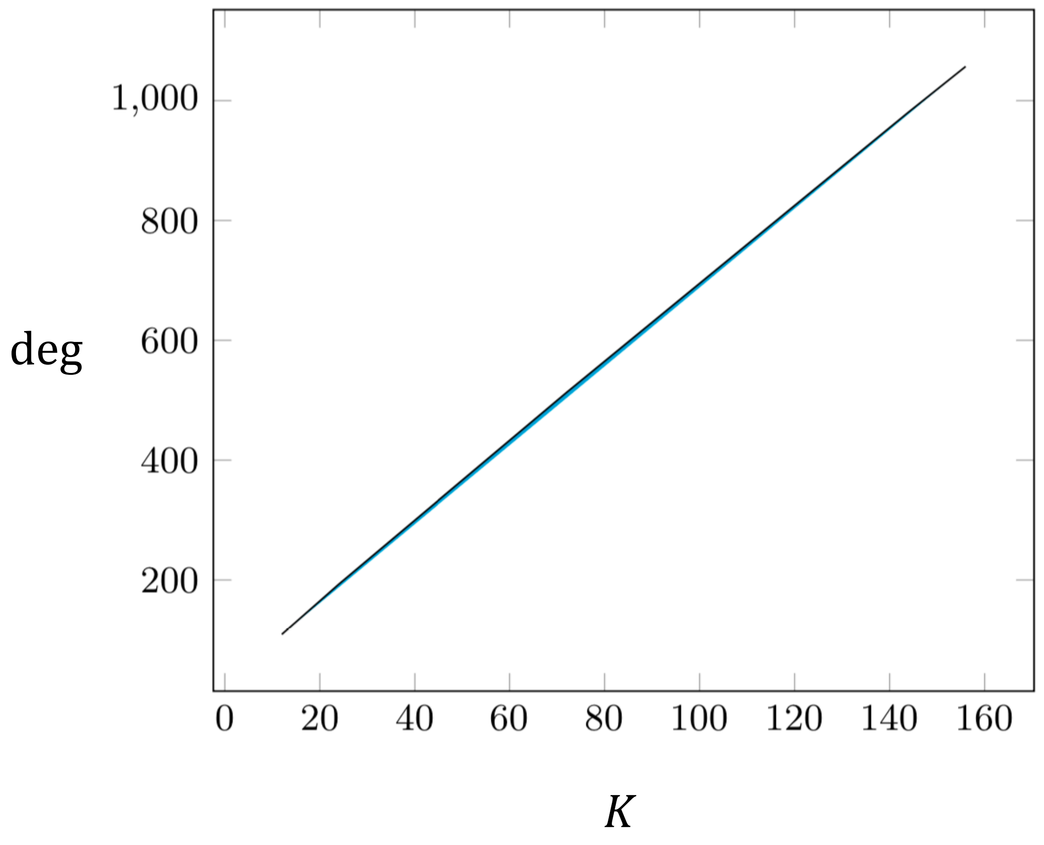
$n = 65536$

$\max K = 360$

Similar to the extreme value distribution.

Chebyshev approximation grows linearly with K

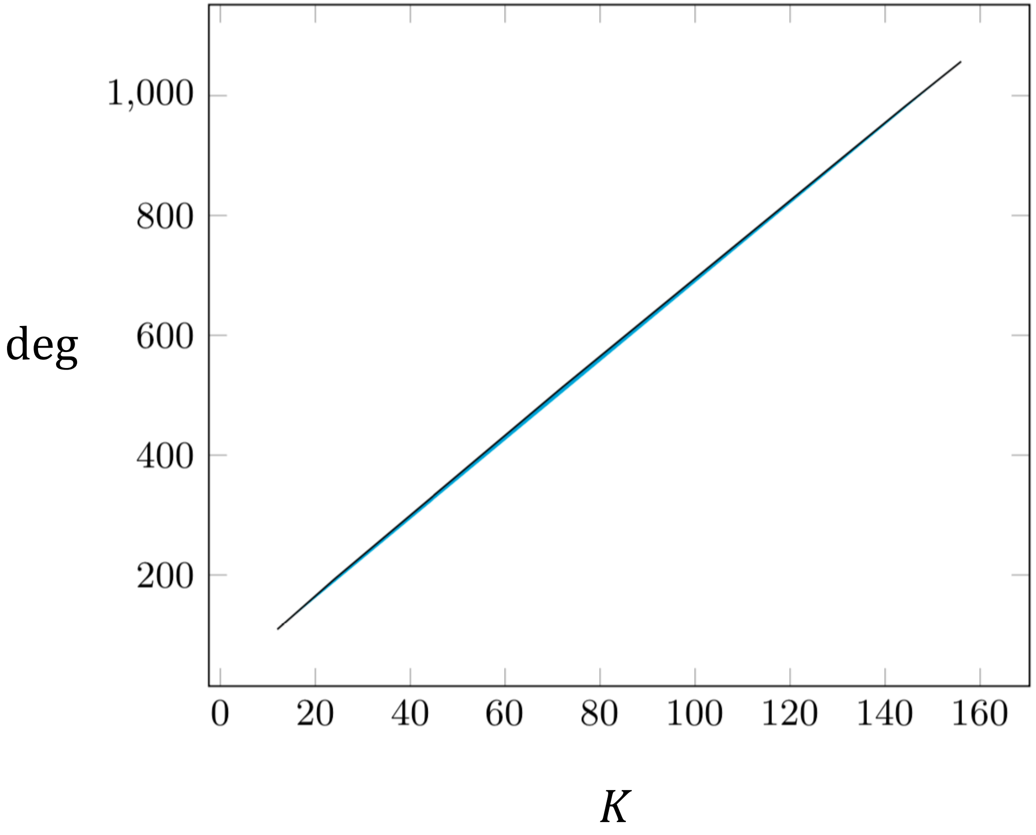
Approximation error: 10^{-12}



$\deg \simeq 7K + 25$

Chebyshev approximation grows linearly with K

Approximation error: 10^{-12}



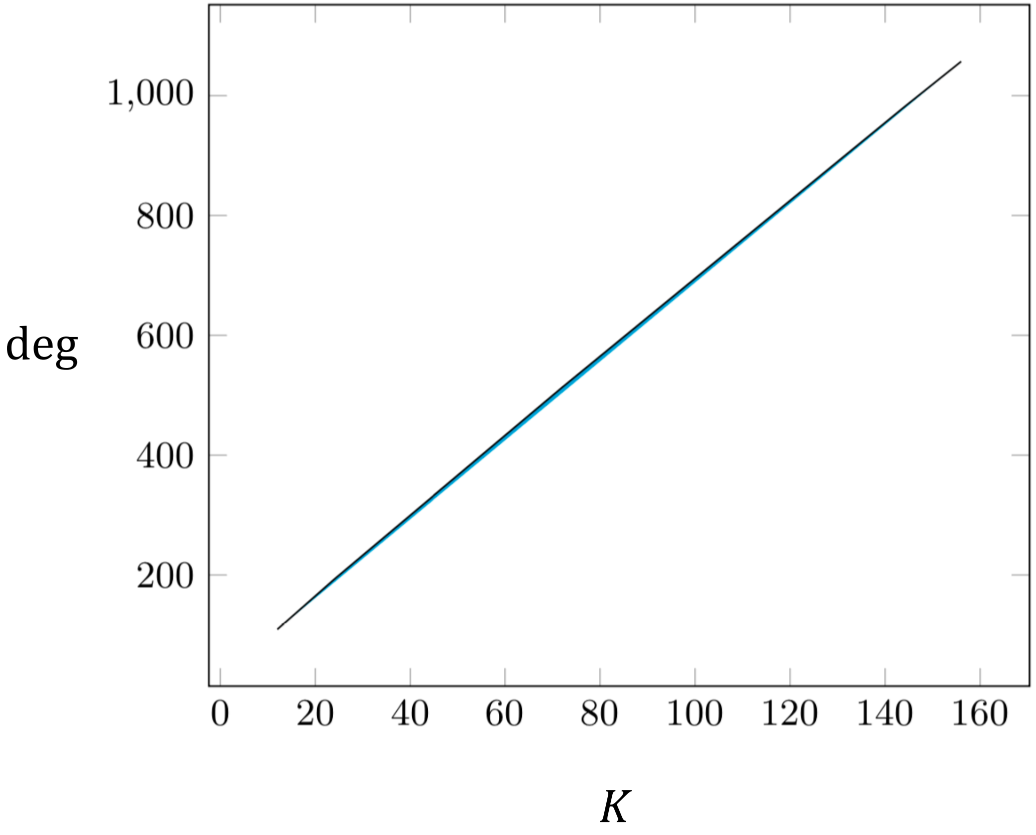
$\text{deg} \simeq 7K + 25$

Using **Paterson-Stockmeyer** such approximation require:

- $\simeq \sqrt{2(7K+25)} + \log_2(2(7K+25))$ multiplications
- $\simeq \log_2 K + 3$ mult. levels

Chebyshev approximation grows linearly with K

Approximation error: 10^{-12}



$\text{deg} \simeq 7K + 25$

Using Paterson-Stockmeyer such approximation require:

- $\simeq \sqrt{2(7K+25)} + \log_2(2(7K+25))$ multiplications
- $\simeq \log_2 K + 3$ mult. levels

Example:

$n = 65536 \Rightarrow K = 360$:

- 84 multiplications
- 12 mult. levels

Use the double-angle formula for cosine

$\cos 2\alpha = 2\cos^2\alpha - 1$

Use the double-angle formula for cosine

$$\cos 2\alpha = 2\cos^2 \alpha - 1$$

1. Take a sufficiently large $K = 2^k$.

Use the double-angle formula for cosine

$$\cos 2\alpha = 2\cos^2\alpha - 1$$

1. Take a sufficiently large $K = 2^k$.
2. Approximate $\cos\left(2\pi x - \frac{\pi}{2K}\right)$ in the range $[-q, q]$ (e.g using Chebyshev).

Use the double-angle formula for cosine

$\cos 2\alpha = 2\cos^2\alpha - 1$

1. Take a sufficiently large $K = 2^k$.
2. Approximate $\cos\left(2\pi x - \frac{\pi}{2K}\right)$ in the range $[-q, q]$ (e.g using Chebyshev).
3. Compute k iterations of the double-angle formula.

Use the double-angle formula for cosine

$\cos 2\alpha = 2\cos^2\alpha - 1$

1. Take a sufficiently large $K = 2^k$.
2. Approximate $\cos\left(2\pi x - \frac{\pi}{2K}\right)$ in the range $[-q, q]$ (e.g using Chebyshev).
3. Compute k iterations of the double-angle formula.

Example:
$n = 65536 \Rightarrow K = 2^9$

Use the double-angle formula for cosine

$\cos 2\alpha = 2\cos^2\alpha - 1$

1. Take a sufficiently large $K = 2^k$.
2. Approximate $\cos\left(2\pi x - \frac{\pi}{2K}\right)$ in the range $[-q, q]$ (e.g using Chebyshev).
3. Compute k iterations of the double-angle formula.

Example:

$n = 65536 \Rightarrow K = 2^9$:

$\cos\left(2\pi x - \frac{\pi}{2K}\right) \simeq p(X), \;\; \deg p(X) = 26$

Use the double-angle formula for cosine

$\cos 2\alpha = 2\cos^2\alpha - 1$

1. Take a sufficiently large $K = 2^k$.
2. Approximate $\cos\left(2\pi x - \frac{\pi}{2K}\right)$ in the range $[-q, q]$ (e.g using Chebyshev).
3. Compute k iterations of the double-angle formula.

Example:

$n = 65536 \Rightarrow K = 2^9$:

$\cos\left(2\pi x - \frac{\pi}{2K}\right) \simeq p(X),\ \ \deg p(X) = 26$

9 iterations of the double-angle formula

Use the double-angle formula for cosine

$$\cos 2\alpha = 2\cos^2 \alpha - 1$$

1. Take a sufficiently large $K = 2^k$.
2. Approximate $\cos\left(2\pi x - \frac{\pi}{2K}\right)$ in the range $[-q, q]$ (e.g using Chebyshev).
3. Compute k iterations of the double-angle formula.

Example:

$n = 65536 \Rightarrow K = 2^9$:

$\cos\left(2\pi x - \dfrac{\pi}{2K}\right) \simeq p(X), \quad \deg p(X) = 26$

9 iterations of the double-angle formula

Total cost:

- 19 multiplications
- 14 levels

Results for the entire pipeline

$n = 65536, \ \ \Delta = 2^{50}, \ q \simeq 2^{60}, \ \ \ \lambda = 128$ bits

Input data: $z \in \mathbb{C}, |\mathrm{Re}(z)|, |\mathrm{Im}(z)| < 16.$

Number of experiments per parameter set: 100

# slots	CtoS levels	StoC levels	After levels	Avg. time, sec	Avg. amort. time, msec
4096	2	2	**9**	**179**	**44**
	3	2	**8**	**114**	**28**
8192	3	2	**8**	**204**	**25**
	4	2	**7**	**121**	**15**
16384	4	3	**6**	**181**	**11**
	5	3	**5**	**159**	**10**

Precision before bootstrapping: $\simeq$ 33 bits
Precision after bootstrapping: $\simeq$ 8 bits

Results for the entire pipeline

$n = 65536, \;\; \Delta = 2^{50}, \;\; q \simeq 2^{60}, \quad \lambda = 128$ bits

Input data: $z \in \mathbb{C}, |\mathrm{Re}(z)|, |\mathrm{Im}(z)| < 16.$

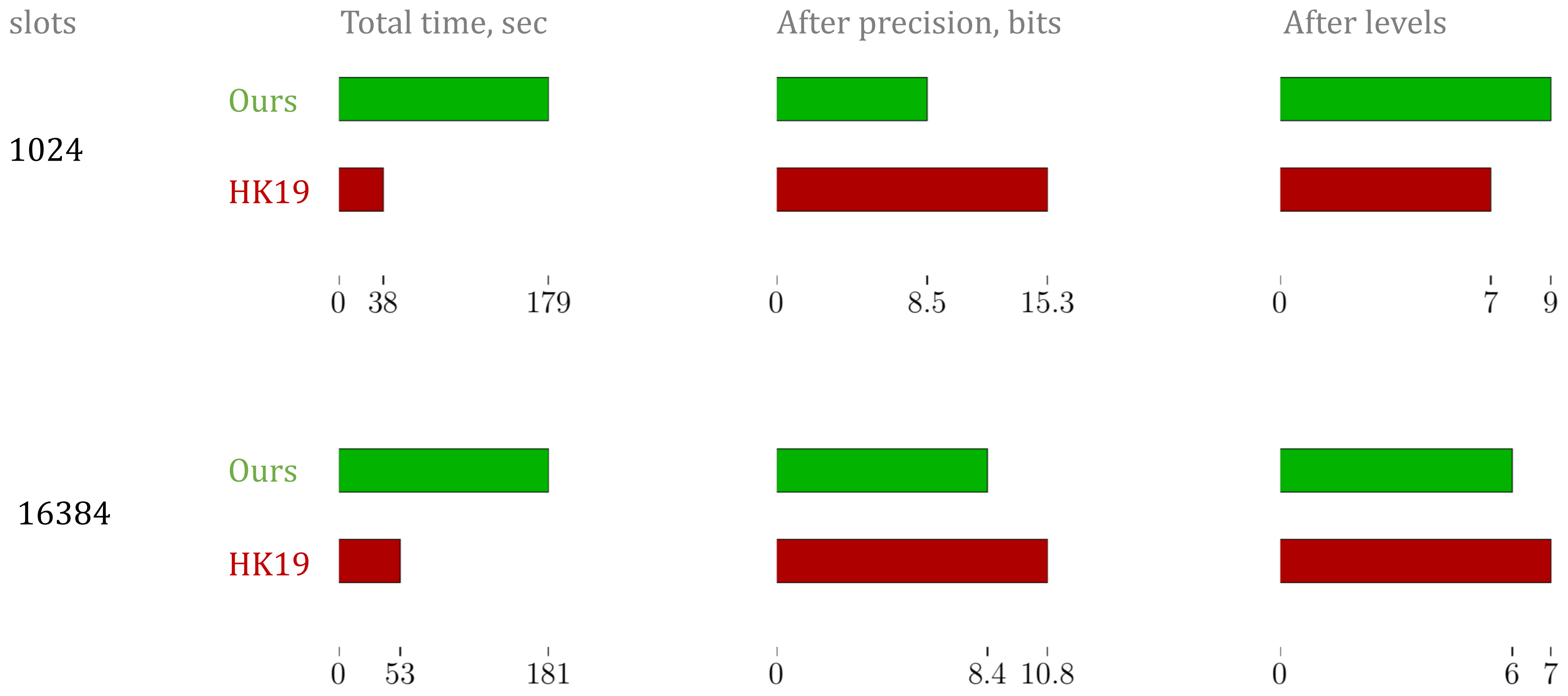
Number of experiments per parameter set: 100

# slots	CtoS levels	StoC levels	After levels	Avg. time, sec	Avg. amort. time, msec
4096	2	2	**9**	**179**	**44**
	3	2	**8**	**114**	**28**
8192	3	2	**8**	**204**	**25**
	4	2	**7**	**121**	**15**
16384	4	3	**6**	**181**	**11**
	5	3	**5**	**159**	**10**

Precision before bootstrapping: $\simeq 33$ bits
Precision after bootstrapping: $\simeq 8$ bits

Memory consumption: ~47GB
(mostly due to key-switching keys)

Comparison to HK19



Conclusion

- Attacks on sparse-secret LWE/RLWE become more powerful.

Conclusion

- Attacks on sparse-secret LWE/RLWE become more powerful.
- HEAAN can avoid sparse secrets as its “bootstrapping” is practically possible without them.

Future work

- Bootstrapping definition for HEAAN.

Future work

- Bootstrapping definition for HEAAN.
- Better approximation of mod q (e.g. Hermite approximation of HK19).

Future work

- Bootstrapping definition for HEAAN.
- Better approximation of mod q (e.g. Hermite approximation of HK19).
- Mixed bootstrapping using other schemes (e.g. TFHE).

Future work

- Bootstrapping definition for HEAAN.
- Better approximation of mod q (e.g. Hermite approximation of HK19).
- Mixed bootstrapping using other schemes (e.g. TFHE).
- Bootstrapping without sparse secrets in other schemes.

Thank you!

Thank you!

We're hiring!