Securing Approximate Homomorphic Encryption using Differential Privacy

Baiyu Li Daniele Micciancio Mark Schultz Jessica Sorrell

University of California San Diego

15 August 2022

Contents

Contents

Fully Homomorphic Encryption

FHE (informal)

A cryptosystem (KeyGen, Enc, Dec) is *fully homomorphic* if it admits two *privacy homomorphisms*

$$\mathsf{Enc}_{\mathsf{pk}}(m) + \mathsf{Enc}_{\mathsf{pk}}(m') \mapsto \mathsf{Enc}_{\mathsf{pk}}(m + m'),$$
$$\mathsf{Enc}_{\mathsf{pk}}(m) \times \mathsf{Enc}_{\mathsf{pk}}(m') \mapsto \mathsf{Enc}_{\mathsf{pk}}(m \times m').$$

Fully Homomorphic Encryption

FHE (informal)

A cryptosystem (KeyGen, Enc, Dec) is *fully homomorphic* if it admits two *privacy homomorphisms*

$$\mathsf{Enc}_{\mathsf{pk}}(m) + \mathsf{Enc}_{\mathsf{pk}}(m') \mapsto \mathsf{Enc}_{\mathsf{pk}}(m + m'),$$
$$\mathsf{Enc}_{\mathsf{pk}}(m) \times \mathsf{Enc}_{\mathsf{pk}}(m') \mapsto \mathsf{Enc}_{\mathsf{pk}}(m \times m').$$

- Suffices to compute arbitrary boolean functions

Fully Homomorphic Encryption

FHE (informal)

A cryptosystem (KeyGen, Enc, Dec) is *fully homomorphic* if it admits two *privacy homomorphisms*

$$\mathsf{Enc}_{\mathsf{pk}}(m) + \mathsf{Enc}_{\mathsf{pk}}(m') \mapsto \mathsf{Enc}_{\mathsf{pk}}(m + m'),$$
$$\mathsf{Enc}_{\mathsf{pk}}(m) \times \mathsf{Enc}_{\mathsf{pk}}(m') \mapsto \mathsf{Enc}_{\mathsf{pk}}(m \times m').$$

- Suffices to compute arbitrary boolean functions
- Many constructions [Gen09, GSW13, CKKS17]

Fully Homomorphic Encryption

FHE (informal)

A cryptosystem (KeyGen, Enc, Dec) is *fully homomorphic* if it admits two *privacy homomorphisms*

$$\mathsf{Enc}_{\mathsf{pk}}(m) + \mathsf{Enc}_{\mathsf{pk}}(m') \mapsto \mathsf{Enc}_{\mathsf{pk}}(m + m'),$$
$$\mathsf{Enc}_{\mathsf{pk}}(m) \times \mathsf{Enc}_{\mathsf{pk}}(m') \mapsto \mathsf{Enc}_{\mathsf{pk}}(m \times m').$$

- Suffices to compute arbitrary boolean functions
- Many constructions [Gen09, GSW13, CKKS17]
- Our focus — approximately correct FHE [CKKS17]

q-IND-CPA^D Security [LM21]

- Example Application of FHE

q-IND-CPA^D Security [LM21]

- Example Application of FHE
  - User encrypts data

q-IND-CPA^D Security [LM21]

- Example Application of FHE
  - User encrypts data
  - Sends it to a server to homomorphically compute

q-IND-CPA^D Security [LM21]

- Example Application of FHE
  - User encrypts data
  - Sends it to a server to homomorphically compute
  - Server sends result to secret-key holder

q-IND-CPA^D Security [LM21]

- Example Application of FHE
  - User encrypts data
  - Sends it to a server to homomorphically compute
  - Server sends result to secret-key holder
  - Secret-key holder decrypts

q-IND-CPA^D Security [LM21]

- Example Application of FHE
  - User encrypts data
  - Sends it to a server to homomorphically compute
  - Server sends result to secret-key holder
  - Secret-key holder decrypts
- User can passively observe
  - Ciphertexts

q-IND-CPA^D Security [LM21]

- Example Application of FHE
  - User encrypts data
  - Sends it to a server to homomorphically compute
  - Server sends result to secret-key holder
  - Secret-key holder decrypts
- User can passively observe
  - Ciphertexts
  - Results of homomorphic computations
  - Behavior of secret key holder depending on decrypted result

q-IND-CPA^D Security [LM21]

- Example Application of FHE
  - User encrypts data
  - Sends it to a server to homomorphically compute
  - Server sends result to secret-key holder
  - Secret-key holder decrypts
- User can passively observe
  - Ciphertexts
  - Results of homomorphic computations
  - Behavior of secret key holder depending on decrypted result
- For correct encryption, an adversary doesn't need to observe this last quantity

q-IND-CPA^D Security [LM21]

- Example Application of FHE
  - User encrypts data
  - Sends it to a server to homomorphically compute
  - Server sends result to secret-key holder
  - Secret-key holder decrypts
- User can passively observe
  - Ciphertexts
  - Results of homomorphic computations
  - Behavior of secret key holder depending on decrypted result
- For correct encryption, an adversary doesn't need to observe this last quantity
  - can locally compute what decrypted result should be
- For incorrect encryption, decryption may leak information

q-IND-CPA^D Security [LM21]

- Example Application of FHE
  - User encrypts data
  - Sends it to a server to homomorphically compute
  - Server sends result to secret-key holder
  - Secret-key holder decrypts
- User can passively observe
  - Ciphertexts
  - Results of homomorphic computations
  - Behavior of secret key holder depending on decrypted result
- For correct encryption, an adversary doesn't need to observe this last quantity
  - can locally compute what decrypted result should be
- For incorrect encryption, decryption may leak information
  - IND-CPA^D: "IND-CPA with decryption oracles"

q-IND-CPA^D Security [LM21]

- Example Application of FHE
  - User encrypts data
  - Sends it to a server to homomorphically compute
  - Server sends result to secret-key holder
  - Secret-key holder decrypts
- User can passively observe
  - Ciphertexts
  - Results of homomorphic computations
  - Behavior of secret key holder depending on decrypted result
- For correct encryption, an adversary doesn't need to observe this last quantity
  - can locally compute what decrypted result should be
- For incorrect encryption, decryption may leak information
  - IND-CPA^D: "IND-CPA with decryption oracles"
  - q-IND-CPA^D: Restrict IND-CPA^D to q decryption queries

Approximate Fully Homomorphic Encryption

- Many applications of FHE in Privacy Preserving Machine Learning

Approximate Fully Homomorphic Encryption

- Many applications of FHE in Privacy Preserving Machine Learning
  - Computations only need to be approximate

Approximate Fully Homomorphic Encryption

- Many applications of FHE in Privacy Preserving Machine Learning
  - Computations only need to be approximate
  - CKKS particularly popular in this setting

Approximate Fully Homomorphic Encryption

- Many applications of FHE in Privacy Preserving Machine Learning
  - Computations only need to be approximate
  - CKKS particularly popular in this setting
    - Easy to use for homomorphic computations involving (approximations of) real numbers

Approximate Fully Homomorphic Encryption

- Many applications of FHE in Privacy Preserving Machine Learning
  - Computations only need to be approximate
  - CKKS particularly popular in this setting
    - Easy to use for homomorphic computations involving (approximations of) real numbers
- What does approximately correct mean?

Approximate Fully Homomorphic Encryption

- Many applications of FHE in Privacy Preserving Machine Learning
  - Computations only need to be approximate
  - CKKS particularly popular in this setting
    - Easy to use for homomorphic computations involving (approximations of) real numbers
- What does approximately correct mean?
  - In the IND-CPA^D setting, impacts security, not only output quality

Approximate Fully Homomorphic Encryption

- Many applications of FHE in Privacy Preserving Machine Learning
  - Computations only need to be approximate
  - CKKS particularly popular in this setting
    - Easy to use for homomorphic computations involving (approximations of) real numbers
- What does approximately correct mean?
  - In the IND-CPA^D setting, impacts security, not only output quality
  - Important to have formal definition

Approximate FHE: Error Definition

Definition: Ciphertext Error

Let Π be an FHE scheme with message space $\mathcal{M}$ that is a subset of a normed space $\widetilde{\mathcal{M}}$. For a ciphertext c, message m, and secret key sk, define the plaintext error to be

$$\mathsf{PtxtError}(c, m, \mathsf{sk}) = \|\mathsf{Dec}_{\mathsf{sk}}(c) - m\|.$$

Approximate FHE: Error Definition

Definition: Ciphertext Error

Let Π be an FHE scheme with message space $\mathcal{M}$ that is a subset of a normed space $\widetilde{\mathcal{M}}$. For a ciphertext c, message m, and secret key sk, define the plaintext error to be

$$\mathsf{PtxtError}(c, m, \mathsf{sk}) = \|\mathsf{Dec}_{\mathsf{sk}}(c) - m\|.$$

- Correct: $\mathsf{PtxtError}(c, m, \mathsf{sk}) = 0$

Approximate FHE: Error Definition

Definition: Ciphertext Error

Let Π be an FHE scheme with message space $\mathcal{M}$ that is a subset of a normed space $\widetilde{\mathcal{M}}$. For a ciphertext c, message m, and secret key sk, define the plaintext error to be

$$\mathsf{PtxtError}(c, m, \mathsf{sk}) = \|\mathsf{Dec}_{\mathsf{sk}}(c) - m\|.$$

- Correct: $\mathsf{PtxtError}(c, m, \mathsf{sk}) = 0$
- Statically Correct: $\mathsf{PtxtError}(c, m, \mathsf{sk}) \leq f(D, \|m_i\|)$ is publically computable

Approximate FHE: Error Definition

Definition: Ciphertext Error

Let Π be an FHE scheme with message space $\mathcal{M}$ that is a subset of a normed space $\widetilde{\mathcal{M}}$. For a ciphertext c, message m, and secret key sk, define the plaintext error to be

$$\mathsf{PtxtError}(c, m, \mathsf{sk}) = \|\mathsf{Dec}_{\mathsf{sk}}(c) - m\|.$$

- Correct: $\mathsf{PtxtError}(c, m, \mathsf{sk}) = 0$
- Statically Correct: $\mathsf{PtxtError}(c, m, \mathsf{sk}) \leq f(D, \|m_i\|)$ is publically computable
- Dynamically Correct: $\mathsf{PtxtError}(c, m, \mathsf{sk}) \leq f(c, \mathsf{sk})$ is computable during decryption

Contents

High Level Argument

- High level

High Level Argument

- High level
  - Start with true (statically-correct) scheme

High Level Argument

- High level
  - Start with true (statically-correct) scheme
  - Post-process decryption with notion of differential privacy

High Level Argument

- High level
  - Start with true (statically-correct) scheme
  - Post-process decryption with notion of differential privacy
  - Decryptions are indistinguishable from (post-processed) correct decryptions

High Level Argument

- High level
  - Start with true (statically-correct) scheme
  - Post-process decryption with notion of differential privacy
  - Decryptions are indistinguishable from (post-processed) correct decryptions
  - Simulate (correct, post-processed) decryptions without a decryption oracle

High Level Argument

- High level
  - Start with true (statically-correct) scheme
  - Post-process decryption with notion of differential privacy
  - Decryptions are indistinguishable from (post-processed) correct decryptions
  - Simulate (correct, post-processed) decryptions without a decryption oracle
- Our work

High Level Argument

- High level
  - Start with true (statically-correct) scheme
  - Post-process decryption with notion of differential privacy
  - Decryptions are indistinguishable from (post-processed) correct decryptions
  - Simulate (correct, post-processed) decryptions without a decryption oracle
- Our work:
  - General analysis of the post-processing

High Level Argument

- High level
  - Start with true (statically-correct) scheme
  - Post-process decryption with notion of differential privacy
  - Decryptions are indistinguishable from (post-processed) correct decryptions
  - Simulate (correct, post-processed) decryptions without a decryption oracle
- Our work:
  - General analysis of the post-processing
  - Give nearly tight bounds

High Level Argument

- High level
  - Start with true (statically-correct) scheme
  - Post-process decryption with notion of differential privacy
  - Decryptions are indistinguishable from (post-processed) correct decryptions
  - Simulate (correct, post-processed) decryptions without a decryption oracle
- Our work:
  - General analysis of the post-processing
  - Give nearly tight bounds
  - Give novel weakening of concrete security that enables smaller parameters

Norm KL Differential Privacy

Norm KL Differential Privacy

A family of randomized algorithms (indexed by $t \geq 0$) is said to be ρ-KL differentially private if, for all x, x' such that $\|x - x'\| \leq t$

$$D(M_t(x)||M_t(x')) \leq \rho$$

Norm KL Differential Privacy

Norm KL Differential Privacy

A family of randomized algorithms (indexed by $t \geq 0$) is said to be ρ-KL differentially private if, for all x, x' such that $\|x - x'\| \leq t$

$$D(M_t(x)||M_t(x')) \leq \rho$$

- Generalization of (Renyi) DP from the Hamming norm to general norms

Norm KL Differential Privacy

Norm KL Differential Privacy

A family of randomized algorithms (indexed by $t \geq 0$) is said to be ρ-KL differentially private if, for all x, x' such that $\|x - x'\| \leq t$

$$D(M_t(x)||M_t(x')) \leq \rho$$

- Generalization of (Renyi) DP from the Hamming norm to general norms
- Vital that $M_t(x)$ takes as input t

Gaussian Noise Flooding

Discrete Gaussian Mechanism

Let $\rho \geq 0$. The mechanism that, on input $x \in \mathbb{Z}^n$, returns a Gaussian sample $x + \mathcal{N}_{\mathbb{Z}^n}(0, (t^2/2\rho)I_n)$ is ρ-KLDP.

- It (and uniform noise flooding) are exceedingly common in the literature

c-Bit Security

Definition: c-Bits of Security [MW18]

A protocol is said to have c-bits of security in some cryptographic game G if for every adversary A

$$c \leq \log_2 \frac{T(A)}{\mathrm{adv}_G^A}.$$

c-Bit Security

Definition: c-Bits of Security [MW18]

A protocol is said to have c-bits of security in some cryptographic game G if for every adversary A

$$c \leq \log_2 \frac{T(A)}{\mathrm{adv}_G^A}.$$

- For search games: adv_G^A is the success probability

c-Bit Security

Definition: c-Bits of Security [MW18]

A protocol is said to have c-bits of security in some cryptographic game G if for every adversary A

$$c \leq \log_2 \frac{T(A)}{\mathrm{adv}_G^A}.$$

- For search games: adv_G^A is the success probability
- For decision games $\mathrm{adv}_G^A = O(\Delta(\cdot,\cdot)^2)$

c-Bit Security

Definition: c-Bits of Security [MW18]

A protocol is said to have c-bits of security in some cryptographic game G if for every adversary A

$$c \leq \log_2 \frac{T(A)}{\mathrm{adv}_G^A}.$$

- For search games: adv_G^A is the success probability
- For decision games $\mathrm{adv}_G^A = O(\Delta(\cdot,\cdot)^2)$
  - Resolves several paradoxes (see [MW18])

Achieving q-IND-CPA^D Security: Main Theorem

Main Theorem: Simplified Version

If CKKS is $(c + \log_2 24)$-bit IND-CPA-secure, then modifying CKKS to decrypt via

Achieving q-IND-CPA^D Security: Main Theorem

Main Theorem: Simplified Version

If CKKS is $(c + \log_2 24)$-bit IND-CPA-secure, then modifying CKKS to decrypt via

- computing the publically computable bound t on $\mathsf{PtxtError}(\cdot, \cdot, \cdot)$

Achieving q-IND-CPA$^{\mathsf{D}}$ Security: Main Theorem

Main Theorem: Simplified Version

If CKKS is $(c + \log_2 24)$-bit IND-CPA-secure, then modifying CKKS to decrypt via

- computing the publically computable bound t on $\mathsf{PtxtError}(\cdot, \cdot, \cdot)$
- returning $\mathsf{Dec}'_{\mathsf{sk}}(c) = \mathcal{N}_{\mathbb{Z}^n}(\mathsf{Dec}_{\mathsf{sk}}(c), 24nq2^c t^2 I_n)$

is c-bit q-IND-CPA$^{\mathsf{D}}$-secure.

Achieving q-IND-CPA$^{\mathsf{D}}$ Security: Main Theorem

Main Theorem: Simplified Version

If CKKS is $(c + \log_2 24)$-bit IND-CPA-secure, then modifying CKKS to decrypt via

- computing the publically computable bound t on $\mathsf{PtxtError}(\cdot, \cdot, \cdot)$
- returning $\mathsf{Dec}'_{\mathsf{sk}}(c) = \mathcal{N}_{\mathbb{Z}^n}(\mathsf{Dec}_{\mathsf{sk}}(c), 24nq2^c t^2 I_n)$

is c-bit q-IND-CPA$^{\mathsf{D}}$-secure.

- $(c/2) + (1/2)\log_2(nq) + O(1)$ additional bits of noise

Achieving q-IND-CPA$^{\mathsf{D}}$ Security: Main Theorem

Main Theorem: Simplified Version

If CKKS is $(c + \log_2 24)$-bit IND-CPA-secure, then modifying CKKS to decrypt via

- computing the publically computable bound t on $\mathsf{PtxtError}(\cdot, \cdot, \cdot)$
- returning $\mathsf{Dec}'_{\mathsf{sk}}(c) = \mathcal{N}_{\mathbb{Z}^n}(\mathsf{Dec}_{\mathsf{sk}}(c), 24nq2^c t^2 I_n)$

is c-bit q-IND-CPA$^{\mathsf{D}}$-secure.

- $(c/2) + (1/2)\log_2(nq) + O(1)$ additional bits of noise
- Main term is $(c/2)$ — can it be lower?

Achieving q-IND-CPA$^{\mathsf{D}}$ Security: Main Theorem

Main Theorem: Simplified Version

If CKKS is $(c + \log_2 24)$-bit IND-CPA-secure, then modifying CKKS to decrypt via

- computing the publically computable bound t on $\mathsf{PtxtError}(\cdot, \cdot, \cdot)$
- returning $\mathsf{Dec}'_{\mathsf{sk}}(c) = \mathcal{N}_{\mathbb{Z}^n}(\mathsf{Dec}_{\mathsf{sk}}(c), 24nq2^c t^2 I_n)$

is c-bit q-IND-CPA$^{\mathsf{D}}$-secure.

- $(c/2) + (1/2)\log_2(nq) + O(1)$ additional bits of noise
- Main term is $(c/2)$ — can it be lower?
  - Not much

Achieving q-IND-CPA$^{\mathsf{D}}$ Security: Main Theorem

Main Theorem: Simplified Version

If CKKS is $(c + \log_2 24)$-bit IND-CPA-secure, then modifying CKKS to decrypt via

- computing the publically computable bound t on $\mathsf{PtxtError}(\cdot, \cdot, \cdot)$
- returning $\mathsf{Dec}'_{\mathsf{sk}}(c) = \mathcal{N}_{\mathbb{Z}^n}(\mathsf{Dec}_{\mathsf{sk}}(c), 24nq2^c t^2 I_n)$

is c-bit q-IND-CPA$^{\mathsf{D}}$-secure.

- $(c/2) + (1/2)\log_2(nq) + O(1)$ additional bits of noise
- Main term is $(c/2)$ — can it be lower?
  - Not much (without a weaker security definition)

Nearly Matching Attack

for $i \in \{0, \ldots, 44\}$ **do**
  $C_i \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_i^{(0)} = 0, m_i^{(1)} = B)$;
end for
for $i \in \{45, \ldots, 59\}$ **do**
  $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}}(m_i^{(0)} = 0, m_i^{(1)} = -B)$;
end for
$c_{60} \leftarrow \mathsf{Eval}_{\mathsf{pk}}(g, \{0, \ldots, 59\})$ for $g(x_0, \ldots, x_{59}) = \sum_{i=0}^{29}(x_i \cdot x_{30+i})$
$m' \leftarrow \mathsf{Dec}_{\mathsf{sk}}(60)$
$V_0 = 30\sigma^4 + O(wn) + \sigma_s^2$
$V_1 = 30\sigma^4 + 60B^2\sigma^2 + O(wn) + \sigma_s^2$
if $|\varphi(m')_0| < \sqrt{\frac{\log(V_1/V_0)V_0V_1}{V_1 - V_0}}$ **then**
  return 0
else
  return 1
end if

- Analysis implies that $c/4 - \tilde{O}(1)$ bits of additional noise is required

Contents

(c, s)-Bit Security

Definition: (c, s)-Bits of Security

A protocol is said to have (c, s)-bits of security in some cryptographic game G if for every adversary A

$$c \leq \log_2 \frac{T(A)}{\mathrm{adv}_G^A} \text{ or } s \leq \log_2 \frac{1}{\mathrm{adv}_G^A}.$$

(c, s)-Bit Security

Definition: (c, s)-Bits of Security

A protocol is said to have (c, s)-bits of security in some cryptographic game G if for every adversary A

$$c \leq \log_2 \frac{T(A)}{\mathrm{adv}_G^A} \text{ or } s \leq \log_2 \frac{1}{\mathrm{adv}_G^A}.$$

- Easier to satisfy due to second bound

(c, s)-Bit Security

Definition: (c, s)-Bits of Security

A protocol is said to have (c, s)-bits of security in some cryptographic game G if for every adversary A

$$c \leq \log_2 \frac{T(A)}{\mathrm{adv}_G^A} \text{ or } s \leq \log_2 \frac{1}{\mathrm{adv}_G^A}.$$

- Easier to satisfy due to second bound
- Captures statistical attacks

(c, s)-Bit Security

Definition: (c, s)-Bits of Security

A protocol is said to have (c, s)-bits of security in some cryptographic game G if for every adversary A

$$c \leq \log_2 \frac{T(A)}{\mathrm{adv}_G^A} \text{ or } s \leq \log_2 \frac{1}{\mathrm{adv}_G^A}.$$

- Easier to satisfy due to second bound
- Captures statistical attacks
  - often can compute adv_G^A explicitly

(c, s)-Bit Security

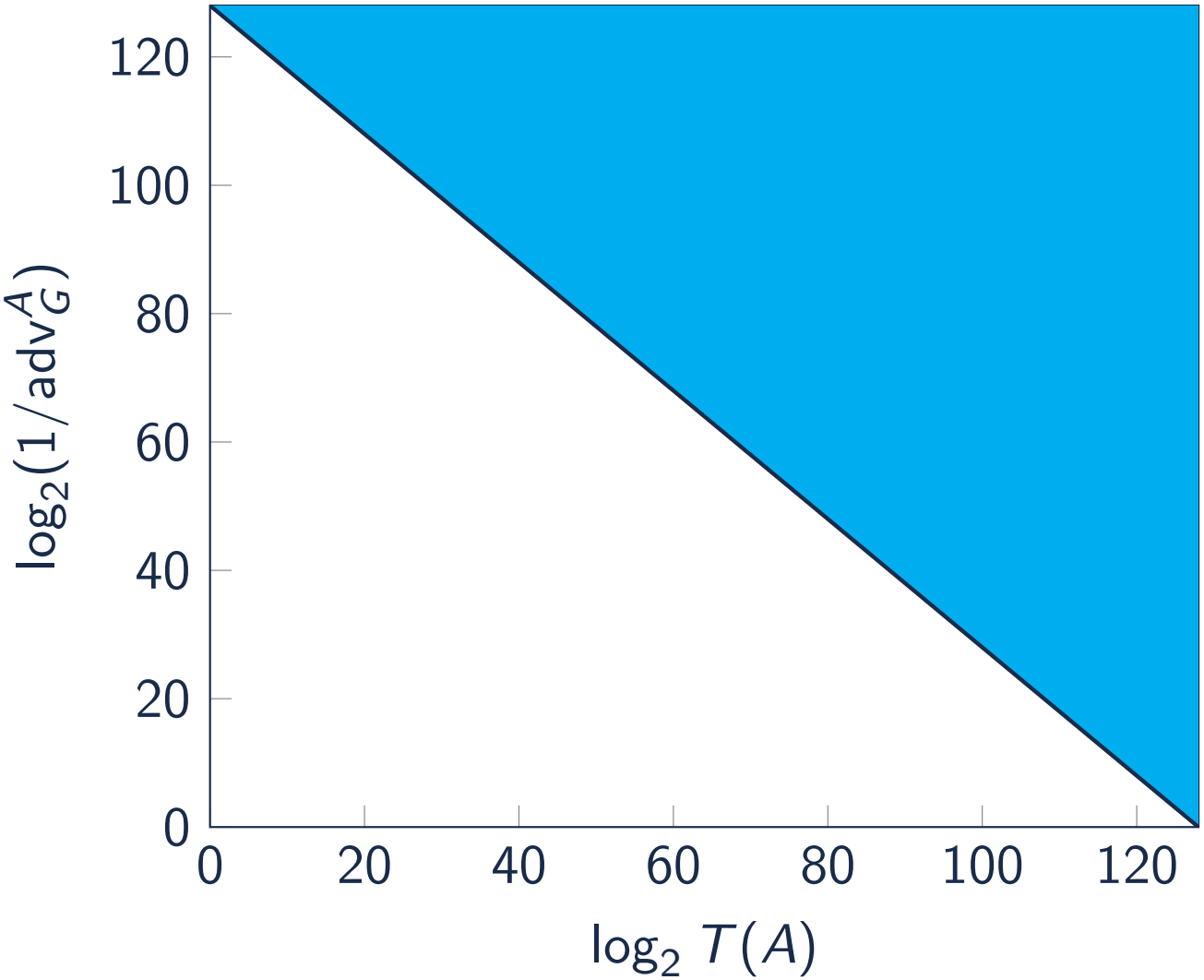
Definition: (c, s)-Bits of Security

A protocol is said to have (c, s)-bits of security in some cryptographic game G if for every adversary A

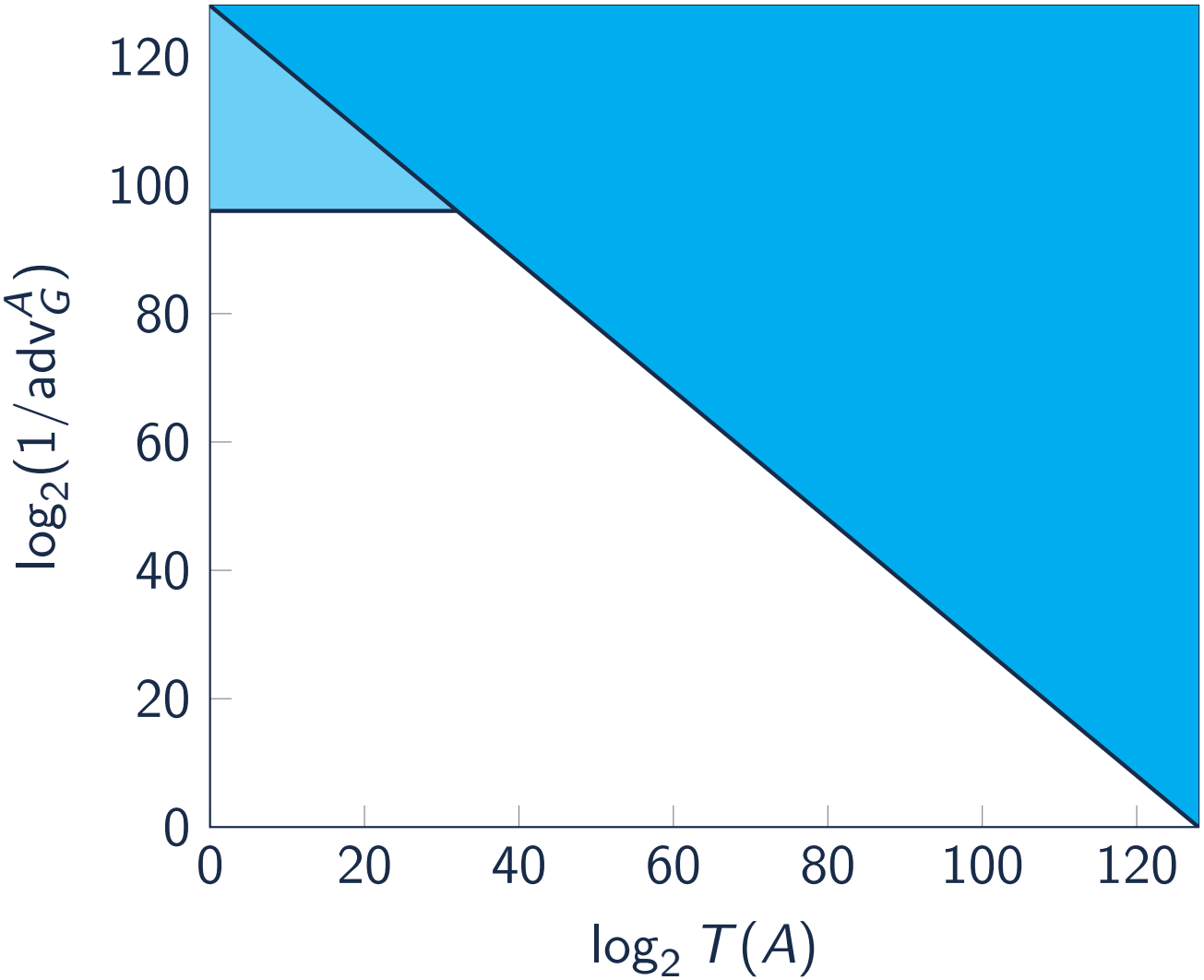
$$c \leq \log_2 \frac{T(A)}{\mathrm{adv}_G^A} \text{ or } s \leq \log_2 \frac{1}{\mathrm{adv}_G^A}.$$

- Easier to satisfy due to second bound
- Captures statistical attacks
  - often can compute adv_G^A explicitly
- Can plausibly set $s \ll c$

128-Bit Security Graphically



(128, 96)-Bit Security Graphically



Main Theorem: Full Version

Main Theorem: Simplified Version

If CKKS is $(c + \log_2 24)$-bit IND-CPA-secure, then modifying CKKS to decrypt via

Main Theorem: Full Version

Main Theorem: Simplified Version

If CKKS is $(c + \log_2 24)$-bit IND-CPA-secure, then modifying CKKS to decrypt via

- computing the publically computable bound t on $\mathsf{PtxtError}(\cdot, \cdot, \cdot)$

Main Theorem: Full Version

Main Theorem: Simplified Version

If CKKS is $(c + \log_2 24)$-bit IND-CPA-secure, then modifying CKKS to decrypt via

- computing the publically computable bound t on $\mathsf{PtxtError}(\cdot, \cdot, \cdot)$
- returning $\mathsf{Dec}'_{\mathsf{sk}}(c) = \mathcal{N}_{\mathbb{Z}^n}(\mathsf{Dec}_{\mathsf{sk}}(c), 24nq2^s t^2 I_n)$

is (c, s)-bit q-IND-CPA$^{\mathsf{D}}$-secure.

Main Theorem: Full Version

Main Theorem: Simplified Version

If CKKS is $(c + \log_2 24)$-bit IND-CPA-secure, then modifying CKKS to decrypt via

- computing the publically computable bound t on $\mathsf{PtxtError}(\cdot, \cdot, \cdot)$
- returning $\mathsf{Dec}'_{\mathsf{sk}}(c) = \mathcal{N}_{\mathbb{Z}^n}(\mathsf{Dec}_{\mathsf{sk}}(c), 24nq2^s t^2 I_n)$

is (c, s)-bit q-IND-CPA$^{\mathsf{D}}$-secure.

- Now $s/2 + \tilde{O}(1)$ bits suffice

Main Theorem: Full Version

Main Theorem: Simplified Version

If CKKS is $(c + \log_2 24)$-bit IND-CPA-secure, then modifying CKKS to decrypt via

- computing the publically computable bound t on $\mathsf{PtxtError}(\cdot, \cdot, \cdot)$
- returning $\mathsf{Dec}'_{\mathsf{sk}}(c) = \mathcal{N}_{\mathbb{Z}^n}(\mathsf{Dec}_{\mathsf{sk}}(c), 24nq2^s t^2 I_n)$

is (c, s)-bit q-IND-CPA$^{\mathsf{D}}$-secure.

- Now $s/2 + \tilde{O}(1)$ bits suffice
- Saves $(c - s)/2$ bits of noise compared to the c-bit security

Parameters for $(128, s)$-bits of q-IND-CPA^D security

$s \backslash q$	1	2^5	2^{10}	2^{15}
128	73.79	76.29	78.79	81.29
112	65.79	68.29	70.79	73.29
96	57.79	60.29	62.79	65.29
80	49.79	52.29	54.79	57.29
64	41.79	44.29	46.79	49.29
48	33.79	36.29	38.79	41.29
32	25.79	28.29	30.79	33.29

Table 1: Table of additional Gaussian noise (in bits) to add to achieve $(128, s)$-bit security with lattice dimension $n \leq 2^{15}$

Practicality of our Countermeasure

- Countermeasures are somewhat large

Practicality of our Countermeasure

- Countermeasures are somewhat large
  - Smaller using the notion of (c, s)-bit security

Practicality of our Countermeasure

- Countermeasures are somewhat large
  - Smaller using the notion of (c, s)-bit security
- Seems inherent due to lower bound for c-bit security

Practicality of our Countermeasure

- Countermeasures are somewhat large
  - Smaller using the notion of (c, s)-bit security
- Seems inherent due to lower bound for c-bit security
- Can our countermeasure be instantiated in practice?

Practicality of our Countermeasure

- Countermeasures are somewhat large
  - Smaller using the notion of (c, s)-bit security
- Seems inherent due to lower bound for c-bit security
- Can our countermeasure be instantiated in practice?
  - It depends on what precision a library supports

Practicality of our Countermeasure

- Countermeasures are somewhat large
  - Smaller using the notion of (c, s)-bit security
- Seems inherent due to lower bound for c-bit security
- Can our countermeasure be instantiated in practice?
  - It depends on what precision a library supports
- 64 bits of precision

Practicality of our Countermeasure

- Countermeasures are somewhat large
  - Smaller using the notion of (c, s)-bit security
- Seems inherent due to lower bound for c-bit security
- Can our countermeasure be instantiated in practice?
  - It depends on what precision a library supports
- 64 bits of precision
  - Seems hard

Practicality of our Countermeasure

- Countermeasures are somewhat large
  - Smaller using the notion of (c, s)-bit security
- Seems inherent due to lower bound for c-bit security
- Can our countermeasure be instantiated in practice?
  - It depends on what precision a library supports
- 64 bits of precision
  - Seems hard
  - aggressive choice of s already uses almost half this budget

Practicality of our Countermeasure

- Countermeasures are somewhat large
  - Smaller using the notion of (c, s)-bit security
- Seems inherent due to lower bound for c-bit security
- Can our countermeasure be instantiated in practice?
  - It depends on what precision a library supports
- 64 bits of precision
  - Seems hard
  - aggressive choice of s already uses almost half this budget
  - still need to fit the plaintext error, likely cannot expose an API supporting general (32-bit) computations

Practicality of our Countermeasure

- Countermeasures are somewhat large
  - Smaller using the notion of (c, s)-bit security
- Seems inherent due to lower bound for c-bit security
- Can our countermeasure be instantiated in practice?
  - It depends on what precision a library supports
- 64 bits of precision
  - Seems hard
  - aggressive choice of s already uses almost half this budget
  - still need to fit the plaintext error, likely cannot expose an API supporting general (32-bit) computations
    - Can maybe support constrained (8 or 16 bit) computations

Practicality of our Countermeasure

- Countermeasures are somewhat large
  - Smaller using the notion of (c, s)-bit security
- Seems inherent due to lower bound for c-bit security
- Can our countermeasure be instantiated in practice?
  - It depends on what precision a library supports
- 64 bits of precision
  - Seems hard
  - aggressive choice of s already uses almost half this budget
  - still need to fit the plaintext error, likely cannot expose an API supporting general (32-bit) computations
    - Can maybe support constrained (8 or 16 bit) computations
- 128 bits of precision

Practicality of our Countermeasure

- Countermeasures are somewhat large
  - Smaller using the notion of (c, s)-bit security
- Seems inherent due to lower bound for c-bit security
- Can our countermeasure be instantiated in practice?
  - It depends on what precision a library supports
- 64 bits of precision
  - Seems hard
  - aggressive choice of s already uses almost half this budget
  - still need to fit the plaintext error, likely cannot expose an API supporting general (32-bit) computations
    - Can maybe support constrained (8 or 16 bit) computations
- 128 bits of precision
  - Seems easy

Practicality of our Countermeasure

- Countermeasures are somewhat large
  - Smaller using the notion of (c, s)-bit security
- Seems inherent due to lower bound for c-bit security
- Can our countermeasure be instantiated in practice?
  - It depends on what precision a library supports
- 64 bits of precision
  - Seems hard
  - aggressive choice of s already uses almost half this budget
  - still need to fit the plaintext error, likely cannot expose an API supporting general (32-bit) computations
    - Can maybe support constrained (8 or 16 bit) computations
- 128 bits of precision
  - Seems easy
  - can conservatively choose $s \approx 100$ and still support general 32-bit computations

Contents

Dynamic Correctness

- Recall Dynamic Correctness: bound on PtxtError$(\cdot, \cdot, \cdot)$ is computed during decryption

Dynamic Correctness

- Recall Dynamic Correctness: bound on PtxtError$(\cdot, \cdot, \cdot)$ is computed during decryption
- Yuriy Polyakov's proposal (implemented in PALISADE):
  - View $\mathcal{M} = X^n$ for a set X

Dynamic Correctness

- Recall Dynamic Correctness: bound on PtxtError$(\cdot, \cdot, \cdot)$ is computed during decryption
- Yuriy Polyakov's proposal (implemented in PALISADE):
  - View $\mathcal{M} = X^n$ for a set X
  - Restrict to the subset $\mathcal{M}' = X^{n/2} \times \{0\}^{n/2}$

Dynamic Correctness

- Recall Dynamic Correctness: bound on PtxtError$(\cdot, \cdot, \cdot)$ is computed during decryption
- Yuriy Polyakov's proposal (implemented in PALISADE):
  - View $\mathcal{M} = X^n$ for a set X
  - Restrict to the subset $\mathcal{M}' = X^{n/2} \times \{0\}^{n/2}$
  - Restrict to circuits $(\mathcal{M}')^k \to \mathcal{M}'$

Dynamic Correctness

- Recall Dynamic Correctness: bound on PtxtError$(\cdot, \cdot, \cdot)$ is computed during decryption
- Yuriy Polyakov's proposal (implemented in PALISADE):
  - View $\mathcal{M} = X^n$ for a set X
  - Restrict to the subset $\mathcal{M}' = X^{n/2} \times \{0\}^{n/2}$
  - Restrict to circuits $(\mathcal{M}')^k \to \mathcal{M}'$
  - Measure exactly the error in the last $n/2$ coordinates during decryption

Dynamic Correctness

- Recall Dynamic Correctness: bound on PtxtError$(\cdot, \cdot, \cdot)$ is computed during decryption
- Yuriy Polyakov's proposal (implemented in PALISADE):
  - View $\mathcal{M} = X^n$ for a set X
  - Restrict to the subset $\mathcal{M}' = X^{n/2} \times \{0\}^{n/2}$
  - Restrict to circuits $(\mathcal{M}')^k \to \mathcal{M}'$
  - Measure exactly the error in the last $n/2$ coordinates during decryption
  - Generalize it to an estimate of the error in the entire ciphertext

Dynamic Correctness

- Recall Dynamic Correctness: bound on PtxtError$(\cdot, \cdot, \cdot)$ is computed during decryption
- Yuriy Polyakov's proposal (implemented in PALISADE):
  - View $\mathcal{M} = X^n$ for a set X
  - Restrict to the subset $\mathcal{M}' = X^{n/2} \times \{0\}^{n/2}$
  - Restrict to circuits $(\mathcal{M}')^k \to \mathcal{M}'$
  - Measure exactly the error in the last $n/2$ coordinates during decryption
  - Generalize it to an estimate of the error in the entire ciphertext
- Experimentally works well

Dynamic Correctness

- Recall Dynamic Correctness: bound on PtxtError$(\cdot, \cdot, \cdot)$ is computed during decryption
- Yuriy Polyakov's proposal (implemented in PALISADE):
  - View $\mathcal{M} = X^n$ for a set X
  - Restrict to the subset $\mathcal{M}' = X^{n/2} \times \{0\}^{n/2}$
  - Restrict to circuits $(\mathcal{M}')^k \to \mathcal{M}'$
  - Measure exactly the error in the last $n/2$ coordinates during decryption
  - Generalize it to an estimate of the error in the entire ciphertext
- Experimentally works well
- Can this better estimate of t reduce the total noise $O(2^{s/2}t)$?

Dynamic Correctness

- Recall Dynamic Correctness: bound on PtxtError$(\cdot, \cdot, \cdot)$ is computed during decryption
- Yuriy Polyakov's proposal (implemented in PALISADE):
  - View $\mathcal{M} = X^n$ for a set X
  - Restrict to the subset $\mathcal{M}' = X^{n/2} \times \{0\}^{n/2}$
  - Restrict to circuits $(\mathcal{M}')^k \to \mathcal{M}'$
  - Measure exactly the error in the last $n/2$ coordinates during decryption
  - Generalize it to an estimate of the error in the entire ciphertext
- Experimentally works well
- Can this better estimate of t reduce the total noise $O(2^{s/2}t)$?
  - No, there are attacks

Attacks on Dynamically Correct FHE

Theorem: Insecurity of Our Methods when applied to Dynamically-Correct FHE

For "natural" Dynamically-correct FHE schemes, post-processing by a DP mechanism is either not useful, or insecure.

Attacks on Dynamically Correct FHE

Theorem: Insecurity of Our Methods when applied to Dynamically-Correct FHE

For "natural" Dynamically-correct FHE schemes, post-processing by a DP mechanism is either not useful, or insecure.

- Similar to attack in the static setting: setup computation such that left/right worlds contain very different plaintext errors

Attacks on Dynamically Correct FHE

Theorem: Insecurity of Our Methods when applied to Dynamically-Correct FHE

For "natural" Dynamically-correct FHE schemes, post-processing by a DP mechanism is either not useful, or insecure.

- Similar to attack in the static setting: setup computation such that left/right worlds contain very different plaintext errors
- Leads to different estimates of PtxtError$(\cdot, \cdot, \cdot)$ used in the DP mechanism

Attacks on Dynamically Correct FHE

Theorem: Insecurity of Our Methods when applied to Dynamically-Correct FHE

For "natural" Dynamically-correct FHE schemes, post-processing by a DP mechanism is either not useful, or insecure.

- Similar to attack in the static setting: setup computation such that left/right worlds contain very different plaintext errors
- Leads to different estimates of PtxtError$(\cdot, \cdot, \cdot)$ used in the DP mechanism
  - No reason to expect $D(M_t(x)||M_{t'}(x'))$ to be small

Attacks on Dynamically Correct FHE

Theorem: Insecurity of Our Methods when applied to Dynamically-Correct FHE

For "natural" Dynamically-correct FHE schemes, post-processing by a DP mechanism is either not useful, or insecure.

- Similar to attack in the static setting: setup computation such that left/right worlds contain very different plaintext errors
- Leads to different estimates of PtxtError$(\cdot, \cdot, \cdot)$ used in the DP mechanism
  - No reason to expect $D(M_t(x)||M_{t'}(x'))$ to be small
- Assumptions required for it to work

Attacks on Dynamically Correct FHE

Theorem: Insecurity of Our Methods when applied to Dynamically-Correct FHE

For "natural" Dynamically-correct FHE schemes, post-processing by a DP mechanism is either not useful, or insecure.

- Similar to attack in the static setting: setup computation such that left/right worlds contain very different plaintext errors
- Leads to different estimates of PtxtError$(\cdot,\cdot,\cdot)$ used in the DP mechanism
  - No reason to expect $D(M_t(x)||M_{t'}(x'))$ to be small
- Assumptions required for it to work
  - Dynamic estimation accurately measures the differently-sized PtxtError$(\cdot,\cdot,\cdot)$

Attacks on Dynamically Correct FHE

Theorem: Insecurity of Our Methods when applied to Dynamically-Correct FHE

For "natural" Dynamically-correct FHE schemes, post-processing by a DP mechanism is either not useful, or insecure.

- Similar to attack in the static setting: setup computation such that left/right worlds contain very different plaintext errors
- Leads to different estimates of PtxtError$(\cdot,\cdot,\cdot)$ used in the DP mechanism
  - No reason to expect $D(M_t(x)||M_{t'}(x'))$ to be small
- Assumptions required for it to work
  - Dynamic estimation accurately measures the differently-sized PtxtError$(\cdot,\cdot,\cdot)$
  - DP mechanism uses smaller estimates to add noticably less noise

Dynamic Attacks on KR^D Security

Attacks on KR^D Security

There exists a dynamically correct IND-CPA-secure FHE scheme that is not KR^D-secure when post-processing with Gaussian noise.

Dynamic Attacks on KR^D Security

Attacks on KR^D Security

There exists a dynamically correct IND-CPA-secure FHE scheme that is not KR^D-secure when post-processing with Gaussian noise.

- KR^D: "Key Recovery with Decryption Oracles"

Dynamic Attacks on KR^D Security

Attacks on KR^D Security

There exists a dynamically correct IND-CPA-secure FHE scheme that is not KR^D-secure when post-processing with Gaussian noise.

- KR^D: "Key Recovery with Decryption Oracles"
- Not CKKS

Dynamic Attacks on KR^D Security

Attacks on KR^D Security

There exists a dynamically correct IND-CPA-secure FHE scheme that is not KR^D-secure when post-processing with Gaussian noise.

- KR^D: "Key Recovery with Decryption Oracles"
- Not CKKS
  - Unnatural construction

Dynamic Attacks on KR^D Security

Attacks on KR^D Security

There exists a dynamically correct IND-CPA-secure FHE scheme that is not KR^D-secure when post-processing with Gaussian noise.

- KR^D: "Key Recovery with Decryption Oracles"
- Not CKKS
  - Unnatural construction
- Rules out generic transformation from IND-CPA-secure (dynamically correct) FHE to any form of security that implies KR^D

Contents

Conclusion

- q-IND-CPA$^{\mathsf{D}}$-security of CKKS:
  - Post-process with Differential Privacy!

Conclusion

- q-IND-CPA$^{\mathsf{D}}$-security of CKKS:
  - Post-process with Differential Privacy!
    - Such as Gaussian Noise

Conclusion

- q-IND-CPA$^{\mathsf{D}}$-security of CKKS:
  - Post-process with Differential Privacy!
    - Such as Gaussian Noise
    - $s/2 + \tilde{O}(1)$ additional bits suffice for (c, s)-bits of security

Conclusion

- q-IND-CPA$^{\mathsf{D}}$-security of CKKS:
  - Post-process with Differential Privacy!
    - Such as Gaussian Noise
    - $s/2 + \tilde{O}(1)$ additional bits suffice for (c, s)-bits of security
    - Lower bound for c-bit securuity means this is plausibly tight

Conclusion

- q-IND-CPA$^{\mathsf{D}}$-security of CKKS:
  - Post-process with Differential Privacy!
    - Such as Gaussian Noise
    - $s/2 + \tilde{O}(1)$ additional bits suffice for (c, s)-bits of security
    - Lower bound for c-bit securuity means this is plausibly tight
- Concretely analyzing "mixed" computational/statistical primitives

Conclusion

- q-IND-CPA$^{\mathsf{D}}$-security of CKKS:
  - Post-process with Differential Privacy!
    - Such as Gaussian Noise
    - $s/2 + \tilde{O}(1)$ additional bits suffice for (c, s)-bits of security
    - Lower bound for c-bit securuity means this is plausibly tight
- Concretely analyzing "mixed" computational/statistical primitives
  - (c, s)-bit security greatly improved our parameters

Conclusion

- q-IND-CPA$^{\mathsf{D}}$-security of CKKS:
  - Post-process with Differential Privacy!
    - Such as Gaussian Noise
    - $s/2 + \tilde{O}(1)$ additional bits suffice for (c, s)-bits of security
    - Lower bound for c-bit securuity means this is plausibly tight
- Concretely analyzing "mixed" computational/statistical primitives
  - (c, s)-bit security greatly improved our parameters
  - No reason to think this is unique to FHE

Conclusion

- q-IND-CPA$^{\mathsf{D}}$-security of CKKS:
  - Post-process with Differential Privacy!
    - Such as Gaussian Noise
    - $s/2 + \tilde{O}(1)$ additional bits suffice for (c, s)-bits of security
    - Lower bound for c-bit securuity means this is plausibly tight
- Concretely analyzing "mixed" computational/statistical primitives
  - (c, s)-bit security greatly improved our parameters
  - No reason to think this is unique to FHE
- Dynamic notions of Correctness

Conclusion

- q-IND-CPA$^{\mathsf{D}}$-security of CKKS:
  - Post-process with Differential Privacy!
    - Such as Gaussian Noise
    - $s/2 + \tilde{O}(1)$ additional bits suffice for (c, s)-bits of security
    - Lower bound for c-bit securuity means this is plausibly tight
- Concretely analyzing "mixed" computational/statistical primitives
  - (c, s)-bit security greatly improved our parameters
  - No reason to think this is unique to FHE
- Dynamic notions of Correctness
  - Quite interesting methods, but (currently) broadly negative results

Conclusion

- q-IND-CPA$^{\mathsf{D}}$-security of CKKS:
  - Post-process with Differential Privacy!
    - Such as Gaussian Noise
    - $s/2 + \tilde{O}(1)$ additional bits suffice for (c, s)-bits of security
    - Lower bound for c-bit securuity means this is plausibly tight
- Concretely analyzing "mixed" computational/statistical primitives
  - (c, s)-bit security greatly improved our parameters
  - No reason to think this is unique to FHE
- Dynamic notions of Correctness
  - Quite interesting methods, but (currently) broadly negative results
  - Seems unlikely to be q-IND-CPA$^{\mathsf{D}}$-secure

Open Problems

- Analyze other KLDP mechanisms?

Open Problems

- Analyze other KLDP mechanisms?
  - Many candidates from the differential privacy literature

Open Problems

- Analyze other KLDP mechanisms?
  - Many candidates from the differential privacy literature
- More general lower bounds?

Open Problems

- Analyze other KLDP mechanisms?
  - Many candidates from the differential privacy literature
- More general lower bounds?
  - For (c, s)-security?

Open Problems

- Analyze other KLDP mechanisms?
  - Many candidates from the differential privacy literature
- More general lower bounds?
  - For (c, s)-security?
  - For general (non-Gaussian) mechanisms?

Open Problems

- Analyze other KLDP mechanisms?
  - Many candidates from the differential privacy literature
- More general lower bounds?
  - For (c, s)-security?
  - For general (non-Gaussian) mechanisms?
- We treat CKKS as a black box and fix it

Open Problems

- Analyze other KLDP mechanisms?
  - Many candidates from the differential privacy literature
- More general lower bounds?
  - For (c, s)-security?
  - For general (non-Gaussian) mechanisms?
- We treat CKKS as a black box and fix it
  - More efficient countermeasures by using details of CKKS?

Open Problems

- Analyze other KLDP mechanisms?
  - Many candidates from the differential privacy literature
- More general lower bounds?
  - For (c, s)-security?
  - For general (non-Gaussian) mechanisms?
- We treat CKKS as a black box and fix it
  - More efficient countermeasures by using details of CKKS?
  - Potentially utilizing dynamic correctness in a non-blackbox way?

References


Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song.
Homomorphic encryption for arithmetic of approximate numbers.
In *ASIACRYPT*, 2017.


Craig Gentry.
Fully homomorphic encryption using ideal lattices.
In *STOC*, 2009.


Craig Gentry, Amit Sahai, and Brent Waters.
Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based.
In *CRYPTO*, 2013.


Baiyu Li and Daniele Micciancio.
On the security of homomorphic encryption on approximate numbers.
In *EUROCRYPT*, 2021.


Daniele Micciancio and Michael Walter.
On the bit security of cryptographic primitives.
In *EUROCRYPT*, 2018.