CS-E4340 - Cryptography D:

Lecture 9: Fully Homomorphic Encryption

Russell W. F. Lai

Introduction

What is fully homomorphic encryption?

- Symmetric-key encryption usually destroys all utility of the encrypted message
  - No meaningful computation can be performed over the encrypted message.
- Public-key encryption schemes are usually (only) linearly homomorphic.
- Fully homomorphic encryption (FHE) allows arbitrary computation over encrypted data.
- Given arbitrary f and $\mathrm{Enc}(x)$, homomorphically evaluating f over $\mathrm{Enc}(x)$ gives $\mathrm{Enc}(f(x))$.

What is fully homomorphic encryption?

- Symmetric-key encryption usually destroys all utility of the encrypted message
  - No meaningful computation can be performed over the encrypted message.
- Public-key encryption schemes are usually (only) linearly homomorphic.
- Fully homomorphic encryption (FHE) allows arbitrary computation over encrypted data.
- Given arbitrary f and $\mathsf{Enc}(x)$, homomorphically evaluating f over $\mathsf{Enc}(x)$ gives $\mathsf{Enc}(f(x))$.

What is fully homomorphic encryption?

- Symmetric-key encryption usually destroys all utility of the encrypted message
  - No meaningful computation can be performed over the encrypted message.
- Public-key encryption schemes are usually (only) linearly homomorphic.
- Fully homomorphic encryption (FHE) allows arbitrary computation over encrypted data.
- Given arbitrary f and $\mathsf{Enc}(x)$, homomorphically evaluating f over $\mathsf{Enc}(x)$ gives $\mathsf{Enc}(f(x))$.

What is fully homomorphic encryption?

- Symmetric-key encryption usually destroys all utility of the encrypted message
  - No meaningful computation can be performed over the encrypted message.
- Public-key encryption schemes are usually (only) linearly homomorphic.
- Fully homomorphic encryption (FHE) allows arbitrary computation over encrypted data.
- Given arbitrary f and $\mathsf{Enc}(x)$, homomorphically evaluating f over $\mathsf{Enc}(x)$ gives $\mathsf{Enc}(f(x))$.

Why fully homomorphic encryption?

- Alice has secret data x.
- Alice wants to compute $f(x)$ for some public f.
- Computing f is expensive $\implies$ Delegate to Bob.

Alice		**Bob**
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$		
	$\xrightarrow{\mathsf{ctxt}_x = \mathsf{Enc}(\mathsf{pk}, x)}$	
		$\mathsf{ctxt}_{f(x)} = f(\mathsf{ctxt}_x)$
	$\xleftarrow{\mathsf{ctxt}_{f(x)}}$	
return $f(x) \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}_{f(x)})$		

- Bob learns nothing about x.
- Alice's work only depends on x, not f.
- 2 rounds of communication (optimal).

Why fully homomorphic encryption?

- Alice has secret data x.
- Alice wants to compute $f(x)$ for some public f.
- Computing f is expensive $\implies$ Delegate to Bob.

Alice		**Bob**
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$		
	$\xrightarrow{\mathsf{ctxt}_x = \mathsf{Enc}(\mathsf{pk}, x)}$	
		$\mathsf{ctxt}_{f(x)} = f(\mathsf{ctxt}_x)$
	$\xleftarrow{\mathsf{ctxt}_{f(x)}}$	
return $f(x) \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}_{f(x)})$		

- Bob learns nothing about x.
- Alice's work only depends on x, not f.
- 2 rounds of communication (optimal).

Why fully homomorphic encryption?

- Alice has secret data x.
- Alice wants to compute $f(x)$ for some public f.
- Computing f is expensive $\implies$ Delegate to Bob.

Alice | **Bob**

$(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathrm{KGen}(1^\lambda)$

$$\xrightarrow{\mathrm{ctxt}_x = \mathrm{Enc}(\mathrm{pk}, x)}$$

$$\mathrm{ctxt}_{f(x)} = f(\mathrm{ctxt}_x)$$

$$\xleftarrow{\mathrm{ctxt}_{f(x)}}$$

return $f(x) \leftarrow \mathrm{Dec}(\mathrm{sk}, \mathrm{ctxt}_{f(x)})$

- Bob learns nothing about x.
- Alice's work only depends on x, not f.
- 2 rounds of communication (optimal).

Lecture Overview

In this lecture, we will

- introduce the notion of fully homomorphic encryption (FHE)
- construct FHE from public-key encryption with (almost-)bilinear decryption
- construct FHE from learning with errors (LWE)

Definition of Fully Homomorphic Encryption

Syntax

Just like public-key encryption $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ but with additional evaluation algorithm Eval.

- † Key Generation: $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$
- † Encryption: $\mathsf{ctxt} \leftarrow \mathsf{Enc}(\mathsf{pk}, x)$
- † Decryption: $x \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt})$
- † Evaluation: $\mathsf{ctxt} \leftarrow \mathsf{Eval}(\mathsf{pk}, f, \mathsf{ctxt}_1, \ldots, \mathsf{ctxt}_n)$

Correctness

Correctness = $\underbrace{\text{decryption correctness}}_{\text{i.e. PKE correctness}}$ + evaluation correctness

- Key Generation: $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$
- Encryption: $\mathsf{ctxt} \leftarrow \mathsf{Enc}(\mathsf{pk}, x)$
- Decryption: $x \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt})$
- Evaluation: $\mathsf{ctxt} \leftarrow \mathsf{Eval}(\mathsf{pk}, f, \mathsf{ctxt}_1, \ldots, \mathsf{ctxt}_n)$

- Decryption Correctness:

$$\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, x)) = x.$$

- Evaluation Correctness: If $\mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}_i) = x_i$ for all $i \in [n]$, then

$$\mathsf{Dec}(\mathsf{sk}, \mathsf{Eval}(\mathsf{pk}, f, \mathsf{ctxt}_1, \ldots, \mathsf{ctxt}_n)) = f(x_1, \ldots, x_n).$$

IND-CPA-Security

Same as IND-CPA-security for PKE.

IND-CPA-Security of FHE Π: For any PPT adversary $\mathcal{A}$,

$$\left|\Pr\left[\mathrm{IND\text{-}CPA}^{0}_{\Pi,\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathrm{IND\text{-}CPA}^{1}_{\Pi,\mathcal{A}}(1^\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda)$$

where

$$\boxed{\begin{array}{l} \underline{\mathrm{IND\text{-}CPA}^{b}_{\Pi,\mathcal{A}}(1^\lambda)} \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda) \\ (x_0, x_1) \leftarrow \mathcal{A}(\mathsf{pk}) \\ \mathsf{ctxt}^* \leftarrow \mathsf{Enc}(\mathsf{pk}, x_b) \\ b' \leftarrow \mathcal{A}(\mathsf{ctxt}^*) \\ \textbf{return } b' \end{array}}$$

FHE from PKE with (Almost-)Bilinear Decryption

(From Daniele Micciancio's "Fully Homomorphic Encryption from the Ground Up" @ Eurocrypt 2019)

Warm-Up: The Noise-Free Case

Given PKE scheme $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ for message space $\mathbb{Z}_q$ with following properties:

- Bilinear decryption:
  - $\mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}) + \mathsf{Dec}(\mathsf{sk}', \mathsf{ctxt}) = \mathsf{Dec}(\mathsf{sk} + \mathsf{sk}', \mathsf{ctxt})$
  - $\mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}) + \mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}') = \mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt} + \mathsf{ctxt}')$
  - For any $a \in \mathbb{Z}_q$, $a \cdot \mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}) = \mathsf{Dec}(a \cdot \mathsf{sk}, \mathsf{ctxt}) = \mathsf{Dec}(\mathsf{sk}, a \cdot \mathsf{ctxt})$
- Corollary: For any linear function L over $\mathbb{Z}_q$, the following hold:
  - Key-Homomorphism: $L(\mathsf{Dec}(\mathsf{sk}_1, \mathsf{ctxt}), \ldots, \mathsf{Dec}(\mathsf{sk}_k, \mathsf{ctxt})) = \mathsf{Dec}(L(\mathsf{sk}_1, \ldots, \mathsf{sk}_k), \mathsf{ctxt})$
  - Ciphertext-Homomorphism: $L(\mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}_1), \ldots, \mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}_k)) = \mathsf{Dec}(\mathsf{sk}, L(\mathsf{ctxt}_1, \ldots, \mathsf{ctxt}_k))$

Goal: Build FHE $\Pi' = (\mathsf{KGen}', \mathsf{Enc}', \mathsf{Dec}', \mathsf{Eval}')$ for $\mathbb{Z}_q$.
(We ignore the minor problem that Π is completely broken for now.)

Warm-Up: The Noise-Free Case

Given PKE scheme $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ for message space $\mathbb{Z}_q$ with following properties:

- Bilinear decryption:
  - $\mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}) + \mathsf{Dec}(\mathsf{sk}', \mathsf{ctxt}) = \mathsf{Dec}(\mathsf{sk} + \mathsf{sk}', \mathsf{ctxt})$
  - $\mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}) + \mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}') = \mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt} + \mathsf{ctxt}')$
  - For any $a \in \mathbb{Z}_q$, $a \cdot \mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}) = \mathsf{Dec}(a \cdot \mathsf{sk}, \mathsf{ctxt}) = \mathsf{Dec}(\mathsf{sk}, a \cdot \mathsf{ctxt})$
- Corollary: For any linear function L over $\mathbb{Z}_q$, the following hold:
  - Key-Homomorphism: $L(\mathsf{Dec}(\mathsf{sk}_1, \mathsf{ctxt}), \ldots, \mathsf{Dec}(\mathsf{sk}_k, \mathsf{ctxt})) = \mathsf{Dec}(L(\mathsf{sk}_1, \ldots, \mathsf{sk}_k), \mathsf{ctxt})$
  - Ciphertext-Homomorphism: $L(\mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}_1), \ldots, \mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}_k)) = \mathsf{Dec}(\mathsf{sk}, L(\mathsf{ctxt}_1, \ldots, \mathsf{ctxt}_k))$

Goal: Build FHE $\Pi' = (\mathsf{KGen}', \mathsf{Enc}', \mathsf{Dec}', \mathsf{Eval}')$ for $\mathbb{Z}_q$.
(We ignore the minor problem that Π is completely broken for now.)

Warm-Up: The Noise-Free Case

Given PKE scheme $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ for message space $\mathbb{Z}_q$ with following properties:

- Bilinear decryption:
  - $\mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}) + \mathsf{Dec}(\mathsf{sk}', \mathsf{ctxt}) = \mathsf{Dec}(\mathsf{sk} + \mathsf{sk}', \mathsf{ctxt})$
  - $\mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}) + \mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}') = \mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt} + \mathsf{ctxt}')$
  - For any $a \in \mathbb{Z}_q$, $a \cdot \mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}) = \mathsf{Dec}(a \cdot \mathsf{sk}, \mathsf{ctxt}) = \mathsf{Dec}(\mathsf{sk}, a \cdot \mathsf{ctxt})$
- Corollary: For any linear function L over $\mathbb{Z}_q$, the following hold:
  - Key-Homomorphism: $L(\mathsf{Dec}(\mathsf{sk}_1, \mathsf{ctxt}), \ldots, \mathsf{Dec}(\mathsf{sk}_k, \mathsf{ctxt})) = \mathsf{Dec}(L(\mathsf{sk}_1, \ldots, \mathsf{sk}_k), \mathsf{ctxt})$
  - Ciphertext-Homomorphism: $L(\mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}_1), \ldots, \mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}_k)) = \mathsf{Dec}(\mathsf{sk}, L(\mathsf{ctxt}_1, \ldots, \mathsf{ctxt}_k))$

Goal: Build FHE $\Pi' = (\mathsf{KGen}', \mathsf{Enc}', \mathsf{Dec}', \mathsf{Eval}')$ for $\mathbb{Z}_q$.
(We ignore the minor problem that Π is completely broken for now.)

Supporting Homomorphic Multiplication

Observeation: $\mathrm{Dec}(y \cdot \mathrm{sk}, \mathrm{ctxt}_x) = \mathrm{Dec}(\mathrm{sk}, y \cdot \mathrm{ctxt}_x) = \mathrm{Dec}(\mathrm{sk}, \mathrm{ctxt}_{x \cdot y}) = x \cdot y$.
Idea: Encrypt $y \cdot \mathrm{sk}$ and evaluate $\mathrm{Dec}(\cdot, \mathrm{ctxt}_x)$ homomorphically.

- † Let $\mathrm{ctxt}_{\mathrm{sk}}$ be public.
- † Given ctxt_x and $\mathrm{ctxt}_{y \cdot \mathrm{sk}} = y \cdot \mathrm{ctxt}_{\mathrm{sk}}$.
- † Define the linear function $L(\cdot) = \mathrm{Dec}(\cdot, \mathrm{ctxt}_x)$.
- † Compute $\mathrm{ctxt}_{L(y \cdot \mathrm{sk})} = L(\mathrm{ctxt}_{y \cdot \mathrm{sk}})$.
- † Since $L(y \cdot \mathrm{sk}) = \mathrm{Dec}(y \cdot \mathrm{sk}, \mathrm{ctxt}_x) = x \cdot y$, we have $\mathrm{ctxt}_{L(y \cdot \mathrm{sk})} = \mathrm{ctxt}_{x \cdot y}$.

Summary: From ctxt_x and $\mathrm{ctxt}_{y \cdot \mathrm{sk}}$, we can get $\mathrm{ctxt}_{x \cdot y}$.

Supporting Homomorphic Multiplication

Observeation: $\mathrm{Dec}(y \cdot \mathrm{sk}, \mathrm{ctxt}_x) = \mathrm{Dec}(\mathrm{sk}, y \cdot \mathrm{ctxt}_x) = \mathrm{Dec}(\mathrm{sk}, \mathrm{ctxt}_{x \cdot y}) = x \cdot y$.
Idea: Encrypt $y \cdot \mathrm{sk}$ and evaluate $\mathrm{Dec}(\cdot, \mathrm{ctxt}_x)$ homomorphically.

- Let $\mathrm{ctxt}_{\mathrm{sk}}$ be public.
- Given ctxt_x and $\mathrm{ctxt}_{y \cdot \mathrm{sk}} = y \cdot \mathrm{ctxt}_{\mathrm{sk}}$.
- Define the linear function $L(\cdot) = \mathrm{Dec}(\cdot, \mathrm{ctxt}_x)$.
- Compute $\mathrm{ctxt}_{L(y \cdot \mathrm{sk})} = L(\mathrm{ctxt}_{y \cdot \mathrm{sk}})$.
- Since $L(y \cdot \mathrm{sk}) = \mathrm{Dec}(y \cdot \mathrm{sk}, \mathrm{ctxt}_x) = x \cdot y$, we have $\mathrm{ctxt}_{L(y \cdot \mathrm{sk})} = \mathrm{ctxt}_{x \cdot y}$.

Summary: From ctxt_x and $\mathrm{ctxt}_{y \cdot \mathrm{sk}}$, we can get $\mathrm{ctxt}_{x \cdot y}$.

Supporting Homomorphic Multiplication

Observeation: $\mathsf{Dec}(y \cdot \mathsf{sk}, \mathsf{ctxt}_x) = \mathsf{Dec}(\mathsf{sk}, y \cdot \mathsf{ctxt}_x) = \mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}_{x \cdot y}) = x \cdot y$.
Idea: Encrypt $y \cdot \mathsf{sk}$ and evaluate $\mathsf{Dec}(\cdot, \mathsf{ctxt}_x)$ homomorphically.

- Let $\mathsf{ctxt}_{\mathsf{sk}}$ be public.
- Given ctxt_x and $\mathsf{ctxt}_{y \cdot \mathsf{sk}} = y \cdot \mathsf{ctxt}_{\mathsf{sk}}$.
- Define the linear function $L(\cdot) = \mathsf{Dec}(\cdot, \mathsf{ctxt}_x)$.
- Compute $\mathsf{ctxt}_{L(y \cdot \mathsf{sk})} = L(\mathsf{ctxt}_{y \cdot \mathsf{sk}})$.
- Since $L(y \cdot \mathsf{sk}) = \mathsf{Dec}(y \cdot \mathsf{sk}, \mathsf{ctxt}_x) = x \cdot y$, we have $\mathsf{ctxt}_{L(y \cdot \mathsf{sk})} = \mathsf{ctxt}_{x \cdot y}$.

Summary: From ctxt_x and $\mathsf{ctxt}_{y \cdot \mathsf{sk}}$, we can get $\mathsf{ctxt}_{x \cdot y}$.

Supporting Homomorphic Multiplication

Observeation: $\mathrm{Dec}(y \cdot \mathrm{sk}, \mathrm{ctxt}_x) = \mathrm{Dec}(\mathrm{sk}, y \cdot \mathrm{ctxt}_x) = \mathrm{Dec}(\mathrm{sk}, \mathrm{ctxt}_{x \cdot y}) = x \cdot y$.
Idea: Encrypt $y \cdot \mathrm{sk}$ and evaluate $\mathrm{Dec}(\cdot, \mathrm{ctxt}_x)$ homomorphically.

- Let $\mathrm{ctxt}_{\mathrm{sk}}$ be public.
- Given ctxt_x and $\mathrm{ctxt}_{y \cdot \mathrm{sk}} = y \cdot \mathrm{ctxt}_{\mathrm{sk}}$.
- Define the linear function $L(\cdot) = \mathrm{Dec}(\cdot, \mathrm{ctxt}_x)$.
- Compute $\mathrm{ctxt}_{L(y \cdot \mathrm{sk})} = L(\mathrm{ctxt}_{y \cdot \mathrm{sk}})$.
- Since $L(y \cdot \mathrm{sk}) = \mathrm{Dec}(y \cdot \mathrm{sk}, \mathrm{ctxt}_x) = x \cdot y$, we have $\mathrm{ctxt}_{L(y \cdot \mathrm{sk})} = \mathrm{ctxt}_{x \cdot y}$.

Summary: From ctxt_x and $\mathrm{ctxt}_{y \cdot \mathrm{sk}}$, we can get $\mathrm{ctxt}_{x \cdot y}$.

Supporting Homomorphic Multiplication

Observeation: $\mathsf{Dec}(y \cdot \mathsf{sk}, \mathsf{ctxt}_x) = \mathsf{Dec}(\mathsf{sk}, y \cdot \mathsf{ctxt}_x) = \mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}_{x \cdot y}) = x \cdot y$.
Idea: Encrypt $y \cdot \mathsf{sk}$ and evaluate $\mathsf{Dec}(\cdot, \mathsf{ctxt}_x)$ homomorphically.

- † Let $\mathsf{ctxt}_{\mathsf{sk}}$ be public.
- † Given ctxt_x and $\mathsf{ctxt}_{y \cdot \mathsf{sk}} = y \cdot \mathsf{ctxt}_{\mathsf{sk}}$.
- † Define the linear function $L(\cdot) = \mathsf{Dec}(\cdot, \mathsf{ctxt}_x)$.
- † Compute $\mathsf{ctxt}_{L(y \cdot \mathsf{sk})} = L(\mathsf{ctxt}_{y \cdot \mathsf{sk}})$.
- † Since $L(y \cdot \mathsf{sk}) = \mathsf{Dec}(y \cdot \mathsf{sk}, \mathsf{ctxt}_x) = x \cdot y$, we have $\mathsf{ctxt}_{L(y \cdot \mathsf{sk})} = \mathsf{ctxt}_{x \cdot y}$.

Summary: From ctxt_x and $\mathsf{ctxt}_{y \cdot \mathsf{sk}}$, we can get $\mathsf{ctxt}_{x \cdot y}$.

Supporting Homomorphic Multiplication

Observeation: $\mathrm{Dec}(y \cdot \mathrm{sk}, \mathrm{ctxt}_x) = \mathrm{Dec}(\mathrm{sk}, y \cdot \mathrm{ctxt}_x) = \mathrm{Dec}(\mathrm{sk}, \mathrm{ctxt}_{x \cdot y}) = x \cdot y$.
Idea: Encrypt $y \cdot \mathrm{sk}$ and evaluate $\mathrm{Dec}(\cdot, \mathrm{ctxt}_x)$ homomorphically.

- Let $\mathrm{ctxt}_{\mathrm{sk}}$ be public.
- Given ctxt_x and $\mathrm{ctxt}_{y \cdot \mathrm{sk}} = y \cdot \mathrm{ctxt}_{\mathrm{sk}}$.
- Define the linear function $L(\cdot) = \mathrm{Dec}(\cdot, \mathrm{ctxt}_x)$.
- Compute $\mathrm{ctxt}_{L(y \cdot \mathrm{sk})} = L(\mathrm{ctxt}_{y \cdot \mathrm{sk}})$.
- Since $L(y \cdot \mathrm{sk}) = \mathrm{Dec}(y \cdot \mathrm{sk}, \mathrm{ctxt}_x) = x \cdot y$, we have $\mathrm{ctxt}_{L(y \cdot \mathrm{sk})} = \mathrm{ctxt}_{x \cdot y}$.

Summary: From ctxt_x and $\mathrm{ctxt}_{y \cdot \mathrm{sk}}$, we can get $\mathrm{ctxt}_{x \cdot y}$.

Supporting Homomorphic Multiplication

Observeation: $\mathrm{Dec}(y \cdot \mathrm{sk}, \mathrm{ctxt}_x) = \mathrm{Dec}(\mathrm{sk}, y \cdot \mathrm{ctxt}_x) = \mathrm{Dec}(\mathrm{sk}, \mathrm{ctxt}_{x \cdot y}) = x \cdot y$.
Idea: Encrypt $y \cdot \mathrm{sk}$ and evaluate $\mathrm{Dec}(\cdot, \mathrm{ctxt}_x)$ homomorphically.

- † Let $\mathrm{ctxt}_{\mathrm{sk}}$ be public.
- † Given ctxt_x and $\mathrm{ctxt}_{y \cdot \mathrm{sk}} = y \cdot \mathrm{ctxt}_{\mathrm{sk}}$.
- † Define the linear function $L(\cdot) = \mathrm{Dec}(\cdot, \mathrm{ctxt}_x)$.
- † Compute $\mathrm{ctxt}_{L(y \cdot \mathrm{sk})} = L(\mathrm{ctxt}_{y \cdot \mathrm{sk}})$.
- † Since $L(y \cdot \mathrm{sk}) = \mathrm{Dec}(y \cdot \mathrm{sk}, \mathrm{ctxt}_x) = x \cdot y$, we have $\mathrm{ctxt}_{L(y \cdot \mathrm{sk})} = \mathrm{ctxt}_{x \cdot y}$.

Summary: From ctxt_x and $\mathrm{ctxt}_{y \cdot \mathrm{sk}}$, we can get $\mathrm{ctxt}_{x \cdot y}$.

Supporting Homomorphic Multiplication

To get an FHE Π':

- Key Generation: Let $\mathsf{pk}' = (\mathsf{pk}, \mathsf{ctxt}_{\mathsf{sk}})$ and $\mathsf{sk}' = \mathsf{sk}$.
- Encryption: $\mathsf{Enc}'(\mathsf{pk}', x) = \mathsf{ctxt}'_x = x \cdot \mathsf{ctxt}_{\mathsf{sk}}$
- Decryption: $\mathsf{Dec}'(\mathsf{sk}, \mathsf{ctxt}'_x)$:
  - $x \cdot \mathsf{sk} \leftarrow \mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}'_x)$.
  - Recover x from $x \cdot \mathsf{sk}$.
- Homomorphic Addition: $\mathsf{Eval}'(\mathsf{pk}', +, \mathsf{ctxt}'_x, \mathsf{ctxt}'_y)$:
  $\mathsf{ctxt}'_x + \mathsf{ctxt}'_y (= \mathsf{ctxt}_{x \cdot \mathsf{sk}} + \mathsf{ctxt}_{y \cdot \mathsf{sk}} = \mathsf{ctxt}_{(x+y) \cdot \mathsf{sk}} = \mathsf{ctxt}'_{x+y})$
- Homomorphic Multiplication: $\mathsf{Eval}'(\mathsf{pk}', \times, \mathsf{ctxt}'_x, \mathsf{ctxt}'_y)$
  - Let $L(\cdot) = \mathsf{Dec}(\cdot, \mathsf{ctxt}'_x) = \mathsf{Dec}(\cdot, \mathsf{ctxt}_{x \cdot \mathsf{sk}})$.
  - Output $\mathsf{ctxt}_{L(y \cdot \mathsf{sk})} = L(\mathsf{ctxt}'_y) = L(\mathsf{ctxt}_{y \cdot \mathsf{sk}})$.
  - (By $L(y \cdot \mathsf{sk}) = \mathsf{Dec}(y \cdot \mathsf{sk}, \mathsf{ctxt}_{x \cdot \mathsf{sk}}) = x \cdot y \cdot \mathsf{sk}$, we get $\mathsf{ctxt}'_{x \cdot y} = \mathsf{ctxt}_{x \cdot y \cdot \mathsf{sk}}$.)

Adding Noise for Security from LWE

- Recall that Π is insecure, so does Π'.
- To get security, as in Regev's and Dual-Regev encryption, add noise to keys and/or ciphertexts.
- Noisy linear decryption: $\mathsf{Dec}(\cdot, \mathsf{ctxt}_x) \approx L_{\mathsf{ctxt}_x}(\cdot)$ where $L_{\mathsf{ctxt}_x}(\mathsf{sk}) = x + \text{error}$.
- Error accumulation: Suppose ctxt_x has error e_x and ctxt_y has error e_y
  - $\mathsf{ctxt}_x + \mathsf{ctxt}_y$ has error $e_x + e_y$.
  - $a \cdot \mathsf{ctxt}_x$ has error $a \cdot e_x$.

To apply the transform $\Pi \to \Pi'$, need to tackle two challenges:

- $\mathsf{Dec}(\cdot, \mathsf{ctxt}_x)$ is non-linear $\implies$ cannot homomorphically evaluate.
- $L_{\mathsf{ctxt}_{x \cdot \mathsf{sk}}}$ has large coefficients $\implies$ $L_{\mathsf{ctxt}_{x \cdot \mathsf{sk}}}(\mathsf{ctxt}_{y \cdot \mathsf{sk}})$ has large error.

Adding Noise for Security from LWE

- Recall that Π is insecure, so does Π'.
- To get security, as in Regev's and Dual-Regev encryption, add noise to keys and/or ciphertexts.
- Noisy linear decryption: $\mathsf{Dec}(\cdot, \mathsf{ctxt}_x) \approx L_{\mathsf{ctxt}_x}(\cdot)$ where $L_{\mathsf{ctxt}_x}(\mathsf{sk}) = x + \text{error}$.
- Error accumulation: Suppose ctxt_x has error e_x and ctxt_y has error e_y
  - $\mathsf{ctxt}_x + \mathsf{ctxt}_y$ has error $e_x + e_y$.
  - $a \cdot \mathsf{ctxt}_x$ has error $a \cdot e_x$.

To apply the transform $\Pi \to \Pi'$, need to tackle two challenges:

- $\mathsf{Dec}(\cdot, \mathsf{ctxt}_x)$ is non-linear $\implies$ cannot homomorphically evaluate.
- $L_{\mathsf{ctxt}_{x \cdot \mathsf{sk}}}$ has large coefficients $\implies$ $L_{\mathsf{ctxt}_{x \cdot \mathsf{sk}}}(\mathsf{ctxt}_{y \cdot \mathsf{sk}})$ has large error.

Adding Noise for Security from LWE

- Recall that Π is insecure, so does Π'.
- To get security, as in Regev's and Dual-Regev encryption, add noise to keys and/or ciphertexts.
- Noisy linear decryption: $\mathsf{Dec}(\cdot, \mathsf{ctxt}_x) \approx L_{\mathsf{ctxt}_x}(\cdot)$ where $L_{\mathsf{ctxt}_x}(\mathsf{sk}) = x + \text{error}$.
- Error accumulation: Suppose ctxt_x has error e_x and ctxt_y has error e_y
  - $\mathsf{ctxt}_x + \mathsf{ctxt}_y$ has error $e_x + e_y$.
  - $a \cdot \mathsf{ctxt}_x$ has error $a \cdot e_x$.

To apply the transform $\Pi \to \Pi'$, need to tackle two challenges:

- $\mathsf{Dec}(\cdot, \mathsf{ctxt}_x)$ is non-linear $\implies$ cannot homomorphically evaluate.
- $L_{\mathsf{ctxt}_{x \cdot \mathsf{sk}}}$ has large coefficients $\implies$ $L_{\mathsf{ctxt}_{x \cdot \mathsf{sk}}}(\mathsf{ctxt}_{y \cdot \mathsf{sk}})$ has large error.

Adding Noise for Security from LWE

- Recall that Π is insecure, so does Π'.
- To get security, as in Regev's and Dual-Regev encryption, add noise to keys and/or ciphertexts.
- Noisy linear decryption: $\mathrm{Dec}(\cdot, \mathrm{ctxt}_x) \approx L_{\mathrm{ctxt}_x}(\cdot)$ where $L_{\mathrm{ctxt}_x}(\mathrm{sk}) = x + \text{error}$.
- Error accumulation: Suppose ctxt_x has error e_x and ctxt_y has error e_y
  - $\mathrm{ctxt}_x + \mathrm{ctxt}_y$ has error $e_x + e_y$.
  - $a \cdot \mathrm{ctxt}_x$ has error $a \cdot e_x$.

To apply the transform $\Pi \to \Pi'$, need to tackle two challenges:

- $\mathrm{Dec}(\cdot, \mathrm{ctxt}_x)$ is non-linear $\implies$ cannot homomorphically evaluate.
- $L_{\mathrm{ctxt}_{x \cdot \mathrm{sk}}}$ has large coefficients $\implies$ $L_{\mathrm{ctxt}_{x \cdot \mathrm{sk}}}(\mathrm{ctxt}_{y \cdot \mathrm{sk}})$ has large error.

Handling Large-Scalar Multiplication

We construct another scheme Π'' from the (now noisy) schemes Π' and Π:

- Let $\mathsf{ctxt}''_x = (\mathsf{ctxt}'_x, \mathsf{ctxt}'_{x\cdot 2}, \ldots, \mathsf{ctxt}'_{x\cdot 2^\ell})$ where $\ell = \lfloor \log q \rfloor$ and each component has error e.
- To compute $\mathsf{ctxt}'_{a\cdot x}$ from $a \in \mathbb{Z}_q$ and ctxt'_x:
  - Write down binary decomposition $a = \sum_{i=0}^{\ell} a_i \cdot 2^i$.
  - Output $\mathsf{ctxt}'_{a\cdot x} := \sum_{i=0}^{\ell} a_i \cdot \mathsf{ctxt}'_{x\cdot 2^i}$.
  - Note: $\mathsf{ctxt}'_{a\cdot x}$ has error at most $e \cdot \log q$.
- Repeat the above for $a \cdot 2 \bmod q, \ldots, a \cdot 2^\ell \bmod q$ to get

$$\mathsf{ctxt}''_{a\cdot x} = (\mathsf{ctxt}'_{a\cdot x}, \mathsf{ctxt}'_{a\cdot x\cdot 2}, \ldots, \mathsf{ctxt}'_{a\cdot x\cdot 2^\ell})$$

Handling Large-Scalar Multiplication

We construct another scheme Π'' from the (now noisy) schemes Π' and Π:

- Let $\mathrm{ctxt}''_x = (\mathrm{ctxt}'_x, \mathrm{ctxt}'_{x\cdot 2}, \ldots, \mathrm{ctxt}'_{x\cdot 2^\ell})$ where $\ell = \lfloor \log q \rfloor$ and each component has error e.
- To compute $\mathrm{ctxt}'_{a\cdot x}$ from $a \in \mathbb{Z}_q$ and ctxt'_x:
  - Write down binary decomposition $a = \sum_{i=0}^{\ell} a_i \cdot 2^i$.
  - Output $\mathrm{ctxt}'_{a\cdot x} := \sum_{i=0}^{\ell} a_i \cdot \mathrm{ctxt}'_{x\cdot 2^i}$.
  - Note: $\mathrm{ctxt}'_{a\cdot x}$ has error at most $e \cdot \log q$.
- Repeat the above for $a \cdot 2 \bmod q, \ldots, a \cdot 2^\ell \bmod q$ to get

$$\mathrm{ctxt}''_{a\cdot x} = (\mathrm{ctxt}'_{a\cdot x}, \mathrm{ctxt}'_{a\cdot x\cdot 2}, \ldots, \mathrm{ctxt}'_{a\cdot x\cdot 2^\ell})$$

Handling Large-Scalar Multiplication

We construct another scheme Π'' from the (now noisy) schemes Π' and Π:

- Let $\mathsf{ctxt}''_x = (\mathsf{ctxt}'_x, \mathsf{ctxt}'_{x\cdot 2}, \ldots, \mathsf{ctxt}'_{x\cdot 2^\ell})$ where $\ell = \lfloor \log q \rfloor$ and each component has error e.
- To compute $\mathsf{ctxt}'_{a\cdot x}$ from $a \in \mathbb{Z}_q$ and ctxt'_x:
  - Write down binary decomposition $a = \sum_{i=0}^{\ell} a_i \cdot 2^i$.
  - Output $\mathsf{ctxt}'_{a\cdot x} := \sum_{i=0}^{\ell} a_i \cdot \mathsf{ctxt}'_{x\cdot 2^i}$.
  - Note: $\mathsf{ctxt}'_{a\cdot x}$ has error at most $e \cdot \log q$.
- Repeat the above for $a \cdot 2 \bmod q, \ldots, a \cdot 2^\ell \bmod q$ to get

$$\mathsf{ctxt}''_{a\cdot x} = (\mathsf{ctxt}'_{a\cdot x}, \mathsf{ctxt}'_{a\cdot x\cdot 2}, \ldots, \mathsf{ctxt}'_{a\cdot x\cdot 2^\ell})$$

Supporting Homomorphic Multiplication

- We revisit the transform $\Pi \to \Pi'$ and apply it to Π''.
- To recap: $\mathsf{ctxt}''_x = (\mathsf{ctxt}_{x \cdot \mathsf{sk}_i \cdot 2^j})_{i=1,j=0}^{n,\ell}$ and $\mathsf{ctxt}''_y = (\mathsf{ctxt}_{y \cdot \mathsf{sk}_i \cdot 2^j})_{i=1,j=0}^{n,\ell}$.
- To compute $\mathsf{ctxt}''_{x \cdot y} = (\mathsf{ctxt}_{x \cdot y \cdot \mathsf{sk}_i \cdot 2^j})_{i=1,j=0}^{n,\ell}$, for each (i,j):
  - Let $L_{i,j}(\cdot) \approx \mathsf{Dec}(\cdot, \mathsf{ctxt}_{x \cdot \mathsf{sk}_i \cdot 2^j})$
  - Write $L_{i,j}(\mathsf{sk}) = \sum_h a_h \cdot \mathsf{sk}_h$ where $a_h = \sum_k^{\ell} a_{h,k} \cdot 2^k$.
  - Compute $\mathsf{ctxt}_{x \cdot y \cdot \mathsf{sk}_i \cdot 2^j}$ by

$$\begin{aligned}
\sum_h \sum_k a_{h,k} \cdot \mathsf{ctxt}_{y \cdot \mathsf{sk}_h \cdot 2^k} &= \mathsf{ctxt}_{y \cdot \sum_h \sum_k a_{h,k} \cdot \mathsf{sk}_h \cdot 2^k} \\
&= \mathsf{ctxt}_{y \cdot L_{i,j}(\mathsf{sk})} \\
&\approx \mathsf{ctxt}_{y \cdot \mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}_{x \cdot \mathsf{sk}_i \cdot 2^j})} \\
&= \mathsf{ctxt}_{x \cdot y \cdot \mathsf{sk}_i \cdot 2^j}
\end{aligned}$$

- If ctxt''_x and ctxt''_y have error e, then $\mathsf{ctxt}''_{x \cdot y}$ has error roughly $e \cdot n \cdot \log q$.

Bootstrapping: Bring Down the Noise

- In Π'', evaluating a depth-d circuit f on ctxt_x brings noise level from e to $\approx e \cdot (n \cdot \log q)^d$.
- If d is too large, the noise blows up and decryption of $\mathsf{ctxt}_{f(x)}$ will fail.
- Gentry's "Bootstrapping" technique brings down the noise level of $\mathsf{ctxt}_{f(x)}$ (if it hasn't blown up yet):
  - Given $\mathsf{ctxt}_{\mathsf{sk}}$ with low noise level.
  - Evaluate exact decryption $\mathsf{Dec}(\cdot, \mathsf{ctxt}_{f(x)})$ homomorphically on $\mathsf{ctxt}_{\mathsf{sk}}$.
  - Since $\mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}_{f(x)}) = f(x)$, we get another ciphertext $\widehat{\mathsf{ctxt}}_{f(x)}$.
  - Noise level of $\widehat{\mathsf{ctxt}}_{f(x)}$ only depends on noise level of $\mathsf{ctxt}_{\mathsf{sk}}$ and complexity of $\mathsf{Dec}(\cdot, \mathsf{ctxt}_{f(x)})$, but not noise level of $\mathsf{ctxt}_{f(x)}$.
  - Choose parameters so that the noise level after bootstrapping is low enough to perform some computation $\implies$ Can perform arbitrary computation.

Bootstrapping: Bring Down the Noise

- In Π'', evaluating a depth-d circuit f on ctxt_x brings noise level from e to $\approx e \cdot (n \cdot \log q)^d$.
- If d is too large, the noise blows up and decryption of $\mathrm{ctxt}_{f(x)}$ will fail.
- Gentry's "Bootstrapping" technique brings down the noise level of $\mathrm{ctxt}_{f(x)}$ (if it hasn't blown up yet):
  - Given $\mathrm{ctxt}_{\mathrm{sk}}$ with low noise level.
  - Evaluate exact decryption $\mathrm{Dec}(\cdot, \mathrm{ctxt}_{f(x)})$ homomorphically on $\mathrm{ctxt}_{\mathrm{sk}}$.
  - Since $\mathrm{Dec}(\mathrm{sk}, \mathrm{ctxt}_{f(x)}) = f(x)$, we get another ciphertext $\widehat{\mathrm{ctxt}}_{f(x)}$.
  - Noise level of $\widehat{\mathrm{ctxt}}_{f(x)}$ only depends on noise level of $\mathrm{ctxt}_{\mathrm{sk}}$ and complexity of $\mathrm{Dec}(\cdot, \mathrm{ctxt}_{f(x)})$, but not noise level of $\mathrm{ctxt}_{f(x)}$.
  - Choose parameters so that the noise level after bootstrapping is low enough to perform some computation $\implies$ Can perform arbitrary computation.

Bootstrapping: Bring Down the Noise

- In Π'', evaluating a depth-d circuit f on ctxt_x brings noise level from e to $\approx e \cdot (n \cdot \log q)^d$.
- If d is too large, the noise blows up and decryption of $\mathsf{ctxt}_{f(x)}$ will fail.
- Gentry's "Bootstrapping" technique brings down the noise level of $\mathsf{ctxt}_{f(x)}$ (if it hasn't blown up yet):
  - Given $\mathsf{ctxt}_{\mathsf{sk}}$ with low noise level.
  - Evaluate exact decryption $\mathsf{Dec}(\cdot, \mathsf{ctxt}_{f(x)})$ homomorphically on $\mathsf{ctxt}_{\mathsf{sk}}$.
  - Since $\mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}_{f(x)}) = f(x)$, we get another ciphertext $\hat{\mathsf{ctxt}}_{f(x)}$.
  - Noise level of $\hat{\mathsf{ctxt}}_{f(x)}$ only depends on noise level of $\mathsf{ctxt}_{\mathsf{sk}}$ and complexity of $\mathsf{Dec}(\cdot, \mathsf{ctxt}_{f(x)})$, but not noise level of $\mathsf{ctxt}_{f(x)}$.
  - Choose parameters so that the noise level after bootstrapping is low enough to perform some computation $\Longrightarrow$ Can perform arbitrary computation.

Bootstrapping: Bring Down the Noise

- In Π'', evaluating a depth-d circuit f on ctxt_x brings noise level from e to $\approx e \cdot (n \cdot \log q)^d$.
- If d is too large, the noise blows up and decryption of $\mathsf{ctxt}_{f(x)}$ will fail.
- Gentry's "Bootstrapping" technique brings down the noise level of $\mathsf{ctxt}_{f(x)}$ (if it hasn't blown up yet):
  - Given $\mathsf{ctxt}_{\mathsf{sk}}$ with low noise level.
  - Evaluate exact decryption $\mathsf{Dec}(\cdot, \mathsf{ctxt}_{f(x)})$ homomorphically on $\mathsf{ctxt}_{\mathsf{sk}}$.
  - Since $\mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}_{f(x)}) = f(x)$, we get another ciphertext $\widehat{\mathsf{ctxt}}_{f(x)}$.
  - Noise level of $\widehat{\mathsf{ctxt}}_{f(x)}$ only depends on noise level of $\mathsf{ctxt}_{\mathsf{sk}}$ and complexity of $\mathsf{Dec}(\cdot, \mathsf{ctxt}_{f(x)})$, but not noise level of $\mathsf{ctxt}_{f(x)}$.
  - Choose parameters so that the noise level after bootstrapping is low enough to perform some computation $\implies$ Can perform arbitrary computation.

Bootstrapping: Bring Down the Noise

- In Π'', evaluating a depth-d circuit f on ctxt_x brings noise level from e to $\approx e \cdot (n \cdot \log q)^d$.
- If d is too large, the noise blows up and decryption of $\mathsf{ctxt}_{f(x)}$ will fail.
- Gentry's "Bootstrapping" technique brings down the noise level of $\mathsf{ctxt}_{f(x)}$ (if it hasn't blown up yet):
  - Given $\mathsf{ctxt}_{\mathsf{sk}}$ with low noise level.
  - Evaluate exact decryption $\mathsf{Dec}(\cdot, \mathsf{ctxt}_{f(x)})$ homomorphically on $\mathsf{ctxt}_{\mathsf{sk}}$.
  - Since $\mathsf{Dec}(\mathsf{sk}, \mathsf{ctxt}_{f(x)}) = f(x)$, we get another ciphertext $\widehat{\mathsf{ctxt}}_{f(x)}$.
  - Noise level of $\widehat{\mathsf{ctxt}}_{f(x)}$ only depends on noise level of $\mathsf{ctxt}_{\mathsf{sk}}$ and complexity of $\mathsf{Dec}(\cdot, \mathsf{ctxt}_{f(x)})$, but not noise level of $\mathsf{ctxt}_{f(x)}$.
  - Choose parameters so that the noise level after bootstrapping is low enough to perform some computation $\implies$ Can perform arbitrary computation.

Fully Homomorphic Encryption from Learning with Errors

(The Gentry-Sahai-Waters (GSW) Construction)

Gadget Matrix

- Let $\ell = \lfloor \log q \rfloor$. Define the "gadget matrix" for mapping binary representations to q-ary ones:

$$\mathbf{G} = \begin{pmatrix} 1 & 2 & \ldots & 2^\ell & & & & & & & & \\ & & & & 1 & 2 & \ldots & 2^\ell & & & & \\ & & & & & & & & \ddots & & & \\ & & & & & & & & & 1 & 2 & \ldots & 2^\ell \end{pmatrix}$$

- Denote binary-decomposition operator by $\mathbf{G}^{-1}(\cdot)$ (not a matrix!), e.g. if $n = 2$ and $q = 7$ then

$$\mathbf{G} = \begin{pmatrix} 1 & 2 & 4 & & & \\ & & & 1 & 2 & 4 \end{pmatrix} \qquad \text{and} \qquad \mathbf{G}^{-1}\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

- Note that for any $\mathbf{X} \in \mathbb{Z}_q^{n \times m}$, $m = n \cdot (\ell + 1)$, we have $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{X}) = \mathbf{X} \bmod q$.

Basic Algorithms

- Key Generation:
  - Sample $\bar{\mathbf{A}} \leftarrow\$ \mathbb{Z}_q^{(n-1)\times m}$, $\mathbf{s} \leftarrow\$ \mathbb{Z}_q^{n-1}$, $\mathbf{e} \leftarrow\$ \mathbb{Z}_\beta^m$
  - Compute $\mathbf{b}^\top = \mathbf{s}^\top \cdot \bar{\mathbf{A}} + \mathbf{e}^\top \bmod q$
  - Output $\mathsf{pk} := \mathbf{A} := \begin{pmatrix} \bar{\mathbf{A}} \\ \mathbf{b}^\top \end{pmatrix}$, $\mathsf{sk} = \mathbf{s}$.
  - Note that $\begin{pmatrix} -\mathbf{s}^\top & 1 \end{pmatrix} \cdot \mathbf{A} = \mathbf{e}^\top \approx \mathbf{0}^\top \bmod q$
- Encryption of $x \in \{0, 1\}$: $\mathbf{R} \leftarrow\$ \mathbb{Z}_\beta^{m\times m}$, $\mathsf{ctxt} := \mathbf{C} := \mathbf{A} \cdot \mathbf{R} + x \cdot \mathbf{G} \bmod q$
- Decryption:

$$\bar{\mathbf{x}} := \begin{pmatrix} -\mathbf{s}^\top & 1 \end{pmatrix} \cdot \mathbf{C}$$

$$\bar{x} := \text{last entry of } \bar{\mathbf{x}}$$

$$x = \begin{cases} 0 & |\bar{x}| < q/4 \\ 1 & \text{otherwise} \end{cases}$$

Decryption Correctness

Public and secret keys satisfy:

$$\begin{pmatrix}-\mathbf{s}^{\mathsf{T}} & 1\end{pmatrix} \cdot \mathbf{A} = \mathbf{e}^{\mathsf{T}} \bmod q.$$

Ciphertexts are of the form:

$$\mathbf{C} = \mathbf{A} \cdot \mathbf{R} + x \cdot \mathbf{G} \bmod q.$$

Decryption:

$$\begin{aligned}
\bar{\mathbf{x}} &= \begin{pmatrix}-\mathbf{s}^{\mathsf{T}} & 1\end{pmatrix} \cdot \mathbf{C} \\
&= \mathbf{e}^{\mathsf{T}} \cdot \mathbf{R} + x \cdot \begin{pmatrix}-\mathbf{s}^{\mathsf{T}} & 1\end{pmatrix} \cdot \mathbf{G} \bmod q \\
\bar{x} &= \underbrace{\mathbf{e}^{\mathsf{T}} \cdot \mathbf{r}}_{\text{short}} + x \cdot 2^{\ell} \bmod q
\end{aligned}$$

† Note $2^{\ell} \approx q/2$.

† If $\left|\mathbf{e}^{\mathsf{T}} \cdot \mathbf{r}\right| < q/4$ then decryption is correct.

IND-CPA-Security

Like Regev's PKE and left as exercise.

Homomorphic Evaluation

We first describe homomorphic addition and multiplication:

- † Homomorphic Addition: $\mathbf{C}_0 + \mathbf{C}_1 \bmod q$
- † Homomorphic Multiplication: $\mathbf{C}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) \bmod q$

Why it (almost) works?

$$\mathbf{C}_0 + \mathbf{C}_1 = (\mathbf{A} \cdot \mathbf{R}_0 + x_0 \cdot \mathbf{G}) + (\mathbf{A} \cdot \mathbf{R}_1 + x_1 \cdot \mathbf{G}) = \mathbf{A} \cdot (\underbrace{\mathbf{R}_0 + \mathbf{R}_1}_{\text{short}}) + (x_0 + x_1) \cdot \mathbf{G} \bmod q$$

$$\begin{aligned}
\mathbf{C}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) &= (\mathbf{A} \cdot \mathbf{R}_0 + x_0 \cdot \mathbf{G}) \cdot \mathbf{G}^{-1}(\mathbf{C}_1) \\
&= \mathbf{A} \cdot \mathbf{R}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) + x_0 \cdot \mathbf{C}_1 \\
&= \mathbf{A} \cdot \mathbf{R}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) + x_0 \cdot (\mathbf{A} \cdot \mathbf{R}_1 + x_1 \cdot \mathbf{G}) \\
&= \mathbf{A} \cdot (\underbrace{\mathbf{R}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) + x_0 \cdot \mathbf{R}_1}_{\text{short}}) + x_0 \cdot x_1 \cdot \mathbf{G} \bmod q
\end{aligned}$$

Homomorphic Evaluation

We first describe homomorphic addition and multiplication:

- † Homomorphic Addition: $\mathbf{C}_0 + \mathbf{C}_1 \bmod q$
- † Homomorphic Multiplication: $\mathbf{C}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) \bmod q$

Why it (almost) works?

$$\mathbf{C}_0 + \mathbf{C}_1 = (\mathbf{A} \cdot \mathbf{R}_0 + x_0 \cdot \mathbf{G}) + (\mathbf{A} \cdot \mathbf{R}_1 + x_1 \cdot \mathbf{G}) = \mathbf{A} \cdot (\underbrace{\mathbf{R}_0 + \mathbf{R}_1}_{\text{short}}) + (x_0 + x_1) \cdot \mathbf{G} \bmod q$$

$$\begin{aligned}\mathbf{C}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) &= (\mathbf{A} \cdot \mathbf{R}_0 + x_0 \cdot \mathbf{G}) \cdot \mathbf{G}^{-1}(\mathbf{C}_1) \\ &= \mathbf{A} \cdot \mathbf{R}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) + x_0 \cdot \mathbf{C}_1 \\ &= \mathbf{A} \cdot \mathbf{R}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) + x_0 \cdot (\mathbf{A} \cdot \mathbf{R}_1 + x_1 \cdot \mathbf{G}) \\ &= \mathbf{A} \cdot (\underbrace{\mathbf{R}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) + x_0 \cdot \mathbf{R}_1}_{\text{short}}) + x_0 \cdot x_1 \cdot \mathbf{G} \bmod q\end{aligned}$$

Homomorphic Evaluation

We first describe homomorphic addition and multiplication:

- † Homomorphic Addition: $\mathbf{C}_0 + \mathbf{C}_1 \bmod q$
- † Homomorphic Multiplication: $\mathbf{C}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) \bmod q$

Why it (almost) works?

$$\mathbf{C}_0 + \mathbf{C}_1 = (\mathbf{A} \cdot \mathbf{R}_0 + x_0 \cdot \mathbf{G}) + (\mathbf{A} \cdot \mathbf{R}_1 + x_1 \cdot \mathbf{G}) = \mathbf{A} \cdot (\underbrace{\mathbf{R}_0 + \mathbf{R}_1}_{\text{short}}) + (x_0 + x_1) \cdot \mathbf{G} \bmod q$$

$$\begin{aligned}
\mathbf{C}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) &= (\mathbf{A} \cdot \mathbf{R}_0 + x_0 \cdot \mathbf{G}) \cdot \mathbf{G}^{-1}(\mathbf{C}_1) \\
&= \mathbf{A} \cdot \mathbf{R}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) + x_0 \cdot \mathbf{C}_1 \\
&= \mathbf{A} \cdot \mathbf{R}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) + x_0 \cdot (\mathbf{A} \cdot \mathbf{R}_1 + x_1 \cdot \mathbf{G}) \\
&= \mathbf{A} \cdot (\underbrace{\mathbf{R}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) + x_0 \cdot \mathbf{R}_1}_{\text{short}}) + x_0 \cdot x_1 \cdot \mathbf{G} \bmod q
\end{aligned}$$

Homomorphic Evaluation

We first describe homomorphic addition and multiplication:

- † Homomorphic Addition: $\mathbf{C}_0 + \mathbf{C}_1 \bmod q$
- † Homomorphic Multiplication: $\mathbf{C}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) \bmod q$

Why it (almost) works?

$$\mathbf{C}_0 + \mathbf{C}_1 = (\mathbf{A} \cdot \mathbf{R}_0 + x_0 \cdot \mathbf{G}) + (\mathbf{A} \cdot \mathbf{R}_1 + x_1 \cdot \mathbf{G}) = \mathbf{A} \cdot (\underbrace{\mathbf{R}_0 + \mathbf{R}_1}_{\text{short}}) + (\underbrace{x_0 + x_1}_{\text{Issue: } 1+1=2 \notin \{0,1\}}) \cdot \mathbf{G} \bmod q$$

$$\begin{aligned}
\mathbf{C}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) &= (\mathbf{A} \cdot \mathbf{R}_0 + x_0 \cdot \mathbf{G}) \cdot \mathbf{G}^{-1}(\mathbf{C}_1) \\
&= \mathbf{A} \cdot \mathbf{R}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) + x_0 \cdot \mathbf{C}_1 \\
&= \mathbf{A} \cdot \mathbf{R}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) + x_0 \cdot (\mathbf{A} \cdot \mathbf{R}_1 + x_1 \cdot \mathbf{G}) \\
&= \mathbf{A} \cdot (\underbrace{\mathbf{R}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) + x_0 \cdot \mathbf{R}_1}_{\text{short}}) + x_0 \cdot x_1 \cdot \mathbf{G} \bmod q
\end{aligned}$$

Homomorphic Evaluation

- Fix: Instead of $+$ and $\times$, we use NAND.
- NAND is functionally complete for Boolean functions.
- Observe that $\mathrm{NAND}(x_0, x_1) = 1 - x_0 \cdot x_1$ (over $\mathbb{Z}$).
- Homomorphic NAND: $\mathbf{G} - \mathbf{C}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) \bmod q$
  ($\mathbf{G}$ is an encryption of 1 with no noise.)

Why it works?

$$\begin{aligned} \mathbf{G} - \mathbf{C}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) &= \mathbf{G} - \mathbf{A} \cdot (\mathbf{R}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) + x_0 \cdot \mathbf{R}_1) - x_0 \cdot x_1 \cdot \mathbf{G} \\ &= \mathbf{A} \cdot (\underbrace{-\mathbf{R}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) - x_0 \cdot \mathbf{R}_1}_{\text{short}}) + (1 - x_0 \cdot x_1) \cdot \mathbf{G} \bmod q \end{aligned}$$

Homomorphic Evaluation

- Fix: Instead of $+$ and $\times$, we use NAND.
- NAND is functionally complete for Boolean functions.
- Observe that $\mathrm{NAND}(x_0, x_1) = 1 - x_0 \cdot x_1$ (over $\mathbb{Z}$).
- Homomorphic NAND: $\mathbf{G} - \mathbf{C}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) \bmod q$
  ($\mathbf{G}$ is an encryption of 1 with no noise.)

Why it works?

$$\begin{aligned}\mathbf{G} - \mathbf{C}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) &= \mathbf{G} - \mathbf{A} \cdot (\mathbf{R}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) + x_0 \cdot \mathbf{R}_1) - x_0 \cdot x_1 \cdot \mathbf{G} \\ &= \mathbf{A} \cdot (\underbrace{-\mathbf{R}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) - x_0 \cdot \mathbf{R}_1}_{\text{short}}) + (1 - x_0 \cdot x_1) \cdot \mathbf{G} \bmod q\end{aligned}$$

Homomorphic Evaluation

- Fix: Instead of $+$ and $\times$, we use NAND.
- NAND is functionally complete for Boolean functions.
- Observe that $\text{NAND}(x_0, x_1) = 1 - x_0 \cdot x_1$ (over $\mathbb{Z}$).
- Homomorphic NAND: $\mathbf{G} - \mathbf{C}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) \bmod q$
  ($\mathbf{G}$ is an encryption of 1 with no noise.)

Why it works?

$$\begin{aligned}\mathbf{G} - \mathbf{C}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) &= \mathbf{G} - \mathbf{A} \cdot \left(\mathbf{R}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) + x_0 \cdot \mathbf{R}_1\right) - x_0 \cdot x_1 \cdot \mathbf{G} \\ &= \mathbf{A} \cdot \big(\underbrace{-\mathbf{R}_0 \cdot \mathbf{G}^{-1}(\mathbf{C}_1) - x_0 \cdot \mathbf{R}_1}_{\text{short}}\big) + \left(1 - x_0 \cdot x_1\right) \cdot \mathbf{G} \bmod q\end{aligned}$$

Bootstrapping and Circular Security

- To evaluate arbitrary functions, we need bootstrapping,
  i.e. evaluating $\mathsf{Dec}(\cdot, \mathsf{ctxt})$ homomorphically on $\mathsf{ctxt}_{\mathsf{sk}}$.
- Bootstrapping requires including $\mathsf{ctxt}_{\mathsf{sk}}$ in pk.
- This causes the security reduction from LWE to IND-CPA-security to fail.
- To this day, all FHE schemes rely on bootstrapping and a "circular security" assumption:
  The scheme remains IND-CPA-secure even when given $\mathsf{ctxt}_{\mathsf{sk}}$.