CSE208: Advanced Cryptography

Daniele Micciancio

UCSD

Fall 2020
Section 1

Introduction
CSE208: Advanced Cryptography

- Graduate Level Advanced Cryptography
- Prerequisites:
  - CSE207 or equivalent
  - Solid theoretical background, cryptographic definitions, etc.
  - Some programming
CSE208: Advanced Cryptography

- Graduate Level Advanced Cryptography
- Prerequisites:
  - CSE207 or equivalent
  - Solid theoretical background, cryptographic definitions, etc.
  - Some programming

- Past topics: Zero Knowledge, Functional Encryption, Secure Computation, etc.
- Not required: CSE206A (Lattice Algorithms)
Graduate Level Advanced Cryptography

Prerequisites:
- CSE207 or equivalent
- Solid theoretical background, cryptographic definitions, etc.
- Some programming

Past topics: Zero Knowledge, Functional Encryption, Secure Computation, etc.

Not required: CSE206A (Lattice Algorithms)

Reading:
- no textbook
- mostly research papers
- see course webpage, canvas, etc.
Fall 2020 Topic

- Fully Homomorphic Encryption:
  - Encryption schemes that supports the evaluation of arbitrary programs on encrypted inputs

- Applications:
  - secure outsourced computing
  - building block for MPC and more
Brief History of Homomorphic Encryption

- 1978: Rivest, Adleman & Dertouzos posed the problem
- 2009: Gentry 2009 proposed the first candidate solution
- 2010-2020: Work towards more efficient solutions based on standard complexity assumptions (Brakerski, Vaikuntanathan, Gentry, Halevi, Smart, ...)

### Project Info

- Ring LWE
- ANT
Software libraries

- IBM HElib (Halevi & Shoup)
- Microsoft SEAL
- NJIT/Duality PALISADE (Rohloff, Cousins & Polyakov)
- Functional Lattice Cryptography LoL (Crockett & Peikert)
- Fastest FHE of the West FHEW (Ducas & Micciancio)
- FHE over the Torus TFHE (Chillotti, Gama, Georgieva & Izabachene)
- Approximate FHE HEAAN (Cheon, Kim, Kim & Song)
Software libraries

- IBM HElib (Halevi & Shoup)
- Microsoft SEAL
- NJIT/Duality PALISADE (Rohloff, Cousins & Polyakov)
- Functional Lattice Cryptography LoL (Crockett & Peikert)
- Fastest FHE of the West FHEW (Ducas & Micciancio)
- FHE over the Torus TFHE (Chillotti, Gama, Georgieva & Izabachene)
- Approximate FHE HEAAN (Cheon, Kim, Kim & Song)

In the News:
- February 21, 2019: Microsoft SEAL open source homomorphic encryption library gets even better for .NET developers!
- June 4, 2020: IBM releases FHE toolkit for MacOS and iOS; Linux and Android Coming Soon
Homework and Evaluation

- Homework assignments:
  - 3 assignments, due within one week from assignment date
  - Cover theoretical/mathematical topics
Homework and Evaluation

- **Homework assignments:**
  - 3 assignments, due within one week from assignment date
  - Cover theoretical/mathematical topics

- **Small Project:**
  - Goal: Try to use one of the many HE libraries
  - Not much coding, but you will have to write and compile a few lines of code
  - Evaluated primarily based on written report
Administrivia:

- **Course webpage**: [http://cseweb.ucsd.edu/classes/fa20/](http://cseweb.ucsd.edu/classes/fa20/)
  - general course information
  - pointers to papers and other reading material
- **Canvas**:
  - recording of lectures
  - homework distribution/collection
  - grades
  - discussion board
Course Schedule (Tentative)

Week 1: Introduction and Definition

- FHE Definition
- Gentry’s Boostrapping theorem
- Homework 1 out

Week 2-4: Fundamental techniques based on general lattices

- LWE encryption
- Linear Homomorphic computations
- Key Switching and Proxy re-encryption
- Nested encryption and homomorphic multiplication
- Ciphertext Tensoring and homomorphic multiplication
- Homomorphic Decryption and Bootstrapping algorithms
- Homework 2 out
Week 5: Algebraic Number Theory

- I really hope you like math!
- Homework 3 out

Week 6-10: Efficient FHE from Ring LWE

- Message packing techniques
- Linear transformations on structured matrices
- Other FHE schemes: GHS, BFV, FHEW, AP13, TFHE, CKKS ...
Section 2

Defining FHE
Public Key Encryption

\[
\text{PKE}(\text{Gen}, \text{Enc}, \text{Dec}) \\
\text{Gen}: () \to (pk, sk) \\
\text{Enc}: (pk, m) \to c \\
\text{Dec}: (sk, c) \to m
\]
Correctness of PKE

For every \((sk, pk) \leftarrow \text{Gen}()\) and \(m \leftarrow [M], r \leftarrow [R]:\)

\[
\text{Dec}(sk, \text{Enc}(pk, m; r)) = m
\]
Chosen Plaintext Attack (CPA) security

- Ciphertext Indistinguishability under Chosen Plaintext Attack
- Experiment:

\[
\text{INDCPA}_\text{game}(b:\{0,1\})
\]

\[
(sk, pk) \leftarrow \text{Gen}()
\]

\[
A(pk) \rightarrow (m_0, m_1)
\]

\[
b' \leftarrow A(\text{Enc}(pk, m_b))
\]

\[
\text{return } b':\{0,1\}
\]

Definition

\[
\text{Adv}(A) = |\Pr(\text{Game}(0) = 1) - \Pr(\text{Game}(1) = 1) |
\]

Definition

An encryption scheme \((\text{Gen}, \text{Enc}, \text{Dec})\) is IND-CPA secure if any polynomial time \(A\) has advantage \(\text{Adv}(A) \approx 0\)
Chosen Plaintext Attack (CPA) security

- Ciphertext Indistinguishability under Chosen Plaintext Attack
- Experiment:

\[
\text{INDCPA}_{\text{game}}(b:\{0, 1\})
\]

\[
(s_k, p_k) \leftarrow \text{Gen}()
\]

\[
A(p_k) \rightarrow (m_0, m_1)
\]

\[
b' \leftarrow A(\text{Enc}(p_k, m_b))
\]

\[
\text{return } b' : \{0, 1\}
\]

**Definition**

\[
\text{Adv}(A) = |\Pr(\text{Game}(0) = 1) - \Pr(\text{Game}(1) = 1)|
\]

**Definition**

An encryption scheme \((\text{Gen}, \text{Enc}, \text{Dec})\) is \textbf{IND-CPA} secure if any polynomial time \(A\) has advantage \(\text{Adv}(A) \approx 0\)
Significance of CPA security

- Adversary can choose messages \( m_0, m_1 \)
  - No assumption about input distribution
  - Adversary may have partial information about messages
  - Adversary may influence the choice of messages

- Ciphertext \( c = \text{Enc}(pk, m_b) \) is computed honestly
  - Adversary cannot tamper with ciphertexts

- Adversary models a passive attacker
Definition of CCA security

**Definition**

An encryption scheme \((\text{Gen}, \text{Enc}, \text{Dec})\) is \textbf{IND-CCA} secure if any polynomial time \(A\) has advantage \(\text{Adv}(A) \approx 0\) in the following game.

\[
\text{Game}(b:\{0,1\}) \\
(\text{sk}, \text{pk}) \leftarrow \text{Gen}() \\
A[D](\text{pk}) \rightarrow (m_0, m_1) \\
c \leftarrow \text{Enc}(\text{pk}, m_b) \\
b' \leftarrow A[D'](c) \\
\text{return} b':\{0,1\}
\]

- \(A[D]\) is an adversary with oracle access to \(D(x) = \text{Dec}(\text{sk}, x)\)

- \(A[D']\) uses a modified oracle (next slide)
IND-CCA1 vs IND-CCA2

There are two variants of CCA security, depending on the type of oracle given to the adversary after receiving the challenge ciphertext:

- **IND-CCA1** security: No decryption oracle after receiving the challenge
  \[ D'(x) = \text{Nil} \]

- **IND-CCA2** security: decrypt any ciphertext, except the challenge \( c \)
  \[
  D'(x) = \begin{cases} 
  \text{Nil} & \text{if } (x \neq c) \\
  \text{Dec}(sk, x) & \text{else}
  \end{cases}
  \]
Significance of CCA security

- Goal: model active attacks, where adversary can tamper with ciphertexts
- Standard notion for regular encryption schemes
- IND-CCA2 theoretically equivalent to non-malleable encryption
  - Any attempt to modify a ciphertext should be detected
Significance of CCA security

- Goal: model active attacks, where adversary can tamper with ciphertexts
- Standard notion for regular encryption schemes
- IND-CCA2 theoretically equivalent to *non-malleable* encryption
  - Any attempt to modify a ciphertext should be detected
- Seems incompatible with homomorphic encryption
  - Ability to modify ciphertexts can be a useful feature
  - Homomorphic encryption is *perfectly malleable*
- We will not consider CCA security
Homomorphic Encryption: first attempt

Assume $f: M \rightarrow M$

$$f(\text{Enc}(pk, m)) = \text{Enc}(pk, f(m))$$

$$\text{Eval}(pk, f, \text{Enc}(pk, m)) = \text{Enc}(pk, f(m))$$
Homomorphic Encryption: second attempt

\[ \text{Dec}(sk, \text{Eval}(pk, f, \text{Enc}(pk, m))) = f(m) \]
Multi-input functions

- Many inputs are encrypted independently

\[ c_1 \leftarrow \text{Enc}(pk, m_1) \]
\[ \ldots \]
\[ c_k \leftarrow \text{Enc}(pk, m_k) \]
Multi-input functions

- Many inputs are encrypted independently
  
  \[ c_1 \leftarrow \text{Enc}(pk, m_1) \]
  
  \[ \ldots \]
  
  \[ c_k \leftarrow \text{Enc}(pk, m_k) \]

- \( k \)-ary function \( f: (m_1, \ldots, m_k) \rightarrow m \)

  \[ \text{Eval}(pk, f, c_1, \ldots, c_k) \]
  
  \[ = \text{Enc}(pk, f(m_1, \ldots, m_k)) \] ???

  \[ \text{Dec}(sk, \text{Eval}(pk, f, c_1, \ldots c_k)) \]
  
  \[ = f(m_1, \ldots, m_k) \]
Multi-key Homomorphic encryption

- Assume multiple users: $P_1, P_2, ...$
- Each user has a key (pair): $P_i : (pk_i, sk_i)$
- Data is encrypted and sent to different users
  
  $$c_1 \leftarrow \text{Enc}(pk_1, m_1)$$
  $$\ldots$$
  $$c_t \leftarrow \text{Enc}(pk_t, m_t)$$

- Users pool data together to perform a joint computation on $c_1, ..., c_t$
Multi-key Homomorphic encryption

- Assume multiple users: $P_1, P_2, ...$
- Each user has a key (pair): $P_i : (pk_i, sk_i)$
- Data is encrypted and sent to different users
  
  $c_1 \leftarrow \text{Enc}(pk_1, m_1)$
  
  $\ldots$
  
  $c_t \leftarrow \text{Enc}(pk_t, m_t)$

- Users pool data together to perform a joint computation on $c_1, ..., c_t$

- Final result is an encryption of $f(m_1, ..., m_t)$ under what key?

  $\text{Eval}(\text{???}, f, c_1, \ldots, c_t) \approx \text{Enc}(\text{???}, f(m_1, \ldots, m_t))$
FHE is a useful and challenging problem already in the single key setting
Restricting Homomorphic Encryption

- FHE is a useful and challenging problem already in the single key setting
- In order to approach the problem we will further restrict it by parametrizing by a set of allowed computations/functions $\text{Func} = \{f: \ldots\}$ where each $f: (M, \ldots, M) \to M$ may take a different number of arguments
Restricting Homomorphic Encryption

- FHE is a useful and challenging problem already in the single key setting.

- In order to approach the problem we will further restrict it by parametrizing by a set of allowed computations/functions $\text{Func} = \{f: \ldots\}$ where each $f: (M, \ldots, M) \rightarrow M$ may take a different number of arguments.

- More generally, one may consider functions $f: (M_1, \ldots, M_k) \rightarrow M$ taking inputs from different sets (types), e.g., $\text{ifThenElse}: (\text{Bool}, \text{Int}, \text{Int}) \rightarrow \text{Int}$. 
Examples and Function Composition

- \((M, +, 0)\): abelian group, e.g., “fixed size” integers (modulo \(N\))
- Addition: \(f(x_1, \ldots, x_t) = x_1 + \ldots + x_t\)
- Scalar multiplication: \(g_a(x) = a \cdot x\)
- Linear combinations: \(h(x_1, \ldots, x_t) = \sum_i 2^{i-1}x_i\)
Examples and Function Composition

- \((M, +, 0)\): abelian group, e.g., “fixed size” integers (modulo \(N\))
- Addition: \(f(x_1, \ldots x_t) = x_1 + \ldots + x_t\)
- Scalar multiplication: \(g_a(x) = a \cdot x\)
- Linear combinations: \(h(x_1, \ldots x_t) = \sum_i 2^{i-1}x_i\)

1-hop, n-hop, multi-hop: can functions \(f\) be composed?

\[ h(x_1, \ldots, x_t) = f(g_1(x_1), \ldots, g_{2^t-1}(x_t)) \]
Correctness of Function Composition

Let $x, y, z \in M$ be messages and $f, g : M \to M$ two functions such that $y = f(x)$ and $z = g(y) = (g \circ f)(x)$.

Assume $(\text{Gen, Enc, Dec, Eval})$ can evaluate $f$ and $g$ correctly:

- $\text{Dec}(sk, \text{Eval}(pk, f, \text{Enc}(pk, x))) = f(x)$
- $\text{Dec}(sk, \text{Eval}(pk, g, \text{Enc}(pk, y))) = g(y)$
Correctness of Function Composition

Let $x, y, z \in M$ be messages and $f, g : M \rightarrow M$ two functions such that $y = f(x)$ and $z = g(y) = (g \circ f)(x)$.

Assume $(Gen, Enc, Dec, Eval)$ can evaluate $f$ and $g$ correctly:

$$\text{Dec}(sk, \text{Eval}(pk, f, \text{Enc}(pk, x))) = f(x)$$
$$\text{Dec}(sk, \text{Eval}(pk, g, \text{Enc}(pk, y))) = g(y)$$

**Question**

*Does it follow that*

\[
\begin{align*}
\text{ctX} & \leftarrow \text{Enc}(pk, x) \\
\text{ctY} & \leftarrow \text{Eval}(pk, f, \text{ctX}) \\
\text{ctZ} & \leftarrow \text{Eval}(pk, g, \text{ctY}) \\
\text{Dec}(sk, \text{ctZ}) & = z
\end{align*}
\]
Formalizing Restricted Composition

- Restrict scheme to a set $\mathcal{F}$ of strongly typed functions:
  $$f : M_1 \times \ldots M_k \rightarrow M_0$$

- $\text{Enc, Dec, Eval}$ are given type information
Formalizing Restricted Composition

- Restrict scheme to a set $\mathcal{F}$ of strongly typed functions:
  \[ f : M_1 \times \ldots M_k \rightarrow M_0 \]

- $\text{Enc}, \text{Dec}, \text{Eval}$ are given type information

- We can use types to bound computation depth:
  - Start from $f : M \rightarrow M$
  - Define $M_i = M$ for $i = 1, \ldots, n$
  - Define $f_i : M_i \rightarrow M_{i+1}$, where $f_i(x) = f(x)$

- $\mathcal{F} = \{ f \}$ allows arbitrary composition

- $\mathcal{F} = \{ f_0 \}$: no composition

- $\mathcal{F} = \{ f_0, f_1, \ldots, f_n \}$: bounded depth composition
(Multi-hop) Correctness Game

State: (initially empty) list $L$ of message-ciphertext pairs

\[
\text{CorrectFHEgame}() = (sk, pk) \leftarrow \text{Gen}()
\]

\[
L \leftarrow []
\]

\[
A[E,F](pk)
\]

\[
(m, c) \leftarrow \text{last}(L)
\]

\[
\text{return } (\text{Dec}(sk, c) \neq m)
\]

\[
E(m) = c \leftarrow \text{Enc}(pk, m)
\]

\[
L \leftarrow L;(m, c)
\]

\[
\text{return } c
\]

\[
F(f, I) = (ms, cs) \leftarrow \text{unzip } L[I]
\]

\[
m \leftarrow f(ms)
\]

\[
c \leftarrow \text{Eval}(pk, f, cs)
\]

\[
L \leftarrow L;(m, c)
\]

\[
\text{return } c
\]
Terminology

Reading papers, you will find references to

- Fully Homomorphic Encryption
- Somewhat Homomorphic Encryption
- Leveled Fully Homomorphic Encryption
- etc.
Reading papers, you will find references to

- Fully Homomorphic Encryption
- Somewhat Homomorphic Encryption
- Leveled Fully Homomorphic Encryption
- etc.

We will use FHE as a catchall term

- Definition is parametrized by a set of functions $\mathcal{F}$
- Functions in $\mathcal{F}$ can be composed only if their types match
- $\mathcal{F}$ is closed under composition
- Can use “phantom” types to limit composition

We will rarely define $\mathcal{F}$ formally, but it is a useful exercise
Security of Homomorphic Encryption

\[
\text{INDCPA}_{\text{game}}(b:\{0,1\})
\]

\[
(sk,pk) \leftarrow \text{Gen}()
\]

\[
A(pk) \to (m_0,m_1)
\]

\[
\text{return } A(\text{Enc}(pk,m_b)) : \{0,1\}
\]

Remark

The IND-CPA security definition depends only on \text{Gen} and \text{Enc}, but not on \text{Dec} (or \text{Eval})

Question

Can the IND-CPA security definition be applied as it is to FHE schemes \((\text{Gen},\text{Enc},\text{Dec},\text{Eval})\)?
Consider the following FHE scheme:

- Let \((\text{Gen,Enc,Dec})\) be IND-CPA secure
- Define \(\text{TrivialFHE} = (\text{Gen,Enc',Dec',Eval})\)

\[
\text{Enc'}(pk, m) = (\text{Enc}(pk, m), [])
\]
\[
\text{Dec'}(sk, (ct, [])) = \text{Dec}(sk, ct)
\]
\[
\text{Dec'}(sk, (ct, [f; fs])) = f(\text{Dec'}(sk, (ct, fs)))
\]
\[
\text{Eval}(pk, f, (ct, [fs])) = (ct, [f; fs])
\]

**Question**

- *Is TrivialFHE a correct FHE scheme?*
- *Is TrivialFHE a secure FHE scheme?*
- *What makes the above scheme “trivial”?*
Compactness

- The TrivialFHE scheme is both correct and secure
- The problem with TrivialFHE is that it is not efficient:
  - Computation is performed by Dec, not Eval!

**Definition**

A FHE scheme is **compact** if the size of ciphertext $ct = \text{Eval}(pk, f, \text{Enc}(pk, m))$ is independent of $\text{Size}(f)$

- Weaker forms of compactness:
  - Ciphertext size may grow logarithmic with $\text{Size}(f)$
  - Ciphertext size may depend on $\text{Depth}(f)$
Function Privacy

\[ f_0(x,y) = x + y \]
\[ f_1(x,y) = y + x \]

\begin{align*}
\text{Game}[A](b: \{0,1\}) \\
&= (sk, pk) \leftarrow \text{Gen}() \\
&= ctX \leftarrow \text{Enc}(pk, x) \\
&= ctY \leftarrow \text{Enc}(pk, y) \\
&= ct \leftarrow \text{Eval}(pk, f_b, ctX, ctY) \\
&= \text{return } A(ct)
\end{align*}

**Question**

Assume \((Gen, Enc, Dec, Eval)\) is a secure FHE scheme. Can an efficient adversary \(A\) recover the bit \(b = \text{Game}[A](b)\)?
Passive Attacks to FHE

Game\[A](b: \{0,1\})

\((sk, pk) \leftarrow Gen()\)

State \leftarrow []

b' \leftarrow A[E, D, F](pk)

return b'

Adversary has access to three stateful oracles:

- Encryption oracle: \(E(m_0, m_1)\)
- Function Evaluation oracle: \(F(f_0, f_1, I)\)
- Decryption oracle: \(D(i)\)
- Joint State: List of message-message-ciphertext triplets \((m_0, m_1, ct)\)
Passive Attack (oracles)

\[
\begin{align*}
E(m_0, m_1) &= ct \leftarrow \text{Enc}(pk, m_b) \\
\text{State} &\leftarrow (\text{State};(m_0, m_1, ct)) \\
\text{return } ct
\end{align*}
\]

\[
\begin{align*}
F(f_0, f_1, I) &= (ms_0, ms_1, cts) \leftarrow \text{unzip State}[I] \\
ct &\leftarrow \text{Eval}(pk, f_b, cts) \\
m_0 &\leftarrow f_0(ms_0) \\
m_1 &\leftarrow f_1(ms_1) \\
\text{State} &\leftarrow \text{State};(m_0, m_1, ct) \\
\text{return } ct
\end{align*}
\]

\[
\begin{align*}
D(i): (m_0, m_1, ct) &\leftarrow \text{State}[i] \\
\text{if } (m_0 \equiv m_1) &\text{ then return Dec}(sk, ct) \\
\text{else return Nil}
\end{align*}
\]
Passive Attack with/without function privacy

- The game we just described guarantees function privacy
- A similar definition without function privacy can be obtained by requiring $f_0 \equiv f_1$ in the function evaluation queries

\[
F'(f, I): (ms_0, ms_1, cts) \leftarrow \text{unzip} \ State[I] \\
ct \leftarrow \text{Eval}(pk, f, cts) \\
m_0 = f(ms_0) \\
m_1 = f(ms_1) \\
State \leftarrow (State; (m_0, m_1, ct))\\n\text{return}\ ct
\]
Example: Circuit Privacy

- Assume messages are single bits \( m: \{0, 1\} \)
- Let \( \text{FHE} = (\text{Gen}, \text{Enc}, \text{Dec}, \text{Eval}) \) a function private FHE scheme supporting \( \text{NAND}(x, y) = \text{not}(x \&\& y) \)
- \( \text{Eval}_C(pk, C, \ldots) \): evaluates boolean circuit \( C: \{0, 1\}^n \rightarrow \{0, 1\} \) one gate at a time using \( \text{Eval}(pk, \text{NAND}, \ldots) \)
- Let \( C_0, C_1: \text{NAND} \) circuits with the same number of inputs and NAND gates
- \( (sk, ps) \leftarrow \text{Gen()} \)
- Let \( xs_0, xs_1 \) be input bits such that \( C_0(xs_0) = C_1(xs_1) \)

**Question**

Are the following two distributions indistinguishable?

\[
(pk, \text{Eval}_C(pk, C_0, \text{Enc}(pk, xs_0)))
\]

\[
(pk, \text{Eval}_C(pk, C_1, \text{Enc}(pk, xs_1)))
\]
Section 3

Bootstrapping
Bootstrapping

- For simplicity: fix message space to \( \{0, 1\} \)

- \( \text{HE} = (\text{Gen}, \text{Enc}, \text{Dec}, \text{Eval}) \)
  - Homomorphic functions: \( \text{Func} = \{ \text{nand} \} \)
  - Supports only bounded computations: \( \text{Depth}(C) < D \)
Bootstrapping

* For simplicity: fix message space to \{0, 1\}

* HE=(Gen, Enc, Dec, Eval)
  - Homomorphic functions: Func = \{ nand \}
  - Supports only bounded computations: \text{Depth}(C) < D

**Question**

* Can we use HE to build a FHE scheme supporting arbitrary circuits/functions?*

* The process of building FHE from HE is called “bootstrapping”*
Decryption as a boolean function

- Everything is a sequence of bits
  - Secret key $sk$: $\{0, 1\}^k$
  - Ciphertext $ct$: $\{0, 1\}^l$

- $\text{Dec}(sk, ct): \{0, 1\}$
Decryption as a boolean function

- Everything is a sequence of bits
  - Secret key $sk$: $\{0, 1\}^k$
  - Ciphertext $ct$: $\{0, 1\}^l$

- $Dec(sk, ct): \{0, 1\}$

- Usually we think of $Dec$ as a function
  - described by secret key $sk$
  - mapping ciphertext $ct$ to message bit $Dec(sk, ct): \{0, 1\}$
Decryption as a boolean function

- Everything is a sequence of bits
  - Secret key $sk: \{0, 1\}^k$
  - Ciphertext $ct: \{0, 1\}^l$

- $Dec(sk, ct): \{0, 1\}$

- Usually we think of $Dec$ as a function
  - described by secret key $sk$
  - mapping ciphertext $ct$ to message bit $Dec(sk, ct): \{0, 1\}$

- But we can also think of $Dec$ as a function
  - described by ciphertext $ct$
  - mapping secret key $sk$ to message bit $Dec(sk, ct): \{0, 1\}$
Homomorphic Decryption

- Fix a ciphertext $c$
- Define $f_c : sk \mapsto Dec(sk, c)$
- Assume $\text{Size}(f_c) < S$, $\text{Depth}(f_c) < D$
- Let $b_k[1..k] = \text{Enc}(pk, sk[1..k])$

**Question**

*What is the result of the following computation?*

$$\text{EvalC}(pk, f_c, bk[1..k])$$
Proxy Re-encryption

- Primary key: $(pk, sk)$
- Secondary key: $(pk_1, sk_1)$
- Re-encryption key: $rk = Enc(pk_1, sk[1..k])$
- Input ciphertext $c = Enc(pk, m)$
- Decryption function $f_c(sk) = Dec(sk, c)$

**Question**

What is the result of the following computation?

$EvalC(pk_1, f_c, rk)$
Decrypt and compute (unary)

- Homomorphic Encryption \((\text{Gen, Enc, Dec, Eval})\)
- Assume \(\text{Func} = \{ f_c \mid c : \text{CipherText} \}\) where
  \[ f_c(sk) = \text{not} \ (\text{Dec}(sk, c)) \]
Homomorphic Encryption (Gen, Enc, Dec, Eval)

Assume Func = \{ f_c | c: CipherText \} where
\[
f_c(sk) = \neg (\text{Dec}(sk, c))
\]

\[(pk, sk) \leftarrow \text{Gen}()
\]
\[
ek = \text{Enc}(pk, sk)
\]
\[
c = \text{Enc}(pk, m)
\]

**Question**

What is the result of the following computation?

\[
\text{EvalC}(pk, f_c, ek)
\]
Decrypt and compute (binary)

- Homomorphic Encryption ($Gen, Enc, Dec, Eval$)
- Assume $Func = \{ f_{c,c'} \mid c,c' \text{: CipherText} \}$ where $f_{c,c'}(sk) = Dec(sk,c) \text{ nand } Dec(sk,c')$
Homomorphic Encryption \((\text{Gen}, \text{Enc}, \text{Dec}, \text{Eval})\)

Assume \(\text{Func} = \{ f_{c,c'} \mid c, c' : \text{CipherText} \}\) where

\[
f_{c,c'}(sk) = \text{Dec}(sk, c) \text{nand} \text{Dec}(sk, c')
\]

\[
\begin{align*}
\text{(pk, sk)} & \leftarrow \text{Gen}() \\
\text{ek} & \leftarrow \text{Enc}(pk, sk) \\
\text{c} & \leftarrow \text{Enc}(pk, m) \\
\text{c'} & \leftarrow \text{Enc}(pk, m')
\end{align*}
\]

**Question**

What is the result of the following computation?

\[
\text{EvalC}(pk, f_{c,c'}, ek)
\]
Bootstrapping

- Given (1-hop) \((\text{Gen, Enc, Dec, Eval})\) supporting functions

\[ f_{c,c'}(sk) = \text{Dec}(sk,c) \text{ nand } \text{Dec}(sk,c') \]
Bootstrapping

- Given (1-hop) $(\text{Gen}, \text{Enc}, \text{Dec}, \text{Eval})$ supporting functions
  \[ f_{c,c'}(sk) = \text{Dec}(sk,c) \textbf{nand} \text{Dec}(sk,c') \]

- Define (multi-hop) FHE scheme with $\text{Func} = \{ \text{\textbf{nand}} \}$

  \[ \text{Gen}'() = (sk,pk) \leftarrow \text{Gen}() \]
  \[ \text{ek} \leftarrow \text{Enc}(pk,sk) \]
  \[ \text{return } (sk,(pk,ek)) \]

  \[ \text{Enc}'((pk,ek),m) = \text{Enc}(pk,m) \]

  \[ \text{Eval}'((pk,ek),\textbf{nand},c,c') \]
  \[ = \text{EvalC}(pk,f_{c,c'},ek) \]
Correctness

Let \((\text{Gen}', \text{Enc}', \text{Dec}, \text{Eval}')\) be the new encryption scheme.

\textbf{Theorem}

\textit{If} \(\text{Dec}(sk, c) = m\) \textit{and} \(\text{Dec}(sk, c') = m'\), \textit{then}

\[
\text{Dec}(sk, \text{Eval}'((pk, ek), \text{nand}, c, c')) = m \text{ nand } m'
\]
Correctness

Let \((\text{Gen}',\text{Enc}',\text{Dec},\text{Eval}')\) be the new encryption scheme.

**Theorem**

If \(\text{Dec}(sk,c) = m\) and \(\text{Dec}(sk,c') = m'\), then

\[
\text{Dec}(sk,\text{Eval}'((pk,ek),\text{nand},c,c')) = m \text{ nand } m'
\]

**Strong correctness property:**

\[
\text{Dec}(sk,\text{Eval}'((pk,ek),\text{nand},c,c')) = \text{Dec}(sk,c) \text{ nand } \text{Dec}(sk,c')
\]

for **any** ciphertexts \(c,c'\)!
Security

- Assume FHE = (Gen, Enc, Dec, Eval) is IND-CPA secure
- Build new scheme FHE':

  \[
  \text{Gen}'() = (sk, pk) \leftarrow \text{Gen}()
  \]
  \[
  ek \leftarrow \text{Enc}(pk, sk)
  \]
  \[
  \text{return } (sk, (pk, ek))
  \]

  \[
  \text{Enc}'((pk, ek), m) = \text{Enc}(pk, m)
  \]

Is FHE' IND-CPA secure?
Leveled Homomorphic Encryption

- Goal: build a FHE supporting NAND circuits of depth up to \( L \), for any given \( L \)
- Key generation procedure takes \( L \) as input:
Goal: build a FHE supporting NAND circuits of depth up to L, for any given L

Key generation procedure takes L as input:

\[
\text{Gen}'(L) = \\
\quad \text{for } (i = 0..L) \\
\quad \quad (sk[i], pk[i]) \leftarrow \text{Gen}() \\
\quad \text{for } (i = 1..L) \\
\quad \quad ek[i] = \text{Enc}(pk[i], sk[i-1]) \\
\quad sk' = sk[0..L] \\
\quad pk' = pk[0..L], ek[1..L] \\
\quad \text{return } (sk', pk')
\]

\[
\text{Enc}'(pk', m) = \text{Enc}(pk[0], m)
\]
FHE Today

State of the art
We can build leveled FHE from standard LWE assumption
- Built using bootstrapping
- Inefficient, but better than nothing

Open problem
Build (non-leveled) FHE from standard LWE
- In practice, one can apply bootstrapping with
  $ek = \text{Enc}(pk, sk)$
- Much smaller key than leveled FHE
- No known attacks to circular security
- Still, it is not known how to prove security
Section 4

LWE
Linear equations

- \( q \): integer modulus
- \( \mathbb{Z}_q \): integers modulo \( q \)
- \( A \in \mathbb{Z}_q^{n \times m} \): matrix
- \( b \in \mathbb{Z}_q^n \)

**Problem**

Given \( A, b \), find \( x \in \mathbb{Z}^m \) such that \( Ax = b \) (mod \( q \))

**Problem**

Given \( A, b \), find \( x \in \{0, 1\}^m \) such that \( Ax = b \) (mod \( q \))

**Question**

Which problem can be efficiently solved?
Worst-case vs Average-case hardness

**Problem**

Given $A, b$, find $x \in \{0, 1\}^m$ such that $Ax = b \pmod{q}$

- NP-hard: no polynomial time algorithm unless P=NP
- Is it hard to solve on the average?
- For what probability distribution?
  - $A \leftarrow \mathbb{Z}_q^{n \times m}$
  - $x \leftarrow \{0, 1\}^m$
  - $b = Ax \pmod{q}$
- Is $f : (A, x) \mapsto (A, Ax \pmod{q})$ is a One-Way Function?
- For what values of $n, m, q$?
One-Way Functions

**Definition**

$f : D \rightarrow R$ is a one-way function if for any PPT algorithm $I$

Pr\{InvertGame(I)\} $\approx 0$ where

**InvertGame:**

\[
\begin{align*}
x & \leftarrow D \\
y & = f(x) \\
x' & \leftarrow I(y) \\
\text{return } (f(x') \overset{?}{=} y)
\end{align*}
\]

- $D = \mathbb{Z}_q^{n \times m} \times \{0, 1\}^m$
- $R = \mathbb{Z}_q^n$
- $f(A, x) = Ax \mod q$
- Asymptotics: $q(m) = 2^{\text{poly}(m)}$, $n(m) = \text{poly}(m)$
One-Way?

A \leftarrow \mathbb{Z}^{n \times m}
\times \leftarrow \{0, 1\}^m
f(A, x) = Ax \mod q

Question

Is \( f \) a one-way function when

1. \( q = 2^m, n = m \)
2. \( q = 2^m, n = m/2 \)
3. \( q = m, n = m/2 \)
4. \( q = m, n = \sqrt{m} \)
Short Integer Solution (SIS) problem

- Parameters:
  - modulus \( q \)
  - dimensions \( n < m \)
  - bound \( \beta \)

**Problem**

\textit{SIS}: Given \( A \in \mathbb{Z}_q^{n \times m} \) and \( b \in \mathbb{Z}_q^n \), find \( x \in \mathbb{Z}^m \) such that
\[
Ax = b \pmod{q} \text{ and } \|x\| \leq \beta
\]

- More generally: \( x \in S \subset \mathbb{Z}^m \)
- Special cases:
  - \( S = \{x : \|x\| \leq \beta\} \)
  - \( S = \{0, 1\}^m \)
  - \( S = \{x : \|x\|_\infty \leq \beta\} \)
Systematic Form

- Assume $n < m$ (e.g., $n = m/2$)
- Let $A = [I, A'] \in \mathbb{Z}^{n \times m}$ for some $A' \in \mathbb{Z}^{n \times (m-n)}$

**Lemma**

*If SIS is hard, then SIS’ is hard*
Learning With Errors

- SIS': $A = [I, A'] \in \mathbb{Z}^{n \times m}$ where $n < m$ (say, $n = m/2$)
- Let $x = (e, s)$
- $A \cdot x = [I, A'](e, s) = A's + e$

**Problem**

$LWE$: Given $A'$ and $b$, find small $e, s$ such that $A's + e = b$

**Problem**

$LWE$: Given $A'$ and $b$, find small $e, s$ such that $A's \approx b$

Notice:

- $A' \in \mathbb{Z}_q^{n \times n}$
- $A's = b$ is easy to solve
- $A's \approx b$ seems hard
LWE problem

Notation:

- secret $s \leftarrow \mathbb{Z}_q^n$, usually chosen at random
- modulus $q(n) = poly(n)$
- $A \leftarrow \mathbb{Z}_q^{m \times n}$
- error $e \leftarrow \chi^m$, usually $|e_i| \approx \sqrt{n}$
- $b = As + e \in \mathbb{Z}_q^m$

Problem

Search LWE: Given $A$ and $b$, find $s$

- Each row of $A$ gives an approximate equation $\langle a, s \rangle \approx b$
- if $m \gg n$, then $s$ is uniquely determined
- Still, hard to find $s$
Uniform vs Small secrets

**Lemma**

If LWE is hard for \( s \leftarrow \chi^n \), then it is hard for \( s \leftarrow \mathbb{Z}_q^n \).
Lemma

If LWE is hard for $s \leftarrow \chi^n$, then it is hard for $s \leftarrow \mathbb{Z}_q^n$

Proof: Assume $Adv$ solves LWE with uniform $s \leftarrow \mathbb{Z}_q^n$

$$Adv'(A, b)$$

$$s \leftarrow \mathbb{Z}_q^n$$

$$b' = b + As$$

$$s' = Adv(A, b')$$

Return $(s' - s)$
Decisional LWE (DLWE)

**Definition**

LWE distribution:

\[
\text{LWE} \left[q, n, m \right] = \begin{align*}
\text{do} & \ A \leftarrow \mathbb{Z}_{q}^{m \times n} \\
\text{do} & \ s \leftarrow \mathbb{Z}_{q}^{n} \\
\text{do} & \ e \leftarrow \chi^{m} \\
b = As + e
\end{align*}
\]

\[\text{return} \ (A, b)\]

**Definition**

Decisional LWE (DLWE): distinguish \( \text{LWE}[q,n,m] \) from \( \text{Uniform}(\mathbb{Z}_{q}^{m \times (n+1)}) \)
Decision to Search reduction

**Theorem**

*If DLWE is hard, then LWE is hard*
Theorem

If DLWE is hard, then LWE is hard

Proof:

- Assume Adv solves LWE
- Given Adv' that solves DLWE

Adv'(A,b):

\[ s \leftarrow \text{Adv}(A,b) \]

\[ \text{if } (As \approx b) \]
\[ \text{then return "LWE"} \]
\[ \text{else return "random"} \]
Search vs Decision

- Is (Search) LWE harder than DLWE?

**Theorem**

*If Search LWE is hard for any $m = \text{poly}(n)$, then DLWE is also hard for any $m = \text{poly}(n)$*

**Theorem**

*For any $m = \text{poly}(n)$, if Search LWE is hard, then DLWE is also hard for any $m = \text{poly}(n)$*
LWE Search to Decision reduction (easy version)

- Assume $Adv$ can distinguish LWE from uniform
- Task: Given $A, b$, find $s$ such that $As \approx b \pmod{q}$
- Assumption: $s$ is unique (holds with very high probability)
- We show how to check if $s_i = \gamma$:

\[
Adv(A, b) :
\begin{align*}
a & \leftarrow \mathbb{Z}_m \\
A' &= A + [0..0, a, 0..0] \\
b' &= b + \gamma a \\
\text{case} \quad Adv(A', b') \quad \text{of} \\
\quad "LWE" : & \quad \text{return} \quad s_i = c \\
\quad "random" : & \quad \text{return} \quad s_i \neq c
\end{align*}
\]

- Recover all entries of $s$, one at a time
In the rest of the course we will just assume that DLWE is hard.

There are several variants of the assumption:

- Uniform vs. small secret $s$
- Different (always small) error distributions $e \leftarrow \chi$
- Fixed vs unbounded number of samples $m$
- Different values of $q$
- Concrete hardness assumptions

By and large all variants are equivalent up to polynomial reductions.
How to Encrypt with LWE

- Fix secret $s$ in $\mathbb{Z}_q^n$
- LWE samples $(a_i, b_i)$ where $a_i \in \mathbb{Z}_q^n$ and $b_i \in \mathbb{Z}$
- Polynomially many samples $(a_i, b_i)$ for $i = 1, 2, ...$
- DLWE: the $b_i$ values are pseudorandom
- Idea: use $b_i$ as a one-time pad to encrypt a message $m$
LWE Symmetric Encryption

\[
\text{Gen}(): \\
\quad s \leftarrow \mathbb{Z}_q^n \\
\quad \text{return } s
\]

\[
\text{Enc}(s, m): \\
\quad a \leftarrow \mathbb{Z}_q^n \\
\quad e \leftarrow \chi \\
\quad b = \langle a, s \rangle + e + m
\]

\[
\text{Dec}(s, (a, b)): \\
\quad \text{return } (b - \langle a, s \rangle)
\]

Is this a valid encryption scheme?
Symmetric Encryption

\[
\text{SKE(\text{Gen, Enc, Dec})}
\]

\[
\begin{align*}
\text{Gen: } & \quad () \rightarrow \text{sk} \\
\text{Enc: } & \quad (\text{sk}, m) \rightarrow c \\
\text{Dec: } & \quad (\text{sk}, c) \rightarrow m
\end{align*}
\]

Correctness: for every \( \text{sk} \leftarrow \text{Gen()} \) and \( m \leftarrow [M], r \leftarrow [R] \):

\[
\text{Dec(\text{sk}, Enc(\text{sk}, m; r))} = m
\]

Question

*Is this a valid encryption scheme?*
Correcting from errors

- Ciphertext modulus $q$
- Message modulus $p$ (assume $p$ divides $q$)
- Message space: $m \in \mathbb{Z}_p$

$\text{Enc}(s, m) = (a, b)$ where

$$a \leftarrow \mathbb{Z}_q^n, \quad e \leftarrow \chi$$

$$b = \langle a, s \rangle + e + \left(\frac{q}{p}\right)m$$

$\text{Dec}(s, (a, b)) = \text{round}\left(\frac{c \cdot p}{q}\right)$ where

$$c = b - \langle a, s \rangle \mod q$$

**Lemma**

If $|e| < \beta$ then $\text{Dec}(s, \text{Enc}(s, m; a, e)) = m$

**Question**

For what value of $\beta$ is the lemma correct?
IND-CPA security for symmetric encryption

\[
\text{INDCPA}_{\text{SKE}}(b : \{0, 1\})
\]

\[
\begin{align*}
\text{sk} & \leftarrow \text{Gen}() \\
b' & \leftarrow A[\text{LR}] \\
\text{return } b' : \{0, 1\}
\end{align*}
\]

\[
\text{LR}(m_0, m_1):
\begin{align*}
\text{ct} & \leftarrow \text{Enc}(sk, m_b) \\
\text{return } ct
\end{align*}
\]

- Similar LR security definition can be given also for PKE: \(A[\text{LR}](\text{pk})\) is given \(\text{pk}\) and oracle access to LR
- Previous PKE IND-CPA game allows only one query to LR

**Question**

*Why can restrict PKE INDCPA game to one query?*
Security of LWE symmetric encryption

- Assume $|e| < \beta = q/(2p)$ for all $e \leftarrow \chi$
- Is LWE INDCPA game SKE secure?

**Theorem**

Assume DLWE holds for a given $q(n)$ and any $m = \text{poly}(n)$. Then LWE symmetric encryption is INDCPA secure, i.e., any adversary $\text{Adv}$ has negligible advantage in the INDCPA game SKE distinguishing game.
RR-CPA security

- LWE encryption satisfies a stronger security property: ciphertext indistinguishability from random

\[ \text{INDCPA}_\text{gameSKE}(b:\{0,1\}) \]

\[ \begin{align*}
    sk & \leftarrow \text{Gen}() \\
    b' & \leftarrow A[\text{RR}] \\
    \text{return } b' & :\{0,1\}
\end{align*} \]

\[ \text{RR}(m) : \]

\[ \begin{align*}
    ct_0 & \leftarrow \text{Enc}(sk,m) \\
    ct_1 & \leftarrow \mathbb{Z}_q^{n+1} \\
    \text{return } ct_b
\end{align*} \]

- “Real or Random” oracle RR
- RR-CPA security also provides a form of anonymity
LeftRight vs RealRand security

**Theorem**

*If a (SKE or PKE) scheme is INDCPA-RR secure, then it is also INDCPA-LR secure.*

**Remark**

*A (SKE or PKE) scheme can be INDCPA-LR secure, but not INDCPA-RR secure.*
Compact LWE Encryption

- Ciphertext expansion: \( \text{bitsize}(ct) / \text{bitsize}(m) \)
- Compact LWE SKE (\( \text{Gen}, \text{Enc}, \text{Dec} \))

\( \text{Gen}() : \)

\[
S \leftarrow \mathbb{Z}_q^{l \times n} \\
\text{return } S
\]

\( \text{Enc}(S, m) = (a, b) \)

\[
a \leftarrow \mathbb{Z}_q^n \\
e \leftarrow \chi^l \\
b = Sa + e + \text{round}((p/q)m)
\]

\( \text{Dec}(S, (a, b)) : \)

\[
c \leftarrow b - S^t a \mod q \\
\text{return } \text{round}(c \times p/q)
\]

**Theorem**

*Compact LWE SKE is correct and IND-CPA-RR secure*
Ciphertext Expansion

Compact LWE encryption:

- Key $S \in \mathbb{Z}_q^{l \times n}$
- Message $m \in \mathbb{Z}_p^l$
- Encryption $\text{Enc}(S, m) = (a, b)$ where $b = Sa + e + mp/q$
- Ciphertext $(a, b) \in \mathbb{Z}_q^{n+l}$

Question

What is the ciphertext/plaintext size ratio?

Example:

- $\text{Enc}(f, x; r) = (f(r), H(r) \oplus m)$ where $f : \{0, 1\}^k \rightarrow \{0, 1\}^k$
- $\text{Enc}(f, .) : \{0, 1\}^m \rightarrow \{0, 1\}^{m+k}$
- Ciphertext expansion: $(m + k)/m = 1 + (k/m)$
LWE Symmetric Encryption

\begin{align*}
\textbf{Gen}() : \\
& s \leftarrow \mathbb{Z}_q^n \\
& \text{return } s
\end{align*}

\begin{align*}
\textbf{Enc}(s,m) : \\
& a \leftarrow \mathbb{Z}_q^n \\
& e \leftarrow \chi \\
& b = \langle a, s \rangle + e + (q/p)m \\
& \text{return } (a, b)
\end{align*}

\begin{align*}
\textbf{Dec}(s,(a,b)) : \\
& d = b - \langle a, s \rangle \mod q \\
& \text{return } \text{round}(d\times p/q))
\end{align*}
Compact (Matrix) LWE

**Gen()**:  
\[ S \leftarrow \mathbb{Z}_q^{l \times n} \]  
\text{return } S

**Enc(S,M) = (A,B)**  
\[ A \leftarrow \mathbb{Z}_q^{n \times w} \]  
\[ E \leftarrow \chi_{l \times w} \]  
\[ B = SA + E + \text{round}((p/q)M) \mod q \]

**Dec(S,(A,B))**:  
\[ D \leftarrow B - SA \mod q \]  
**return** \( \text{round}(D \times p/q) \)

**Notation:**
- \([A, B] \): horizontal concatenation
- \((A, B) \): vertical concatenation
Linearity of the LWE function

- Let \( \text{LWE}(S, X; A, E) = SA + X + E \) be the raw LWE function.
- Encryption: \( \text{Enc}(S, M) = \text{LWE}(S, (q/p)M; A, E) \) for random \( A, E \).
- Linear properties:
  \[
  \text{LWE}(S, X; A, E) + \text{LWE}(S, X'; A', E') = \text{LWE}(S, X+X'; A+A', E+E')
  \]
  \[
  \text{LWE}(S, X; A, E) - \text{LWE}(S, X'; A', E') = \text{LWE}(S, X-X'; A-A', E-E')
  \]
  \[
  c * \text{LWE}(S, X; A, E) = \text{LWE}(S, cX; cA, cE)
  \]
Linearity of the LWE function

- Let $LWE(S, X; A, E) = SA + X + E$ be the raw LWE function
- Encryption: $Enc(S, M) = LWE(S, (q/p)M; A, E)$ for random $A, E$
- Linear properties:
  
  $LWE(S, X; A, E) + LWE(S, X'; A', E') = LWE(S, X+X'; A+A', E+E')$

  $LWE(S, X; A, E) - LWE(S, X'; A', E') = LWE(S, X-X'; A-A', E-E')$

  $c \cdot LWE(S, X; A, E) = LWE(S, c \cdot X; c \cdot A, c \cdot E)$

- Key Homomorphism:
  
  $LWE(S, X; A, E) + LWE(S', X'; A, E') = LWE(S+S', X+X'; A, E+E')$

- Ciphertexts must use the same $A$!
Ciphertexts that “encrypt” $X$ under $S$ with error $E$.

**Definition**

$LWE(S,X;E) = \{ (A,B) : B = LWE(S,X;A,E) \}$

$LWE(S,X;\beta) = \{ (A,B) : B = LWE(S,X;A,E), |E|_\infty < \beta \}$
Linearity of Ciphertexts

Ciphertexts that “encrypt” \( X \) under \( S \) with error \( E \).

Definition

\[
\begin{align*}
\text{LWE}(S, X; E) &= \{ (A, B) : B = \text{LWE}(S, X; A, E) \} \\
\text{LWE}(S, X; \beta) &= \{ (A, B) : B = \text{LWE}(S, X; A, E), |E|_\infty < \beta \}
\end{align*}
\]

- \( \text{LWE}(S, X; E) + \text{LWE}(S, X'; E') \subseteq \text{LWE}(S, X+X'; E+E') \)
- \( \text{LWE}(S, X; E) - \text{LWE}(S, X'; E') \subseteq \text{LWE}(S, X-X'; E-E') \)
- \( c\times \text{LWE}(S, X; E) \subseteq \text{LWE}(S, c\times X; c\times E) \)
Linearity of Ciphertexts

Ciphertexts that “encrypt” $X$ under $S$ with error $E$.

**Definition**

$LWE(S, X; E) = \{ (A, B) : B = LWE(S, X; A, E) \}$

$LWE(S, X; \beta) = \{ (A, B) : B = LWE(S, X; A, E), |E|_\infty < \beta \}$

- $LWE(S, X; E) + LWE(S, X'; E') \subseteq LWE(S, X+X'; E+E')$
- $LWE(S, X; E) - LWE(S, X'; E') \subseteq LWE(S, X-X'; E-E')$
- $c \cdot LWE(S, X; E) \subseteq LWE(S, c \cdot X; c \cdot E)$

**Question**

$LWE(S, X; \beta) + LWE(S, X'; \beta') \subseteq LWE(S, X+X'; \beta + \beta')$ ?

**Question**

$LWE(S, X; \beta) - LWE(S, X'; \beta') \subseteq LWE(S, X+X'; \beta - \beta')$ ?
Message and Ciphertext Operations

- **Addition:**
  - \( M_0 + M_1 \in \mathbb{Z}_q^{l \times w} \)
  - \( (A_0, B_0) + (A_1, B_1) = (A_0 + A_1, B_0 + B_1) \in \mathbb{Z}_q^{(n+l) \times w} \)

- **Subtraction**
  - \( M_0 - M_1 \in \mathbb{Z}_q^{l \times w} \)
  - \( (A_0, B_0) - (A_1, B_1) = (A_0 - A_1, B_0 - B_1) \in \mathbb{Z}_q^{(n+l) \times w} \)

- **Scalar multiplication**
  - \( c \cdot M \in \mathbb{Z}_q^{l \times w} \)
  - \( c \cdot (A, B) = (c \cdot A, c \cdot B) \in \mathbb{Z}_q^{(n+l) \times w} \)

- Arbitrary linear transformations . . .
Additive Homomorphism Encryption

Homomorphic Encryption supporting the *addition* of ciphertexts

\[ \begin{aligned} &\text{sk} \leftarrow \text{Gen}() \\
&c_0 \leftarrow \text{Enc}(\text{sk}, m_0) \\
&c_1 \leftarrow \text{Enc}(\text{sk}, m_1) \\
&c = c_0 + c_1 \\
&m = m_0 + m_1 \\
&D\text{ec}(\text{sk}, c) \overset{?}{=} m \end{aligned} \]

**Question**

*Does LWE encryption satisfy the additive homomorphic property? For what error bound $|\chi| < \beta$?*

**Question**

*Is ciphertext $c$ distributed according to $\text{Enc}(m_0 + m_1)$?*
Summation

- Homomorphic Encryption supporting the *addition* of ciphertexts

\[
\begin{align*}
    sk & \leftarrow \text{Gen}() \\
    c_1 & \leftarrow \text{Enc}(sk, m_1) \\
    c_2 & \leftarrow \text{Enc}(sk, m_2) \\
    \cdots \\
    c_k & \leftarrow \text{Enc}(sk, m_k) \\
    c & = c_1 + c_2 + \cdots + c_k \\
    m & = m_1 + m_2 + \cdots + m_k \\
    \text{Dec}(sk, c) & \equiv m
\end{align*}
\]

Question

*For any given bound \(|\chi| < \beta\), what is the largest value of \(k\) for which one can add \(k\) ciphertexts?*
Subtraction and Scalar multiplication

- Subtraction $m_0 - m_1$: similar to addition $m_0 + m_1$
- $\pm 1$-linear combinations: similar to summation
- Scalar multiplication $c \cdot m$: error grows by a factor $c$
- Ciphertexts can be multiplied only by small scalars!
Concatenation

- \( \text{LWE}(S, X; A, E) = SA + X + E \)
  - \( S \in \mathbb{Z}_{q}^{k \times n} \)
  - \( A \in \mathbb{Z}_{q}^{n \times w} \)
  - \( X, E \in \mathbb{Z}_{q}^{k \times w} \)

- The same \( S \) can be used with messages \( X \) with any number of columns \( w \)

- Message Concatenation \( X \mid X' = [X, X'] \)

**Definition**

\( (A, B) \mid (A', B') = ([A, A'], [B, B']) \)

**Theorem**

\( \text{LWE}(S, X; A, E) \mid \text{LWE}(S, X'; A', E) \subseteq \text{LWE}(S, [X, X']; [A, A'], [E, E']) \)
Linear Transforms

- Left multiplication by a constant matrix: \( M \rightarrow M^T \)
- Ciphertext \( C = \text{LWE}(S, M; E) \)
- Notice: \( M \) and \( C \) have the same number of columns
- We can apply \( T \) to \( C \): \( C \rightarrow CT \)

**Theorem**

\[
\text{LWE}(S, X; A, E)^*T \subseteq \text{LWE}(S, XT; AT, ET)
\]

\[
\text{LWE}(S, X; E)^*T \subseteq \text{LWE}(S, XT; ET)
\]

**Special case:**

- Addition: \( C + C' = [C|C']^T \) for \( T=(I,I) \)
- Subtraction: \( C - C' = [C|C']^T \) for \( T=(I,-I) \)
Constant Messages

Question

Can you compute an LWE encryption of a message $M$ without knowing the secret key $S$?

- I pick $S \leftarrow \text{Gen}()$ and keep it secret
- Goal: find ciphertext $C$ such that $\text{Dec}(S, C) = M$
**Question**

*Can you compute an LWE encryption of a message $M$ without knowing the secret key $S$?*

- I pick $S \leftarrow \text{Gen}()$ and keep it secret.
- Goal: find ciphertext $C$ such that $\text{Dec}(S, C) = M$.
- Let $(A, B) = (0, (q/p)M)$.
- We write $\text{Const}(M)$ for the constant ciphertext $(0, (q/p)M)$.

**Remarks:**

- The ciphertext $C$ is independent of $S$.
- $C = \text{LWE}((q/p)M; 0)$ is a “noiseless” encryption of $M$. 
Constant Messages as Homomorphic properties

- \( \text{LWE}(S, M; E) + \text{LWE}((q/p)M'; 0) = \text{LWE}(S, M+M'; E) \)

- **Homomorphism for “nullary functions”** \( f_M() = M \)
  - Given an empty sequence of ciphertexts [], produce an encryption of \( f_M([]) = M \)

- **Homomorphism for unary functions** \( f_M(M') = M + M' \)
  - Given an encryption of \( M' \), produce an encryption of the shifted message \( M + M' \)
Circular security

- A PKE scheme is “circular secure” if one can securely publish the encryption $\text{Enc}(pk, sk)$.
- A SKE scheme is “circular secure” if one can securely publish the encryption $\text{Enc}(sk, sk)$.

**Definition**

A PKE scheme $(\text{Gen}, \text{Enc}, \text{Dec})$ is circular secure if $(\text{Gen}', \text{Enc}', \text{Dec})$ is IND-CPA secure where

$$
\begin{align*}
\text{Gen}'() : \\
(sk, pk) &\leftarrow \text{Gen}() \\
ct &\leftarrow \text{Enc}(pk, sk) \\
\text{pk}' &= (pk, ct) \\
\text{Enc}'((pk, ct), msg) &= \text{Enc}(pk, msg)
\end{align*}
$$
Can we transform Secret Key Encryption to Public Key Encryption?

- Not in general: black box separations
- Impagliazzo’s worlds: Minicrypt vs Cryptomania
Application: Public key encryption

- Can we transform Secret Key Encryption to Public Key Encryption?
  - Not in general: black box separations
  - Impagliazzo’s worlds: Minicrypt vs Cryptomania

- What if we start from an Additively Homomorphic SKE scheme?
  - Black box separation results break down

- What about a weakly (bounded) additive scheme?

- What about our LWE SKE scheme?
PKE: Construction

- Start from SKE \((\text{Gen, Enc, Dec})\)
- Construct a PKE \((\text{Gen}', \text{Enc}', \text{Dec})\)

\textbf{Gen'}():

\begin{verbatim}
sk ← Gen()
for i=1..n
    pk[i] ← Enc(sk, 0)
pk = pk[1..n]
return (sk, pk)
\end{verbatim}

\textbf{Enc'}(pk, msg):

\begin{verbatim}
for i=1..n
    r[i] ← \{0,1\}
ct = Const(msg) + \text{sum} \{ pk[i] : r[i] = 1 \}
return ct
\end{verbatim}
Correctness of PKE

\[ \text{Dec} (sk, \text{msg} + \text{Enc}(sk,0) + \ldots + \text{Enc}(sk,0)) = \text{msg} + 0 + \ldots + 0 = \text{msg} \]

**Theorem**

*If SKE is (1-hop) homomorphic under constant increment and n-summation, then PKE is correct.*

**Theorem**

*If SKE is (1-hop) homomorphic under constant increment and hn-summation, then PKE is correct and homomorphic under constant increment and n-summation.*
Correctness of PKE

\[
\text{Dec}(sk, \text{msg} + \text{Enc}(sk,0) + \ldots + \text{Enc}(sk,0)) \\
= \text{msg} + 0 + \ldots + 0 = \text{msg}
\]

**Theorem**

If SKE is (1-hop) homomorphic under constant increment and \(n\)-summation, then PKE is correct.

**Theorem**

If SKE is (1-hop) homomorphic under constant increment and \(hn\)-summation, then PKE is correct and homomorphic under constant increment and \(n\)-summation.

**Question**

Assume SKE is an IND-CPA secure and homomorphic. Is PKE secure?
For what value of $n$?

Certainly not secure for $n = 1$ (or even $n = 0$!)

What about large $n$?

How large?

Answer: Secure, for large enough $n$ and any additively homomorphic SKE [Rothblum, TCC 2011]
The case of LWE SKE

- Consider the PKE scheme obtained from our LWE-based SKE

\[\text{Gen}'() : \]
\[S \leftarrow \text{Gen}() \]
\[P = \text{Enc}(S,0) | .. | \text{Enc}(S,0) = \text{Enc}(S,[0..0]) \]
\[\text{return} \ (S,P) \]

\[\text{Enc}'(P,M) : \]
\[R \leftarrow \{0,1\}^{*} \]
\[PR + \text{Const}(M) \]

**Theorem**

*LWE PKE is RR-IND secure.*
Universal Hashing

Definition

A function family $H = \{ h : X \rightarrow Y | h \}$ is 2-universal if for any $a, b \in X$,

$$\{(h(a), h(b)) | h \in H\} \equiv \{(f(a), f(b)) | f : X \rightarrow Y\}$$

- Let $(X, +)$ be an additive group
- For any vector $a \in X^n$, define the subset-sum function $h(a, S) = \sum\{a_i : i \in S\}$

Question

Which of the following function families is 2-universal?

1. $\{h_a : S \rightarrow h(a, S) | a \in X^n\}$
2. $\{h_S : a \rightarrow h(a, S) | S \subseteq \{1, \ldots, n\}\}$
3. Both
4. Neither
Universal Hashing (continued)

- \( h_a(S) = \sum_{i \in S} a_i \) is not 2-universal

- What about \( g_{a,b}(S) = b + h_a(S) \)?
  - Yes, this is 2-universal
  - Prove it as an exercise

- \( \{ h_a : \{0,1\}^n \rightarrow X \} \) still satisfies a weaker property which is enough for our purposes

**Definition**

For any \( a \neq b \), \( \Pr_h\{ h(a) = h(b) \} = 1/|X| \)

- We will refer to this weaker property as 2-universal'
Universal Hashing: proof

Lemma

For any group \((X, +)\), the function family \(\{h_a(S) = \sum_{i \in S} a_i\}_a\) is 2-universal’, i.e., for all \(S \neq T\) we have
\[
\Pr_h\{h(S) = h(T)\} = \frac{1}{|X|}
\]

Proof.

- Let \(j \in S \setminus T\)
- Fix \(a_i\) for all \(i \neq j\)
- Let \(T' = T \setminus S\) and \(S' = S \setminus (T \cup \{i\})\)
- \(c = \sum_{i \in T'} a_i - \sum_{i \in S'} a_i\) does not depend on \(a_j\)
- \(h_a(S) = h_a(T)\) iff \(a_j = c\)
- \(\Pr\{a_j = c\} = 1/|X|\)
Leftover Hash Lemma

**Lemma**

For any 2-universal’ family \(\{h : X \rightarrow Y | h \in H\}\), the distributions

\[
\{(h, h(x)) | h \leftarrow H, x \leftarrow X\}
\]

\[
\{(h, y) | h \leftarrow H, y \leftarrow Y\}
\]

are within statistical distance \(\Delta \leq \sqrt{|Y|/|X|}\).

**Proof Steps:**

1. If \(H\) is 2-universal’, then \((H, H(X))\) has small collision probability
2. If \((H, H(X))\) has small collision probability, then it is statistically close to uniform
Collision Probability and Uniformity

- $Z, Z'$ i.i.d., with $\Pr\{Z = z\} = p(z)$

**Definition**

Collision Probability:

$C(Z) = \Pr\{Z = Z'\} = \sum_z p(z)^2$

- $\sum_z (p(z) - 1/|Z|)^2 = C(Z) - 1/|Z|$
- **Norm inequality:** $\forall \mathbf{v} \in \mathbb{R}^n. \|\mathbf{v}\|_1 \leq \sqrt{n} \|\mathbf{v}\|_2$
- $\Delta(Z, U) = \frac{1}{2} \sum_z |p(z) - 1/|Z||$
Collision Probability and Uniformity

- $Z, Z'$ i.i.d., with $\Pr\{Z = z\} = p(z)$

**Definition**

Collision Probability:
$$C(Z) = \Pr\{Z = Z'\} = \sum_z p(z)^2$$

- $\sum_z (p(z) - 1/|Z|)^2 = C(Z) - 1/|Z|$
- Norm inequality: $\forall v \in \mathbb{R}^n. \|v\|_1 \leq \sqrt{n} \|v\|_2$
- $\Delta(Z, U) = \frac{1}{2} \sum_z |p(z) - 1/|Z||$
- $\Delta \leq \frac{1}{2} \sqrt{|Z|} \sqrt{\sum_z (p(z) - 1/|Z|)^2}$
- $\Delta \leq \frac{1}{2} \sqrt{|Z|} C(Z) - 1$
Collision Probability of Universal Hashing

- $Z = (H, H(X))$, 2-universal function family $H : X \rightarrow Y$
- Collision Probability of $Z$:
  \[ C(Z) = \Pr(h = h', h(x) = h'(x')| h, h' \leftarrow H, x, x' \leftarrow X) \]
- $C = \frac{1}{|H|} \Pr_{x,x'}[\Pr_h(h(x) = h(x'))]$
Collision Probability of Universal Hashing

- \( Z = (H, H(X)) \), 2-universal function family \( H : X \rightarrow Y \)
- Collision Probability of \( Z \):
  \[
  C(Z) = \Pr(h = h', h(x) = h'(x') | h, h' \leftarrow H, x, x' \leftarrow X)
  \]
- \( C = \frac{1}{|H|} \Pr_{x,x'}[\Pr_{h}(h(x) = h(x'))] \)
- Union bound:
  - \( \Pr(x = x') = \frac{1}{|X|} \)
  - If \( x \neq x' \), then \( \Pr_{h}(h(x) = h(x')) \leq \frac{1}{|Y|} \)
Collision Probability of Universal Hashing

- $Z = (H, H(X))$, 2-universal function family $H : X \rightarrow Y$

- Collision Probability of $Z$:
  \[ C(Z) = \Pr(h = h', h(x) = h'(x') | h, h' \leftarrow H, x, x' \leftarrow X) \]

- $C = \frac{1}{|H|} \Pr_{x,x'}[\Pr_h(h(x) = h(x'))]$

- Union bound:
  \[ \Pr(x = x') = \frac{1}{|X|} \]
  \[ \text{If } x \neq x', \text{ then } \Pr_h(h(x) = h(x')) \leq \frac{1}{|Y|} \]

- $C \leq \frac{1}{|H|} \left( \frac{1}{|X|} + \frac{1}{|Y|} \right)$
Collision Probability of Universal Hashing

- \( Z = (H, H(X)) \), 2-universal function family \( H : X \rightarrow Y \)

- Collision Probability of \( Z \):
  \[
  C(Z) = \Pr(h = h', h(x) = h'(x') | h, h' \leftarrow H, x, x' \leftarrow X)
  \]

- \( C = \frac{1}{|H|} \Pr_{x,x'}[\Pr_h(h(x) = h(x'))] \)

- Union bound:
  - \( \Pr(x = x') = 1/|X| \)
  - If \( x \neq x' \), then \( \Pr_h(h(x) = h(x')) \leq 1/|Y| \)

- \( C \leq \frac{1}{|H|} \left( \frac{1}{|X|} + \frac{1}{|Y|} \right) \)

- Using \( |Z| = |H| \cdot |Y| \) we get
  \[
  \Delta \leq \frac{1}{2} \sqrt{|Z|C - 1} = \frac{1}{2} \sqrt{|Y|/|X|}
  \]
Security of LWE PKE

\[ \text{Gen()} : S, E \leftarrow \ldots \]
\[ P = \text{Enc}(S, [0..0]) = (A, SA+E) \]
\[ \text{return } (S, P) \]

\[ \text{Enc}(P, M) : R \leftarrow \{0, 1\}^* \]
\[ \text{return } PR + \text{Const}(M) \]

**Theorem**

*LWE PKE is RR-IND secure.*
Security of LWE PKE

Theorem

LWE PKE is RR-IND secure.

Proof:

1. Assume Adv breaks PKE
2. LWE Assumption: \( P = (A, SA+E) \approx (A, B) \)
3. Adv breaks RR-CPA when \( P \) is uniform
4. If \( P \) is uniform, then \( (P, PR) \) is close to uniform
Details

Claim: \((P, PR)\) is close to uniform

- Enough to look at a single column \((P, Pr)\)
  - Statement for matrix \((P, PR)\) follows by hybrid argument

- \(P: r \rightarrow Pr\) is 2-universal
  - Columns of \(P\) belong to a group \((\mathbb{Z}_q^{n+l}, +)\)
  - \(r\) selects a subset of the columns of \(P\)
  - Apply Leftover Hash Lemma
Homomorphic PKE

- \( \text{Enc}(P, M) = PR + \text{Const}(M) \)

- \( \text{Enc}(P, M) + \text{Enc}(P, M') = PR + \text{Const}(M) + PR' + \text{Const}(M') = P(R + R') + \text{Const}(M + M') \)

- \( \text{Enc}(P, M) + \text{Enc}(P, M') \approx \text{Enc}(P, M + M') \)
  - Noise: \( E + E' \)
Homomorphic PKE

- $\text{Enc}(P,M) = PR + \text{Const}(M)$

- $\text{Enc}(P,M) + \text{Enc}(P,M') = PR + \text{Const}(M) + PR' + \text{Const}(M') = P(R+R') + \text{Const}(M+M')$

- $\text{Enc}(P,M) + \text{Enc}(P,M') \approx \text{Enc}(P,M+M')$
  - Noise: $E+E'$

- $[\text{Enc}(P,M) | \text{Enc}(P,M')] = \text{Enc}(P,[M|M'])$
  - Noise: $[E|E']$

- $\text{Enc}(P,M)T \approx \text{Enc}(P,MT)$
  - Noise: $ET$
  - $T$ must be small
Encoding modulo q

- Ciphertext modulus $q$. Assume $q = 2^k$
- Plaintest modulus $p \ll q$, e.g., $p=2$. Use scaling $\text{Const}(msg) = (0, (q/p)\text{msg})$ to allow error correction and correct decryption
Ciphertext modulus $q$. Assume $q = 2^k$

Plaintest modulus $p \ll q$, e.g., $p=2$. Use scaling

$$\text{Const}(\text{msg}) = (0, (q/p)\text{msg})$$

to allow error correction and correct decryption

What if we want to encrypt $\text{msg} \in \mathbb{Z}_q$?
Encoding modulo q

- Ciphertext modulus \( q \). Assume \( q = 2^k \)
- Plaintest modulus \( p \ll q \), e.g., \( p=2 \). Use scaling \( \text{Const}(msg) = (0, (q/p) \cdot msg) \) to allow error correction and correct decryption

- What if we want to encrypt \( msg \in \mathbb{Z}_q \)?

- Idea:
  - write \( msg = \sum_i m_i 2^i \), where \( m_i \in \{0, 1\} \)
  - Encrypt each bit individually: \( \text{Enc}(m_0), \ldots, \text{Enc}(m_k) \)
Encoding modulo q

- Ciphertext modulus $q$. Assume $q = 2^k$
- Plaintext modulus $p \ll q$, e.g., $p=2$. Use scaling $\text{Const}(msg) = (0, (q/p) \text{msg})$ to allow error correction and correct decryption

$$\text{Enc}(m: \{0,1\}^k) = (a, Sa+e+(q/2)m)$$

$$\text{bitDecomp}(msg: \mathbb{Z}_q) =$$
$$\text{for} \ i=0..k-1$$
$$m[i] = (msg >> i) \mod 2$$
$$\text{return} \ m[]$$

$$\text{Enc'}(msg: \mathbb{Z}_q) =$$
$$\text{return} \ (\text{Enc}(\text{bitDecomp}(msg)))$$
Linear Encoding

- Bit encoding: \((\text{msg} : \mathbb{Z}_q) \rightarrow (m[*] : \{0, 1\}^k)\)
  - good: works for any message space
  - bad: breaks linear homomorphic properties

- We need to use a linear encoding function:
  - \((\text{msg} : \mathbb{Z}_q) \rightarrow (m[*] : \mathbb{Z}_q^k)\)
  - \(\text{msg} \rightarrow \text{msg}^*(1, 2, 4, 8, \ldots)\)
Linear Encoding

- **Bit encoding:** \((\text{msg} : \mathbb{Z}_q) \rightarrow (m[*] : \{0,1\}^k)\)
  - good: works for any message space
  - bad: breaks linear homomorphic properties

- **We need to use a linear encoding function:**
  - \((\text{msg} : \mathbb{Z}_q) \rightarrow (m[*] : \mathbb{Z}_q^k)\)
  - \(\text{msg} \rightarrow \text{msg} \times (1,2,4,8,...)\)

- **Column encoding:**
  - \(\text{pow2col} = (1,2,4,8,...)\)
  - \(\text{Enc}'(S,\text{msg}) = \text{LWE}(S,\text{msg} \times \text{pow2col}) = (a,b)\)

- **Row encoding:**
  - \(\text{pow2row} = [1,2,4,8,...]\)
  - \(\text{Enc}'(s,\text{msg}) = \text{LWE}(s,\text{msg} \times \text{pow2row}) = (A,b)\)
Decoding modulo $q$

**Question**

- *Can you decrypt*
  
  $\text{Enc}'(S, \text{msg}) = \text{LWE}(S, \text{msg} \times \text{pow2col}) = (a, b)$?

- *Can you decrypt*
  
  $\text{Enc}'(s, \text{msg}) = \text{LWE}(s, \text{msg} \times \text{pow2row}) = (A, b)$?

- *For what error bound* $|e|_\infty < \beta$?
Decryption algorithm

\[ \text{Enc}'(s, \text{msg}) = \text{LWE}(s, \text{msg} \times \text{pow2row}) = (A, b) \text{ where} \]
\[ b = sA + e + \text{msg} \times \text{pow2row} \]

\[ \text{Dec}'(s, (A, b)):\]
\[ \text{msg} \leftarrow 0 \]
\[ \text{for } i = 0 \ldots (k-1) \]
\[ \text{ct} \leftarrow (A[k-i-1], b[k-i-1] - \text{msg} \times 2^{k-i}) \]
\[ m[i] \leftarrow \text{Dec}(s, \text{ct}) \]
\[ \text{msg} \leftarrow \text{msg} + m[i] \ll (i) \]
\[ \text{return } \text{msg} \]
Decryption algorithm

- \( \text{Enc}'(s,\text{msg}) = \text{LWE}(s,\text{msg} \cdot \text{pow2row}) = (A, b) \) where 
  
  \( b = sA + e + \text{msg} \cdot \text{pow2row} \)

- \( \text{Dec}'(s, (A, b)) : \)
  
  \begin{align*}
  \text{msg} &\leftarrow 0 \\
  \text{for } i = 0 \ldots (k - 1) \\
  \text{ct} &\leftarrow (A[k-i-1], b[k-i-1] - \text{msg} \cdot 2^{k-i}) \\
  m[i] &\leftarrow \text{Dec}(s, \text{ct}) \\
  \text{msg} &\leftarrow \text{msg} + m[i] \lll (i) \\
  \text{return } \text{msg}
  \end{align*}

Theorem

\((\text{Gen}, \text{Enc}', \text{Dec}')\) is a valid encryption algorithm for \( \beta = q / 4 \)

Question

Does a similar algorithm work for \text{pow2col}?
Arbitrary linear transformations

- Starting point: \( \text{Enc}() \) linearly homomorphic for small \( t \)
  - \( \text{Enc}(P,m) \times t \approx \text{Enc}(P,mt) \)
  - problem: error grows by a factor \( t \)
Arbitrary linear transformations

- Starting point: $\text{Enc}()$ linearly homomorphic for small $t$
  - $\text{Enc}(P,m) \cdot t \approx \text{Enc}(P,mt)$
  - problem: error grows by a factor $t$

- What about computations modulo $q$?
  - $\text{pow2row} = [1,2,4,8,...]$  
  - $\text{Enc}'(s, \text{msg}) = \text{LWE}(s, \text{msg} \cdot \text{pow2row}) = (A,b)$

- Multiplying by any $t \in \mathbb{Z}_q$
  - Compute $t\text{Bin[]} = \text{bitDecomp}(t)$
  - Compute scalar product with vector $t\text{Bin}[]$
Correctness of scalar multiplication

\[ \text{Enc}'(s, \text{msg}) \ast t \text{Bin}[] \]
\[ = \text{LWE}(s, \text{msg} \ast \text{pow2row}; e) \ast \text{tBin}[] \]
\[ = \text{LWE}(s, \text{msg} \ast \text{pow2row} \ast \text{tBin}[]; e \ast \text{tBin}[]) \]
\[ = \text{LWE}(s, \text{msg} \ast t; e') \]

**pow2row** \* t \text{Bin}[] = \sum_i 2^i \cdot \text{tBin}[i] = t
Correctness of scalar multiplication

\[ \text{Enc}'(s, \text{msg}) \ast t\text{Bin}[] \]
\[ = \text{LWE}(s, \text{msg} \ast \text{pow2row}; e) \ast t\text{Bin}[] \]
\[ = \text{LWE}(s, \text{msg} \ast \text{pow2row} \ast t\text{Bin}[]; e \ast t\text{Bin}[]) \]
\[ = \text{LWE}(s, \text{msg} \ast t; e') \]

- \text{pow2row} \ast t\text{Bin}[] = \sum_i 2^i \cdot t\text{Bin}[i] = t
- \text{if } |e| < \beta, \text{ then } |e'| = |\sum_i e_i \cdot t\text{Bin}[i]| \leq k \cdot \beta
- \text{Error grows only by } k = \log q
Correctness of scalar multiplication

\[ \text{Enc}'(s, \text{msg}) \ast \text{tBin[]} = \text{LWE}(s, \text{msg} \ast \text{pow2row}; e) \ast \text{tBin[]} = \text{LWE}(s, \text{msg} \ast \text{pow2row} \ast \text{tBin[]}; e \ast \text{tBin[]}) = \text{LWE}(s, \text{msg} \ast \text{t}; e') \]

- **pow2row** \( \ast \) \text{tBin[]} = \sum_i 2^i \cdot \text{tBin}[i] = t

- If \( |e| < \beta \), then \( |e'| = |\sum_i e_i \cdot \text{tBin}[i]| \leq k \cdot \beta \)

- Error grows only by \( k = \log q \)

- **Problem:**
  - Result \( \text{msg} \ast \text{t} \) is a value modulo \( q \)
  - \( \text{Enc}(s, \text{msg} \ast \text{t}; e') \) is not properly encoded
  - We need an encryption of \( \text{msg} \ast \text{t} \ast \text{pow2row} \)
Constant Multiplication algorithm

- Enc'(s, msg) = LWE(s, msg*\text{pow2row})
- Enc'(s, msg)* \text{bitDecomp}(t) = LWE(s, msg*t; e')

CMul(C, t):

\[
T = \text{bitDecomp}(t * \text{pow2row})
\]

\text{return } C * T

Proof:
Extensions and Generalizations

- **Matrix messages**
  
  \[ M \otimes \text{pow2row} = [M, M \times 2, M \times 4, M \times 8, \ldots] \]

- **Arbitrary message modulus:**
  
  \[ \text{round}(m \times (q/p), m \times (q/p)/2, m \times (q/p) \times 4, \ldots) \]

- **Other gadgets, e.g., based on Chinese Remainder Theorem**
  
  - \( q = \prod_i p_i \) product of small primes
  - encoding vector \( \text{crtRow} = [q/p_1, q/p_2, \ldots, q/p_k] \)
  - \( \text{crtRow} \times \text{crtDecomp}(t) = t \)
At this point we have an encryption algorithm

\[ \text{Enc}'(S,M) = \text{LWE}(S,M \otimes \text{pow2row}) \]

with message space \( \mathbb{Z}_q^{w \times l} \), and supporting the homomorphic evaluation of the following operations:

- \textbf{Const}(M): noiseless encryption of \( M \)
- (+): addition of ciphertexts
- (-): subtraction of ciphertexts
- \textbf{CMul}(.,T): multiplication by any linear transformation modulo \( q \)
Section 6

Key Switching
Remember Proxy Re-encryption?

- Primary key: \((pk, sk)\)
- Secondary key: \((pk1, sk1)\)
- Re-encryption key: \(rk = \text{Enc}(pk1, sk[1..k])\)
- Input ciphertext: \(c = \text{Enc}(pk, m)\)
- Decryption function: \(f_c(sk) = \text{Dec}(sk, c)\)

**Question**

*What is the result of the following computation?*

\[\text{Eval}(pk1, f_c, rk)\]
Remember Proxy Re-encryption?

- Primary key: \((pk, sk)\)
- Secondary key: \((pk1, sk1)\)
- Re-encryption key: \(rk = Enc(pk1, sk[1..k])\)
- Input ciphertext \(c = Enc(pk, m)\)
- Decryption function \(f_c(sk) = Dec(sk, c)\)

**Question**

What is the result of the following computation?

\(Eval(pk1, f_c, rk)\)

**Question**

Can you implement proxy re-encryption using LWE?
LWE-based Proxy Re-encryption?

\[
\begin{align*}
\text{sk}[1..n] & \in \mathbb{Z}_q^n \\
\text{sk}'[1..n] & \in \mathbb{Z}_q^n \\
\text{Enc}(\text{sk}, \text{msg}) & = \text{LWE}(\text{sk}, \text{msg} \cdot \text{pow2row}) = (A[], b[]) \\
\text{rk}[i] & = \text{Enc}(\text{sk}', \text{sk}[i]) \\
\text{Dec}'(\text{sk}, (A, b))[j] & = b[j] - \sum_i \text{sk}[i] \cdot A[i,j] \\
& \approx \text{msg} \cdot \text{pow2row} \\
\text{Dec}(\text{sk}, (A, b)) & = \text{decode}(\text{Dec}'(\text{sk}, (A, b)))
\end{align*}
\]

**Question**

*Can you compute \text{Dec}' homomorphically?*

*Does it give you a proxy re-encryption scheme?*
LWE-based Proxy Re-encryption

\[
\begin{align*}
\text{Enc}(sk, msg) &= \text{LWE}(sk, msg \times \text{pow2row}) \\
\text{rk}[i] &= \text{Enc}(sk', sk[i]) \\
\text{Dec}'(sk, (A, b)) &= b[j] - \sum_i sk[i] \times A[i, j]
\end{align*}
\]

Goal: homomorphically evaluate the function

\[
f_{A, b}(sk) = \text{Dec}'(sk, (A, b)) \\
\text{Eval}(f_{A, b}, rk) = ?
\]
LWE-based Proxy Re-encryption

\[
\text{Enc}(\text{sk}, \text{msg}) = \text{LWE}(\text{sk}, \text{msg} \cdot \text{pow2row})
\]
\[
rk[i] = \text{Enc}(\text{sk}', \text{sk}[i])
\]
\[
\text{Dec}'(\text{sk}, (A, b)) = b[j] - \sum_i \text{sk}[i] \cdot A[i, j]
\]

Goal: homomorphically evaluate the function

\[
f_{A,b}(\text{sk}) = \text{Dec}'(\text{sk}, (A, b))
\]
\[
\text{Eval}(f_{A,b}, \text{rk}) = \text{?}
\]

Solution: \(\text{Eval}(f_{A,b}, \text{rk}) = \text{ct}\)

\[
\text{ct}[j] = \text{Const}(b[j]) - \sum_i \text{CMul}(rk[i], A[i,j])
\]
Key Switching

- Generalize proxy re-encryption:
  - $sk, sk'$ may have different dimensions and moduli
  - $Enc(sk, ..), Enc'(sk', ..)$ may use different plaintext moduli and message encodings

- Example
  - Message space $msg: \mathbb{Z}_p$
  - Ciphertext modulus $q$
  - $sk[1..n], sk'[1..n] \in \mathbb{Z}_q^n$
  - $Enc(sk, m) = LWE(sk, (q/p) \times msg) \mod q$
  - Evaluation key: $rk[i] = Enc(sk', sk[i])$

- Do you see any problem?
Key Switching

Source scheme:

\[
\begin{align*}
\text{msg} & : \mathbb{Z}_p \\
\text{sk}[1..n] & \in \mathbb{Z}_q^n \\
\text{Enc}(\text{sk}, \text{msg}) & = \text{LWE}(\text{sk}, \frac{q}{p} \text{msg}) = (a[], b) \mod q
\end{align*}
\]

Target scheme:

\[
\begin{align*}
\text{msg}' & : \mathbb{Z}_q \\
\text{sk}'[1..n'] & \in \mathbb{Z}_q^{n'} \\
\text{Enc}'(\text{sk}', \text{msg}') & = \text{LWE}(\text{sk}', \text{msg}' \star \text{pow2row})
\end{align*}
\]

Evaluation:

\[
\begin{align*}
\text{ek}[i] & = \text{Enc}'(\text{sk}', \text{sk}[i]) \\
\text{KeySwitch}(\text{ek}, (a[], b)) & = \\
& \text{Const}(b) - \sum_i \text{CMul}(a[i], \text{ek}[i])
\end{align*}
\]
Correctness

\[
\text{msg: } \mathbb{Z}_p; \ sk[1..n] \in \mathbb{Z}_q^n \\
\text{msg': } \mathbb{Z}_q; \ sk'[1..n'] \in \mathbb{Z}_q^{n'} \\
\]

\[
\text{Enc}(sk,\text{msg}) = \text{LWE}(sk, \frac{q}{p} \text{msg}) = (a[],b) \mod q \\
\text{Enc'}(sk',\text{msg'}) = \text{LWE}(sk',\text{msg'}*=\text{pow2row}) \\
\]

\[
ek[i] = \text{Enc'}(sk',sk[i]) \\
\]

\[
\text{KeySwitch}(ek,(a[],b)) \\
= \text{Const}(b) - \sum_i \text{CMul}(a[i],ek[i]) \\
\]
Correctness

\[
\text{msg: } \mathbb{Z}_p; \; \text{sk}[1..n] \in \mathbb{Z}_q^n \\
\text{msg': } \mathbb{Z}_q; \; \text{sk}'[1..n'] \in \mathbb{Z}_q^{n'}
\]

\[
\text{Enc}(\text{sk}, \text{msg}) = \text{LWE}(\text{sk}, \frac{q}{p} \text{msg}) = (a[], b) \mod q \\
\text{Enc}'(\text{sk}', \text{msg'}) = \text{LWE}(\text{sk}', \text{msg'}*\text{pow2row})
\]

\[
\text{ek}[i] = \text{Enc}'(\text{sk}', \text{sk}[i])
\]

\[
\text{KeySwitch}(\text{ek}, (a[], b)) = \text{Const}(b) - \sum_i \text{CMul}(a[i], \text{ek}[i])
\]

\[
= \text{Const}(b) - \sum_i \text{CMul}(a[i], \text{Enc}'(\text{sk}', \text{sk}[i]))
\]
**Correctness**

\[
\begin{align*}
\text{msg: } & \mathbb{Z}_p; \ sk[1..n] \in \mathbb{Z}_q^n \\
\text{msg': } & \mathbb{Z}_q; \ sk'[1..n'] \in \mathbb{Z}_q^{n'} \\
\text{Enc}(sk, msg) = \text{LWE}(sk, \frac{q}{p} \cdot msg) = (a[], b) \mod q \\
\text{Enc}'(sk', msg') = \text{LWE}(sk', msg' \cdot \text{pow2row}) \\
\text{ek}[i] = \text{Enc}'(sk', sk[i]) \\
\text{KeySwitch}(ek, (a[], b)) &= \text{Const}(b) - \sum_i \text{CMul}(a[i], ek[i]) \\
&= \text{Const}(b) - \sum_i \text{CMul}(a[i], \text{Enc}'(sk', sk[i])) \\
&= \text{LWE}(sk', b - \sum_i a[i] \cdot sk[i]) \\
&= \text{LWE}(sk', \frac{q}{p} \cdot msg + e) \\
&= \text{Enc}(sk', msg)
\end{align*}
\]
Remarks

- Source and Target schemes may use different moduli
  
  \[
  \text{Enc}'(sk', msg') = LWE(sk', \frac{q}{q} msg' \times \text{pow2row})
  \]
Remarks

- Source and Target schemes may use different moduli
  - \( \text{Enc}'(sk', msg') = \text{LWE}(sk', \frac{q}{q} \text{msg}' \times \text{pow2row}) \)

- Input ciphertext may use compact (matrix) LWE
  - \( \text{Enc}(SK, msg[]) = \text{LWE}(SK, \frac{q}{p} \text{msg[]}) \)
  - \( \text{RK}' = \text{Enc}'(SK', SK) \)
Source and Target schemes may use different moduli

\[ \text{Enc}'(sk', msg') = \text{LWE}(sk', \frac{q}{q'} \text{msg'} \times \text{pow2row}) \]

Input ciphertext may use compact (matrix) LWE

\[ \text{Enc}(SK, \text{msg[]}) = \text{LWE}(SK, \frac{q}{p} \text{msg[]}) \]
\[ \text{RK}' = \text{Enc}'(SK', SK) \]

Key Switching:

\[ \text{Input: Enc}(sk, \text{msg: mod p): mod q} \]
\[ \text{Switching Key: Enc}'(sk', sk: \text{mod q'): mod q'} \]
\[ \text{Output: Enc}(sk', \text{msg: mod p): mod q'} \]
Remarks

- Source and Target schemes may use different moduli
  - \( \text{Enc}'(sk', \text{msg}') = \text{LWE}(sk', \frac{q}{q'} \text{msg}' \cdot \text{pow2row}) \)

- Input ciphertext may use compact (matrix) LWE
  - \( \text{Enc}(SK, msg[]) = \text{LWE}(SK, \frac{q}{p} \text{msg[]}) \)
  - \( RK' = \text{Enc}'(SK', SK) \)

- Key Switching:
  - Input: \( \text{Enc}(sk, \text{msg}: \text{mod p}): \text{mod q} \)
  - Switching Key: \( \text{Enc}'(sk', sk: \text{mod q}): \text{mod q'} \)
  - Output: \( \text{Enc}(sk', \text{msg}: \text{mod p}): \text{mod q'} \)

- Input/Output can use arbitrary encoding, e.g.,
  - Input: \( \text{Enc}(sk, \text{msg}) = \text{LWE}(sk, \text{msg} \cdot \text{pow2row}) \)
  - Output: \( \text{Enc}(sk', \text{msg}) = \text{LWE}(sk', \text{msg} \cdot \text{pow2row}) \)
Sub-key Switching

- Application: reduce key size $SK \rightarrow SK'$
- Always: $SK, SK'$ must have the same number of rows
- Often $SK$ is a “sub-matrix” of $SK = [SK', SK'']$
- Switching Key

$$[RK', RK''] = Enc'(SK', SK)$$
$$= Enc'(SK', [SK' | SK''])$$
$$= [Enc'(SK', SK') | Enc'(SK', SK'') ]$$

- But $RK'$ is publicly known! (remember circular security?)
- Can use a smaller switching key $RK'' = Enc'(SK', SK'')$

**Question**

Does it work? What if $SK''=[]$? Then, $RK''=[]$ and $SK = SK'$!

Is it trivial? Is it useful?
Modulus switching

- Subkey switching from \(SK\) to \(SK' = SK\) can still be useful to change the ciphertext modulus from \(q\) to \(q'\).

- So far we used the simplifying assumption that \(p | q\).

- Switching from \(q\) to \(q'\) requires a switching key with:
  - plainext modulus \(q\)
  - ciphertext modulus \(q'\)
  - but if \(q | q'\), this only allows to increase the modulus

- (Sub-)Key Switching works also for \(p \not| q\)
  - but introduces a “small” rounding error
  - for subkey switching the rounding error if proportional to \(SK\)
  - switching to a smaller modulus requires “small” key \(SK\)
Subkey and Modulus Switching

- **Subkey switching**
  - Input: \( ct = Enc([SK', SK''], m) \) and \( RK = Enc'(SK', SK'') \)
  - \( SubkeySwitch(RK, ct) = ct' \) such that \( Dec(SK', ct') = m \)
Subkey and Modulus Switching

- **Subkey switching**
  - **Input:** \( ct = \text{Enc}([SK', SK''], m) \) and \( RK = \text{Enc}'(SK', SK'') \)
  - **SubkeySwitch\( (RK, ct)\) = \( ct' \) such that \( \text{Dec}(SK', ct') = m \)

**Question**

Give explicit description of **SubkeySwitch algorithm**
Subkey and Modulus Switching

- **Subkey switching**
  - Input: \( ct = \text{Enc}([SK',SK''],m) \) and \( RK = \text{Enc}'(SK',SK'') \)
  - \( \text{SubkeySwitch}(RK,ct) = ct' \) such that \( \text{Dec}(SK',ct') = m \)

**Question**

Give explicit description of SubkeySwitch algorithm

- **Modulus switching**
  - Assume \( SK \) has small entries
  - Input: \( ct = \text{Enc}(SK,m) \mod q \) and nothing else
  - \( \text{ModSwitch}(ct) = ct' \mod q' \) such that \( \text{Dec}(SK,ct') = m \)
Subkey and Modulus Switching

- **Subkey switching**
  - Input: $ct = \text{Enc}([SK', SK''], m)$ and $RK = \text{Enc}'(SK', SK'')$
  - $\text{SubkeySwitch}(RK, ct) = ct'$ such that $\text{Dec}(SK', ct') = m$

**Question**

Give explicit description of SubkeySwitch algorithm

- **Modulus switching**
  - Assume $SK$ has small entries
  - Input: $ct = \text{Enc}(SK, m) \mod q$ and nothing else
  - $\text{ModSwitch}(ct) = ct' \mod q'$ such that $\text{Dec}(SK, ct') = m$

**Question**

Give explicit description of ModSwitch algorithm
Section 7

Multiplication
What we have done so far

Simple LWE Encryption: private key encryption supporting

- small message modulus \((p \ll q)\)
- homomorphic addition
- homomorphic multiplication by small constants
- enough to obtain public key encryption
- circular security (for small keys)

Extended LWE Encryption to support

- large message modulus \((p = q)\)
- homomorphic multiplication by arbitrary constants
- circular security (for arbitrary keys)
- key switching
Next: Homomorphic Multiplication

Problem

Given $\text{Enc}(sk, \text{msg}[0])$ and $\text{Enc}(sk, \text{msg}[1])$, compute a ciphertext $ct$ such that $\text{Dec}(sk, ct) = \text{msg}[0] \times \text{msg}[1]$

- Can this be done for our LWE encryption scheme?
- Can it be done with the help of some additional key material?
- Yes, in fact, there are multiple ways to do it
  - Nested encryption
  - Homomorphic decryption
  - Tensor product
Method 1: Nested Encryption

- $\text{msg}[0], \text{msg}[1] \in \mathbb{Z}_q$
- $\text{ct}[0] = \text{Enc}(\text{sk}[0], \text{msg}[0])$
- $\text{ct}[1] = \text{Enc}(\text{sk}[1], \text{msg}[1])$
- Multiply encryption of $\text{msg}[0]$ by $\text{ct}[1]$

\[
\text{ct}[0] \ast \text{ct}[1] = \text{Enc}(\text{sk}[0], \text{msg}[0]) \ast \text{ct}[1] = \text{Enc}(\text{sk}[0], \text{msg}[0] \ast \text{ct}[1])
\]

- Inner multiplication:

\[
\text{msg}[0] \ast \text{ct}[1] = \text{msg}[0] \ast \text{Enc}(\text{sk}[1], \text{msg}[1]) = \text{Enc}(\text{sk}[1], \text{msg}[0] \ast \text{msg}[1])
\]

- Final result: $\text{Enc}(\text{sk}[0], \text{Enc}(\text{sk}[1], \text{msg}[0] \ast \text{msg}[1]))$
Details

- ct[1] = Enc(sk[1], msg[1]) is a vector!
  - ct[0] = Enc(sk[0], msg[0]*I)
  - (msg[0]*I)*ct[1] = msg[0]*ct[1]

- msg[0]*Enc(sk[1], msg[1]; e[1]) = Enc(sk[1], msg[0]*msg[1]; msg[0]*e[1])
  - Assume msg[0] is small (e.g., 10, 1)
  - May set Enc(sk[1], msg[1]) = LWE(sk[1], (q/2)*msg[1])

- Using Enc(sk[0], Enc(sk[1], msg))
  - Keep nesting?
  - Ciphertexts get larger and larger!
Key Nesting

- Recall: $\text{Enc}(S, M) = \text{LWE}(S, M) = (A, S \cdot A + E + M)$
- Claim: Nested encryption $\text{Enc}(Z, \text{Enc}(S, M)) = \text{Enc}(Z \odot S, M)$

**Question**

*For what key $Z \odot S$?*
Key Nesting

- Recall: $\text{Enc}(S, M) = \text{LWE}(S, M) = (A, S \times A + E + M)$
- Claim: Nested encryption $\text{Enc}(Z, \text{Enc}(S, M)) = \text{Enc}(Z \diamond S, M)$

**Question**

*For what key $Z \diamond S$?*

- $S : \mathbb{Z}[k, n], Z : \mathbb{Z}[n+k, n]$
- $Z = (Z_n, Z_k)$ where $Z_n[n, n]$ and $Z_k[k, n]$
- $Z \diamond S = [S \times Z_n + Z_k, S] = [S, I]Z$
  - $\text{Enc}(Z, \text{Enc}(S, m; e); e') = \text{Enc}(Z \diamond S, m; e'')$
  - $e'' = e + [S, I]e'$
- Key $S$ needs to be small!
Nested Encryption + (Sub)Key Switching

Combine nested multiplication with key switching:

- Input keys: $Z, S$
- Evaluation key: $W = \text{Enc}(S,[S,I]Z;F)$
- Input ciphertexts:
  - $CT[0] = \text{Enc}(Z, \text{msg}[0]*I;E[0])$
  - $CT[1] = \text{Enc}(S, \text{msg}[1]*I;E[1])$
- Output: $\text{SubkeySwitch}(W,CT[0]*CT[1]) = \text{Enc}(S, \text{msg}*I;E)$
  - $\text{msg} = \text{msg}[0]*\text{msg}[1]$
  - $E = \text{msg}[0]*E[1] + [S,I]*E[0]*X + F*Y$ for binary matrices $X,Y$
- Key $S$ needs to be small!
- Security Assumption: Standard LWE
Method 1.5: Homomorphic Decryption

- Assume both ciphertexts use the same key $S$
- Nested Encryption:
  1. Homomorphic multiplication: $\text{Enc}(S, \text{msg}[0]) \times \text{CT}[1]$
  2. Key Switching: Homomorphic multiplication by $[S, I]$
- Method 1: $\text{Eval}([S, I], \text{Enc}(S, \text{msg}[0]) \times \text{CT}[1])$
- Combine the two homomorphic multiplications:
  - Bring $[S, I]$ inside the first ciphertext
  - $\text{Enc}(S, \text{msg}[0] \times [S, I]) \times \text{CT}[1]$
- Define a new LWE encryption variant:
  $\text{Enc}^#(S, \text{msg}) = \text{Enc}(S, \text{msg} \times [S, I])$

\[
\begin{align*}
\text{Enc}^#(S, \text{msg}[0]) \times \text{Enc}(S, \text{msg}[1]) \\
= \text{Enc}(S, \text{msg}[0] \times [S, I] \times \text{Enc}(S, \text{msg}[1])) \\
= \text{Enc}(S, \text{msg}[0] \times \text{msg}[1])
\end{align*}
\]
Security

\[ \text{Enc}^*(S, \text{msg}) = \text{Enc}(S, \text{msg}*[S,I]) = [\text{Enc}(S, \text{msg}*S), \text{Enc}(S, \text{msg}*I)] \]

- Circular security:
  - Can compute \( \text{Enc}(S, \text{msg}*S) = \text{msg}*\text{Enc}(S,S) \) without knowing \( S \)
  - Problem: \( \text{msg}*(-I,0) \) reveals \( \text{msg} \)!
  - Solution: \( \text{Enc}(S,0) + \text{msg}*\text{Enc}(S,S) \)

\textbf{Theorem}

\textit{Enc is secure under the LWE assumption}
Remarks

- Second encryption scheme can be chosen arbitrarily
  \[ \text{Enc}^*(S, m_0) \times \text{Enc} (S, m_1) = \text{Enc} (S, m_0 m_1) \]
  \[ \text{Enc}^*(S, m_0) \times \text{Enc}^*(S, m_1) = \text{Enc}^*(S, m_0 m_1) \]

- No need for key switching
  - Product \( \text{Enc}(S, m_0 m_1) \) uses the same key as the input
  - Key \( S \) does not have to be small
  - No evaluation key!

- \( \text{Enc} \) is a homomorphic encryption scheme supporting
  - Ciphertext addition
  - Ciphertext multiplication
  - without any evaluation key!

- Too good to be true?
Error growth

- \( \text{Enc}^\#(m_0; E_0) \times \text{Enc}^\#(m_1; E_1) = \text{Enc}^\#(m_0 m_1; E) \)
- Error: \( E \approx m_0 \times E_1 + E_0 \times X \)

- Multiplying many ciphertexts
  - CT\[i\] = \( \text{Enc}^\#(m_i; E_i) \)
  - Assume \( m_i \in 0, 1 \)
  - Given CT\[1\], \ldots, CT\[k\]
  - Goal: compute \( \text{CT}[1]\times\ldots\times\text{CT}[k] = \text{Enc}^\#(\prod_i m_i) \)

- How? Several options (multiplication is associative):
  - Left to right multiplication chain
  - Right to left multiplication chain
  - Binary tree (minize circuit depth)

Question

*What order is best?*
Arithmetic and Boolean operations

- **Addition**
  - Can add polynomially many ciphertexts
  - Error grows by polynomial factor (e.g., $O(\log(n))$ bits)

- **Multiplication**
  - Assume **binary** message space
  - Can multiply polynomially many ciphertexts in a **chain**
  - Error grows by polynomial factor (e.g., $O(\log(n))$ bits)
Arithmetic and Boolean operations

- **Addition**
  - Can add polynomially many ciphertexts
  - Error grows by polynomial factor (e.g., $O(\log(n))$ bits)

- **Multiplication**
  - Assume **binary** message space
  - Can multiply polynomially many ciphertexts in a **chain**
  - Error grows by polynomial factor (e.g., $O(\log(n))$ bits)

- **Bit operations:**
  - $m_0, m_1 \in \{0, 1\}$
  - $m_0 \land m_1 = m_0 \cdot m_1$
  - $\neg m_0 = 1 - m_0$
  - $m_0 \lor m_1 = \neg(\neg m_0 \land \neg m_1)$

- **Conditional:** $(b, m_0, m_1) \mapsto m_b$
  - $m_b = (1 - b) \cdot m_0 + b \cdot m_1$

- **Arbitrary log-depth boolean circuits**
Method 2: Tensor and Key Switch

- Why? Efficiency! Allows SIMD operations using polynomial rings

- Ciphertext as a function
  - \( f_C(S) = \text{Dec}'(S, C) = [S, I]C \)
  - \( f_C \) is linear in \([S, I]\)

- Product ciphertext \( C = C_0 \ast C_1 \)
  - Goal: \( \text{Dec}'(S, C) = \text{Dec}'(S, C_0) \ast \text{Dec}'(S, C_1) \)
  - \( f_{C_0, C_1}(S) = \text{Dec}'(S, C_0) \ast \text{Dec}'(S, C_1) \) is bilinear in \([S, I]\)

- Tensor product: \( Z = [S, I] \otimes [S, I] = [S \otimes S, S, S, I] \)
  - Any bilinear function of \([S, I]\) is linear in \( Z \)
  - \( C = C_0 \otimes C_1 \) decrypts to \( m_0 \cdot m_1 \) under \( Z \)
Mixed product property

**Theorem**

For any $A, B, X, Y$,

$$(A \otimes B) \cdot (X \otimes Y) = (A \cdot X) \otimes (B \cdot Y)$$
Mixed product property

**Theorem**

For any $A, B, X, Y$,

$$(A \otimes B) \cdot (X \otimes Y) = (A \cdot X) \otimes (B \cdot Y)$$

$$([S, l] \otimes [S, l]) \cdot (C_0 \otimes C_1) = ([S, l]C_0) \otimes ([S, l]C_1)$$

$$= (X_0 + E_0) \otimes (X_1 + E_1)$$

Result: $X_0 \otimes X_1 + X_0 \otimes E_1 + E_0 \otimes X_1 + E_0 \otimes E_1$

- Assume scalar messages: $x_0 \otimes x_1 = x_0 \cdot x_1$
- Messages must be encoded: $x_i = \frac{q}{p} m_i$
Encoding issues

- Encode scalar messages: \( x_i = \frac{q}{p} m_i \)
- Product: \((x_0 + e_0)(x_1 + e_1) = x_0x_1 + x_0e_1 + e_0x_1 + e_0e_1\)
- Issues:
  - Error terms \( x_0e_1 + e_0x_1 = \frac{q}{p}(m_0e_1 + e_0m_1) \) are too large
  - Main term \( x_0x_1 = (q/p)^2 m_0m_1 \) is not properly encoded
- Solutions:
  - **Modular arithmetics**: assume \( \gcd(q, p) = 1 \), and multiply result by \( p \mod q \)
  - **Modulus lifting**: Compute the product modulo \( q^2 \), and then switch to smaller modulus \( q \)
Modular arithmetics

- Compute $c = p \cdot c_0 \otimes c_1 \pmod{q}$
- Output $c$ decrypts (under $sk \otimes sk$) to

$$p(x_0 + e_0)(x_1 + e_1) = \frac{q}{p}(-qm_0m_1) + pe_0e_1$$
Modular arithmetics

- Compute $c = p \cdot c_0 \otimes c_1 \pmod{q}$
- Output $c$ decrypts (under $sk \otimes sk$) to

$$p(x_0 + e_0)(x_1 + e_1) = \frac{q}{p}(-qm_0 m_1) + pe_0 e_1$$

- Assume $q = -1 \pmod{p}$
  - Error growth: $\beta \mapsto p\beta^2$

- Arbitrary $q, p$
  - Multiply result by $(−q)^{-1} \pmod{p}$
  - Error growth: $\beta \mapsto p^2\beta^2$

- Modulus switching can be used to reduce $\beta$ to a fixed polynomial $\sigma = \|s\|_1 = O(n)$, and substantially slow down the error growth
Modulus Lifting

- Compute $c = p \cdot c_0 \otimes c_1 \pmod{q^2}$
- Assume key $\|s\|_1 < \sigma$ has small entries
- Analyze the relative error: $c_i = \text{Enc}(m_i; (q/p)e_i)$

**Theorem**

The product $p(c_0 \pmod{q}) \otimes (c_1 \pmod{q})$ is an encryption of $m_0m_1 \pmod{p}$ under key $s \otimes s \pmod{q^2}$ with error $(q^2/p)e$

$$e \leq 3e_0e_1 + \frac{p}{2}(\sigma + 1)(e_0 + e_1)$$
Relative error growth

- Fixed polynomials $\beta \approx \sqrt{n}$, $\sigma = \|s\|_1 \approx O(n^{1.5})$

- Modulus lifting error growth
  - relative error: input $(q/p)e_i$, output $(q^2/p)e$
  - assume $|e_i| < \epsilon$
  - output (multiplication) error $\approx p\sigma\epsilon$

- After $L$ levels of multiplications, error $\approx (p\sigma)^L\epsilon < 1$

- Input ciphertext modulus must be $q \approx (p\sigma)^L$
  - Better than modular arithmetics approach $q > (p\beta)^{2L}$
  - Similar growth to modular arithmetics + modulus switching
Tensoring + Key Switching

- Both methods produce a ciphertext under key $[S \otimes S, I \otimes S, S \otimes I]$
- For scalar messages $I = [1]$, and $I \otimes S = S \otimes I = S$
- Can use subkey switching from $[S \otimes S, S]$ to $S$
- Evaluation key: $\text{Enc}(S, S \otimes S)$
- Security:
  - Does not follow from circular security of LWE
- Using standard LWE:
  - Evaluation key $\text{Enc}(Z, S \otimes S)$
  - Use a sequence of keys $S_0, ..., S_L$, one for each multiplicative level of circuit/computation
  - Can you still use subkey switching?
Arithmetic computations using Tensor products

- Message encoding: \((q/p)m\)
- Plaintext arithmetic modulo \(p\) (both addition and multiplication)
- Error grows with multiplicative depth of the circuit
- Use small key \(\|s\|_1 < \sigma\) to use modulus switching and slow down error growth
- Error at depth \(L\): \(\approx (p\sigma)^L < q\)
  - \(L = O(\log n): q = n^{O(\log n)}\)
  - \(L = \text{poly}(n): q = 2^{\text{poly}(n)}\)
- Impact of modulus:
  - Efficiency: running time \(\text{poly}(\log q)\)
  - Security: requires hardness of of approximating lattice problems within \(\gamma\approx q/\beta\)
Section 8

FHE!!
Given (1-hop) \((\text{Gen}, \text{Enc}, \text{Dec}, \text{Eval})\) supporting functions 
\(f_{c,c'}(sk) = \text{Dec}(sk,c) \text{ nand } \text{Dec}(sk,c')\)

Define (multi-hop) FHE scheme with \(\text{Func} = \{ \text{nand} \}\)
\[
\text{Gen}'() = (sk,pk) \leftarrow \text{Gen}()
\]
\[
ek \leftarrow \text{Enc}(pk,sk)
\]
\[
\text{return } (sk,(pk,ek))
\]

\[
\text{Enc}'((pk,ek),m) = \text{Enc}(pk,m)
\]

\[
\text{Eval}'((pk,ek),\text{nand},c,c')
\]
\[
= \text{EvalC}(pk,f_{c,c'},ek)
\]
LWE Homomorphic Encryption

- **Goal:** homomorphic evaluation of
  \[ f_{c,c'}(sk) = Dec(sk,c) \text{ nand } Dec(sk,c') \]

- **LWE-based cryptosystem**
  - Supports bounded depth addition and multiplication
  - Bit operations: \( x \text{ nand } y = 1 - (1-x) \cdot (1-y) \)

- **Key Switching**
  \[
  ek[i] = Enc'(sk',sk[i])
  KeySwitch(ek,(a[],b)) =
  Const(b) - \sum_i CMul(a[i],ek[i])
  \]

- **Homomorphic evaluation of** \( Dec'(a,b) = b - Sa \)
Not enough

- Key switching only computes the linear part of $\text{Dec}$
- We also need to round the result to $\text{decode}(b - Sa)$
- Is this really needed?
  - Yes, $b - Sa = (q/p)m + e$
  - Key switching gives a noisy encryption of $(q/p)+e$
  - Without rounding, noise keeps getting bigger

Questions

- Can we express rounding as a polynomial function (mod $q$)?
- What is the degree of the polynomial?
Error growth and bounded computation

We have seen two methods to multiply ciphertexts:

- **Tensor products**
  - error growth \( \sim \beta \rightarrow \beta \sigma \)
  - can evaluate arbitrary circuits with **multiplicative depth** \( L \)
  - even for \( L = \log n \), requires superpolynomial modulus \( q > \sigma^L \approx n^{O(\log n)} \)

- **Nested Encryption / Homomorphic Decryption**
  - asymmetric error growth: \( (m_0, e_0) \times (m_1, e_1) \rightarrow m_0 e_1 + e_0 \beta \)
  - can evaluate arbitrary multiplication **chains** of \( L \) fresh encryptions of **binary** messages
  - even for large \( L \), polynomial modulus \( q \approx L \beta^2 \) is enough
Roadmap

For each multiplication method

1. Describe/analyze a bootstrapping algorithm
2. Homomorphically evaluate the algorithm using an appropriate cryptographic data structure (encrypted accumulator)
3. Implement the cryptographic data structure using LWE
Cryptographic accumulators

- Cryptographic Data Structure $\text{ACC}[v]$
  
  - Holds a value $v \in V$ in encrypted form
  - Input Encryption scheme: $\text{Enc}'$
  - Output Encryption scheme: $\text{Enc}''$

- Operations on $\text{ACC}[v]$
  
  - Given $\text{Enc}'(x)$, update $\text{ACC}[v] \rightarrow \text{ACC}[f(v, x)]$
  - Given $\text{ACC}[v]$, output $\text{Enc}''(f(v))$

- Bootstrapping:
  
  - Bootstrapping key: $\text{Enc}'(s)$
  - Final output: $\text{Enc}''(m)$
Boostrapping problem

- Assume $p = 2, m \in \{0, 1\}$

- Decryption Algorithm:
  - Input: $a[1..n] \in \mathbb{Z}_q^n, b \in \mathbb{Z}_q$
  - Secret key: $s[1..n] \in \mathbb{Z}^n$
  - Compute $d = b - \sum_i a[i]s[i] + (q/4) \pmod{q}$
  - Round $d$ to $\text{MSB}(d) = \lfloor 2d/q \rfloor$

- Homomorphic Computation:
  - Given $\text{Enc}(s[i])$
  - Compute $\text{Enc}(\text{MSB}(d))$

- Simplifying assumption:
  - $s[i] \in \{0, 1\}$
  - without loss of generality using $(a, 2a, 4a, ...)$
Ripple-carry addition

- Standard schoolbook method
  - using binary digits
  - add $n$ numbers at a time
  - *carry* in $\{0, \ldots, n\}$

- Input digits are encrypted
Ripple-carry accumulator

- Parameters:
  - Message space \( V = \{v', \ldots, v''\} \)
  - Input: \( Enc'(x) = Enc^\#(x) \)
  - Output: \( Enc''(x) = LWE(x) \)
  - \( ACC[x] = (Enc''("x=v"): v \in V) \)
  - \( \text{Init}(v) = ACC[v] \)
  - \( \text{Function application: } f(ACC[v]) = ACC[f(v)] \)
  - \( \text{Selection: } Enc'(b) \text{? ACC}[v0] : ACC[v1] = ACC[b?v0:v1] \)
  - \( \text{Output: } p(ACC[v]) = Enc''(p(b)) \)
Bootstrapping algorithm

\[ b + q/4 = \sum_j 2^j b[j] \]
\[ a[i] = \sum_j 2^j a[i,j] \]

\[
\begin{align*}
\text{ACC} & \leftarrow \text{ACC}[0] \\
\text{for } h = 0..k-1 \\
\text{ACC}[x] & \leftarrow f(\text{ACC}[x]) \text{ where } f(x) = (x/2) + b[h] \\
\text{forall } i,j \\
\text{if } (a[i,j] = 1) \\
\text{ACC}[x + s[i]] & \leftarrow \text{Enc}'(s[i]) \ ? \ \text{ACC}[x] : \text{ACC}[x+1] \\
\text{return} \ (\text{even(ACC}[x]) = \text{Enc}''(\text{even}(x)) \end{align*}
\]
Carry-save accumulator

- Parameters: bit length $k$

  - $ACC[x] = (x_0, x_1)$
    - $x = x_0 + x_1 \pmod{2^k}$
    - $x_0[0, \ldots, k-1]$ and $x_1[0, \ldots, k-1]$
    - redundant representation

- Operations:
  - add $y$ to $ACC$
  - compute $MSB(ACC)$
Carry-save addition

Add(ACC(x₀, x₁), y):

\[ x₀'[i] = (x₀[i] + x₁[i] + y[i]) \mod 2 \]

\[ x₁'[i+1] = (x₀[i] + x₁[i] + y[i] > 1) \]

\[ \text{return } \text{ACC}(x₀', x₁') \]
MSB computation

- Standard MSB computation
  - addition $x_0 + x_1$ with carry propagation
  - $O(\log(k))$ depth circuit where $k = \log(q)$

- Can also add in $\log(k)$ depth
  - Compute both $\text{MSB(ACC)}$ and $\text{MSB(ACC+1)}$
  - $\text{ACC}[k]: k$-bit accumulator
  - Recursive algorithm: split
    $\text{ACC}[k] = (\text{HiACC}[k/2], \text{LoACC}[k/2])$

$\text{MSBs(ACC}=(\text{HiACC,LoACC}}))$:
  parallel:
  \[
  \begin{align*}
  \text{hi}[0,1] &= \text{MSBs(HiACC)} \\
  \text{lo}[0,1] &= \text{MSBs(LoACC)}
  \end{align*}
  \]
  $\text{out}[0] = \text{hi}[\text{lo}[0]]$
  $\text{out}[1] = \text{hi}[\text{lo}[1]]$
  $\text{return } \text{out}$
Bootstrapping algorithm

\[ ACC[0] = b + q/4 \]
\[ \text{for } i = 1 \ldots n \]
\[ ACC[i] = s[i] \cdot a[i] \]
\[ ACC = \text{Sum}(ACC[0], \ldots, ACC[n]) \]
\[ \text{return } \text{MSB}(ACC) \]

\[ \text{Sum}(ACC[0..n]) \]
\[ \text{if } n = 0 \]
\[ \text{then return } ACC[0] \]
\[ \text{else } h = n/2 \]
\[ ACC0 = \text{Sum}(ACC[0..h-1]) \]
\[ ACC1 = \text{Sum}(ACC[h..n]) \]
\[ (x0, x1) = ACC1 \]
\[ \text{return } (ACC0 + x0) + x1 \]
Summary

Bootstrapping functions can be computed by

1. $O(n \log q)$-long sequence of multiplications, or
2. $\log(n) + \log \log(q)$-depth arithmetic circuits

Error growth:

1. Using $\text{LWE} \circ$: final error $\approx O(n) \cdot \beta$
2. Using $\text{LWE} \otimes$: final error $\approx \sigma^{\log n + \log \log q} = \sigma^{O(\log n)}$

Parameters $\beta(n), \sigma(n)$: fixed polynomials in $n$

Modulus:

1. polynomial modulus $q(n) \approx O(n^\beta) = n^{O(1)}$
2. quasipolynomial $q(n) = n^{O(\log n)}$
Summary (security)

- Hardness of lattice problems within factor $\gamma \approx q/\beta$
  1. LWE $\odot$: polynomial $\gamma = n^{O(1)}$
  2. LWE $\otimes$: quasipolynomial $\gamma = n^{O(\log n)}$

- Circular security assumption
  - Needed by tensor product multiplication / keyswitching
  - Needed to apply bootstrapping
  - Not needed for leveled homomorphic encryption
Summary (security)

- Hardness of lattice problems within factor $\gamma \approx q/\beta$
  1. $\text{LWE} \circ$: polynomial $\gamma = n^{O(1)}$
  2. $\text{LWE} \otimes$: quasipolynomial $\gamma = n^{O(\log n)}$

- Circular security assumption
  - Needed by tensor product multiplication / keyswitching
  - Needed to apply bootstrapping
  - Not needed for leveled homomorphic encryption

Question

Remove circular security assumption:
- Can you build (unbounded) FHE from standard LWE?
- Can you build (unbounded) linearly homomorphic HE?
Efficiency

- Main security parameter $n > 100$ (typically, $n \approx 1000$)
- Modulus $q(n) < 2^n$ has bitsize $\log q < n$
- Assume 1GHz, arithmetic operations modulo $q$
- Bootstrapping: homomorphically evaluate decryption algorithm (once or twice per gate)

**Question**

*Can you estimate the cost of a single FHE operation?*
Section 9

Project Info
Implementation and Libraries

Libraries:

- SEAL
- HElib
- PALISADE
- Lattigo
- ...

Interface:

- try to hide math as much as possible
- offer encoding, decoding and SIMD operations
Project:

- Use one of the libraries
- Open ended, do anything you want
- Goal: demonstrate you managed to use the library
- Extra points: do something interesting
- Submission: pdf report describing your work + supporting code

Teams:

- You can work in pairs if you like
- Larger teams only if doing something more substantial
- Individual project required to use for master competency
Project Deadlines:

Deadlines:

- Next lecture (Tue, Dec 1): need to know what you are doing (team, library)
- End of finals week (Fri, Dec 18): project submission (canvas, pdf+code)

In the meantime:

- in class, mathematics underlying Ring LWE used by the libraries
- useful to understand/improve the libraries
- not required to use the libraries
Section 10

Ring LWE
(In)efficiency of LWE

Standard LWE

- Ciphertexts: \( (a, b) \in \mathbb{Z}_q^{(n+1) \times \log q} \) store one value (mod \( p \))
- Ciphertext size: \( O(n \log q) \)
- Addition, Scalar multiplication: \( T \approx n \log q \)
- Ciphertext multiplication: \( T \approx n^2 \log^2 q \)

Compact LWE

- Ciphertexts: \( (a, b) \in \mathbb{Z}_q^{(2n) \times \log q} \) store \( n \) values (mod \( p \))
- Amortized ciphertext size: \( O(\log q) \)
- Amortized addition, scalar multiplication: \( T \approx \log q \)
- Ciphertext multiplication?
Ring LWE

- Generalize LWE using a ring $R$ instead of $\mathbb{Z}$
- Ring of polynomials $\mathbb{Z}[X]$
- Monic irreducible $p(X)$ of degree $n$
  - e.g., $p(X) = X^n - 1$
- Quotient ring $R = \mathbb{Z}[X]/p(X)$
  - isomorphic to $(\mathbb{Z}^n, +)$
  - convolution product
  - $R_q = R / qR$

Ring LWE

- Key: $s(X) \in R$
- Ciphertexts $(a, b) \in R_q^2$
- Messages: $m \in R_p$
Ring LWE vs Compact LWE

Both methods:

- Encrypt $n$ values (mod $p$) using $O(n)$ values (mod $q$)
- Efficient (linear time) vector addition and scalar multiplication

Multiplication:

- Compact LWE: tensor multiplication, cost $O(n^2)$
- Ring LWE: polynomial multiplication, cost $O(n \log n)$ using FFT

Applications / Programming model:

- Addition, scalar multiplication: SIMD
- Multiplication: convolution is usually not what you want
- Encode data to perform SIMD multiplication
Data encoding

- Polynomial representation
  - \( p(x_1), \ldots, p(x_n) \in \mathbb{Z}_q^n \)
  - \( p(x) = a_0 + a_1 x_1 + \ldots + a_{n-1} x^{n-1} \equiv \mathbb{Z}_q^n \)
  - Polynomial multiplication: SIMD multiplication of evaluation representations

- Quasilinear time transformations:
  - \((y_1, \ldots, y_n) \rightarrow (a_0, \ldots, a_{n-1})\): polynomial interpolation
  - \((a_0, \ldots, a_{n-1}) \rightarrow (y_1, \ldots, y_n)\): polynomial evaluation

- Other operations:
  - SIMD: great to run same program on \( n \) data sets
  - Need also to pack, unpack, shuffle, etc. for general computations
Security

- Is Ring LWE secure?
- For what rings?

Short answer:

- Working modulo \( p(X) = X^n - 1 \) is not a good idea
- Better to work with *cyclo
tomic* polynomials
- SWIFFT ring: \( p(X) = X^n + 1 \) where \( n = 2^k \)

Useful both for

- security, pseudorandomness, search/decision reductions
- efficient implementation using Number Theoretic Transform (NTT)
Implementation and Libraries

Libraries:

- SEAL
- HElib
- PALISADE
- Lattigo
- ...

Interface:

- try to hide math as much as possible
- offer encoding, decoding and SIMD operations
Cyclic lattices

- A lattice is cyclic if it is closed under
  \[ \text{rot}(v_1, \ldots, v_n) = (v_n, v_1, v_2, \ldots, v_{n-1}) \]

- Equivalently
  - view vectors as coefficients of a polynomial
  - lattice is closed under \( \text{rot}(v(X)) = X \ast v(X) \mod (X^n - 1) \)

- Commonly used in coding theory (over finite fields)
  - cyclic codes: linear code, closed under rotation
  - equivalently, set of polynomials in \( \mathbb{F}[X]/(X^n - 1) \), closed under multiplication by \( X \)
Theorem

Any cyclic code over finite a field $\mathbb{F}$ can be written as

$$C = \{g(X) \cdot f(X) \mod (X^n - 1) | f(X)\}$$

for some $g(X)$

Proof.
Generators

**Theorem**

*Any cyclic code over finite a field $\mathbb{F}$ can be written as*

$$C = \{g(X) \cdot f(X) \mod (X^n - 1) | f(X)\}$$

*for some $g(X)$*

**Proof.**

**Question**

*Is the same true for cyclic lattices?*
Cyclic lattices and one-way functions

- NTRU (1998): public key encryption, efficient, no proof
- First provable construction, (M., FOCS 2002): one-way function
  - \( R_q = \mathbb{Z}[X]/(q, X^n - 1) \)
  - key: \( a_1(X), \ldots, a_m(X) \in R_q \)
  - input: \( v_1(X), \ldots, v_m(X) \in \{0, 1\}^n \subset R_q \)
  - output: \( w(X) = \sum_i a_i(X) \cdot v_i(X) \in R_q \)
  - compression function: \( m = 2n \log_2(q) \)

- One-way: given \( a_1, \ldots, a_m \) and \( w \),
  - easy to find \( v_1, \ldots, v_m \in R_q \) such that \( \sum_i a_i v_i = w \in R_q \)
  - hard to find \( v_1, \ldots, v_m \in \{0, 1\}^n \)
- Intuition: Compact knapsack, circulant matrices
Compact knapsack, circulant matrices

- Polynomials: \( a(X) \in \mathbb{Z}[X]/(X^n - 1) \)
- Equivalently: \( A \in \mathbb{Z}^{n \times n} \) circulant matrix
  - \( a_1 + a_2 \equiv A_1 + A_2 \)
  - \( a_1 \cdot a_2 \equiv A_1 \cdot A_2 \)
- Compact knapsack
Collision resistance?

- Regular knapsack:
  - given random $a_1, \ldots, a_m \in \mathbb{Z}_q$
  - $m = 2 \log_2(q)$
  - collisions exist
  - collisions are hard to find

- Compact knapsack:
  - given random $a_1, \ldots, a_m \in \mathbb{Z}_q[X]/(X^n - 1)$
  - $m = 2n \log_2(q)$
  - collisions exist

Question

Are collisions hard to find?
Collisions in compact knapsacks

- Multiply each “circulant” matrix $a_i$ by the all-one vector
- Find collision in $\mathbb{Z}_q$
- Algebraic description:
  - multiply each $a_i(X)$ by $u(X) = (1 + X + X^2 + ....)$
  - Notice $(X^n - 1) = u(X) \cdot (X - 1)$
  - CRT: $R \equiv (\mathbb{Z}[X]/(X - 1)) \times (\mathbb{Z}[X]/u(X))$
  - Multiplication by $u(X)$ maps $R$ to $\mathbb{Z}[X]/(X - 1) \equiv \mathbb{Z}$
Anti-Cyclic lattices

- A lattice is anticyclic if it is closed under \( \text{rot}(x_1, ..., x_n) = (-x_n, x_1, x_2, ..., x_{n-1}) \)

- Equivalently: work in \( R = \mathbb{Z}[X]/(X^n + 1) \)

- Questions:
  1. Are compact knapsacks over \( R \) collision resistant?
  2. Does \( (X^n + 1) \) have small degree factors?
Anti-Cyclic lattices

- A lattice is anticyclic if it is closed under 
  $$\text{rot}(x_1, \ldots, x_n) = (-x_n, x_1, x_2, \ldots, x_{n-1})$$

- Equivalently: work in 
  $$R = \mathbb{Z}[X]/(X^n + 1)$$

- Questions:
  1. Are compact knapsacks over \( R \) collision resistant?
  2. Does \((X^n + 1)\) have small degree factors?

**Theorem**

\( X^n + 1 \) is irreducible if and only if \( n \) is a power of \( 2 \)
Roots of Unity

- \( \omega_m = \exp(2\pi i / m) \in \mathbb{C} \), primitive \( m \)th root of unity
- Observation: \( X^m - 1 = \prod_{k=0}^{m-1} (X - \omega_m^k) \)

\[
X^m - 1 = \prod_{d \mid m} \prod_{\gcd(k,m)=d} (X - \omega_m^k) \]

\[
= \prod_{d \mid m} \prod_{k \in \mathbb{Z}_m^*} (X - \omega_m^{k/d})
\]

**Definition**

Cyclotomic Polynomial: \( \Phi_m(X) = \prod_{k \in \mathbb{Z}_m^*} (X - \omega_m^k) \in \mathbb{C}[X] \)

- Question: does \( \Phi_m \) have integer coefficients?
Division Theorem

- $(R, +, *, 0, 1)$: any ring
- $R[X]$: polynomials with coefficients in $X$

**Theorem**

*For any $a(X) \in R[X]$ and monic $b(X) \in R[X]$, there exists unique $q(X), r(X) \in R[X]$ such that*

- $a(X) = q(X) \ast b(X) + r(X)$
- $\deg(r(X)) < \deg(b(X))$
Dividing Algorithm

\[
\text{divRem} :: \text{Poly} \rightarrow \text{Poly} \rightarrow \text{Poly}
\]
\[
\text{divRem} a b =
\]
\[
\text{if } (\deg a < \deg b)
\]
\[
\text{then } (0, a)
\]
\[
\text{else let } aL = \text{leadingTerm} a
\]
\[
 bL = \text{leadingTerm} b
\]
\[
 qL = aL / bL
\]
\[
 a' = a - b * qL
\]
\[
 (q', r) = \text{divRem} a' b
\]
\[
 q = qL + q'
\]
\[
\text{in } \text{divRem} (q, r)
\]

- Dividing by \( b(X) \) requires divisions by the leading coefficient of \( b \)
- If \( R \) is a field, we can divide by any non-zero \( b(X) \):
- If \( b(X) \) is monic, division is possible in any ring \( R \)
Question

Divide $a(X) = 5X^8 + 4X^6 - 5X^3 + 4$ by $b(X) = X^3 - X + 7$
Polynomial Division: Example

Question

Divide \( a(X) = 5X^8 + 4X^6 - 5X^3 + 4 \) by \( b(X) = X^3 - X + 7 \)

Solution:

- quotient: \( q(X) = 5X^5 + 9X^3 - 35X^2 + 9X - 103 \)
- remainder: \( r(X) = 254X^2 - 166X + 725 \)
Remarks about Division Algorithm

- Division Algorithm:
  \[(a(X), b(X) \in R[X]) \mapsto (q(X), r(X) \in R[X])\]

- For any subring \(S \subseteq R\), and \(a(X), b(X) \in S[X]\)
  - Result of dividing \(a(X)\) by \(b(X)\) is in \(S[X]\)
  - Division as polynomials in \(R[X]\) or as polynomials in \(S[X]\) produces the same result
Polynomial GCD

- \( \mathbb{F}[X] \): polynomials with coefficients in a field \( \mathbb{F} \)

- The Greatest Common Divisor (gcd) of \( a(X), b(X) \in \mathbb{F}[X] \) is a polynomial \( g(X) \in \mathbb{F}[X] \) such that
  - \( g(X) \) divides \( a(X) \) and \( b(X) \)
  - any \( d(X) \in \mathbb{F}[X] \) that divides both \( a(X) \) and \( b(X) \) also divides \( g(X) \)

**Theorem**

For any \( a(X), b(X) \in \mathbb{F}[X] \)

\[
\gcd(a(X), b(X)) = u(X)a(X) + v(X)b(X)
\]

for some \( u(X), v(X) \in \mathbb{F}[X] \).
Euclid’s Algorithm

- **Input**: \( a(X), b(X) \in \mathbb{F}[X] \)
- **Output**: \( u(X), v(X) \in \mathbb{F}[X] \) such that
  \[ u(X)a(X) + v(X)b(X) = \gcd(a(X), b(X)) \]
- **Invariant**: \( \gcd(a(X), b(X)) = \gcd(b(X), a(X) \mod b(X)) \)

\[
\text{euclid} :: (\text{Poly}, \text{Poly}) \rightarrow (\text{Poly}, \text{Poly}) \\
\text{euclid} (a,b) = \\
\quad \text{if} \ (\deg b \equiv 0) \\
\quad \text{then} \ (1,0) \\
\quad \text{else let} \ (q,r) = \text{divRem} b a \\
\quad \quad (u,v) = \text{euclid} (b,r) \\
\quad \quad \text{in} (-q*v , u+v) \\
\]

- **Base case**: \( 1*a+0*b = a = \gcd(a,b) \)
- **Induction**: \( (-qv)a+(u+v)b = ub + v(b-qa)= ub+vr \)
Remarks about Euclid Algorithm

\[ \text{euclid} :: (\text{Poly}, \text{Poly}) \rightarrow (\text{Poly}, \text{Poly}) \]

\[ \text{euclid} (a,b) = \]

\[ \begin{align*}
    \text{if} & \ (\deg b \equiv 0) \\
    \text{then} & \ (1,0) \\
    \text{else let} & \ (q,r) = \text{divRem} \ b \ a \\
        \ (u,v) & \ = \ \text{euclid} \ (b,r) \\
        \ \text{in} & \ (-q*v, u+v)
\end{align*} \]

- Euclid Algorithm works over a field:
  - Even if \( b(X) \) is monic, \( r(X) = b(X) \mod a(X) \) may not be
  - If \( a(X), b(X) \in R[X] \) have coefficients in a domain \( R \subseteq F \), then we can compute \( \gcd(a(X), b(X)) \in F[X] \)
Cyclotomic Polynomials

\[ X^m - 1 = \prod_{d|m} \Phi_m(X) \]

**Theorem**

\[ \Phi_m(X) \in \mathbb{Z}[X] \]
Theorem

\[ \phi_m(X) \in \mathbb{Z}[X] \]

Proof:

- For \( m = 1 \), \( \phi_1(X) = (X - 1) \)
- For \( m > 1 \), \( b(X) = \prod_{m > d|m} \phi_d(X) \) is in \( \mathbb{Z}[X] \) by induction
- Compute \((q(X), r(X)) = \text{divRem}(X^m - 1, b(X))\) in \( \mathbb{Z}[X] \)
- \( r(X) = 0 \) because \( b(X) \) divides \( X^m - 1 \)
- \( \phi_m(X) = q(X) \) is in \( \mathbb{Z}[X] \)
Irreducibility of Cyclotomics

Theorem
\[ \Phi_m(X) \in \mathbb{Z}[X] \text{ is irreducible} \]

Theorem
\[ C_m \equiv \mathbb{Z}[X]/\Phi_m(X) = \mathbb{Z}[\omega_m] \]

- simple proof, helps intuition

Algebraic Number Fields
- finite dimensional extensions of \( \mathbb{Q} \)
- key concepts: field extensions, vector spaces

Algebraic Number Rings
- finite dimensional extensions of \( \mathbb{Z} \), i.e., lattices
- key concepts: ring extensions, modules over a ring
Factoring primes in Cyclotomic rings

- $\Phi_m(X) \in \mathbb{Z}[X]$: $m$th cyclotomic polynomial
- $\Phi_m(X)$ is irreducible in $\mathbb{Z}[X]$
- Let $p$ be a prime, and assume $\gcd(m, p) = 1$
- Question: if $\Phi_m(X)$ irreducible also in $\mathbb{Z}_p[X]$?
- Answer: no, and this is very useful

**Question**

*Question: What’s the factorization of $\Phi_m(X)$ modulo $p$?*

Technically, this is the problem of factoring (the ideal generated by) the prime $p$ in the ring of polynomials modulo $\Phi_m(X)$
Factoring primes in Cyclotomic rings

- $\Phi_m(X) \in \mathbb{Z}[X]$: $m$th cyclotomic polynomial
- $\Phi_m(X)$ is irreducible in $\mathbb{Z}[X]$
- Let $p$ be a prime, and assume $\gcd(m, p) = 1$
- Question: if $\Phi_m(X)$ irreducible also in $\mathbb{Z}_p[X]$?
- Answer: no, and this is very useful

**Question**

*Question*: What’s the factorization of $\Phi_m(X)$ modulo $p$?

Technically, this is the problem of factoring (the ideal generated by) the prime $p$ in the ring of polynomials modulo $\Phi_m(X)$

"The obvious mathematical breakthrough would be development of an easy way to factor large prime numbers"

*(Bill Gates, The Road Ahead, p. 265)*
Motivation

- \( R = \mathbb{Z}[X]/\Phi_m(X) \)
- \( R_p = R/(pR) \equiv \mathbb{Z}[X]/\langle \Phi_m(X), p \rangle_{\mathbb{Z}[X]} \)
- Equivalently, \( R_p \equiv \mathbb{Z}_p[X]/\Phi_m(X) \)
- The structure of \( R_p \) is equivalently described by
  - the factorization of \( (pR) \) in \( R \), or
  - the factorization of \( \Phi_m \) in \( \mathbb{Z}_p[X] \)
Section 11

ANT
Basic Algebra

Review of basic algebraic structures:

- (Commutative) monoids and groups
- Rings and Fields
- Modules and Vector spaces

Some common examples:

- $\mathbb{Q} \subset \mathbb{R} \subset \mathbb{C}$: the fields of rational, real and complex numbers
- $\mathbb{Z}, \mathbb{Z}_n$: the rings of integers, and integers modulo $n$
- $R[X]$: The ring of polynomials with coefficients in $R$
Monoids and Groups

- A monoid \((A, *, 1)\) is a set \(A\) with a binary operation \((*) : A \times A \rightarrow A\) and unit element \(1 \in A\) such that
  - \((x * y) * z = x * (y * z)\) (associativity)
  - \(1 * x = x * 1 = x\) (identity)

- A monoid is commutative if
  - \(x * y = y * x\) (commutativity)

- An element \(x\) is invertible if there is a \(y\) such that
  - \(x * y = y * x = 1\)

- A group is a monoid such that all elements are invertible

- Abelian group: commutative groups, additive notation \((A, +, 0)\), additive inverse \(-x\)
Rings and Fields

- A (commutative) Ring \((R, +, *, 1, 0)\) is a set with two binary operations such that
- \((R, +, 0)\) is an abelian group
- \((R, *, 1)\) is a (commutative) monoid
- \(x \ast (y + z) = x \ast y + x \ast z\) and \((x + y) \ast z = x \ast z + y \ast z\) (distributivity)
- Subring \(S \subseteq R\), subset of a ring closed under \(+, *, 0, 1\)
- A commutative ring \((F, +, *, 1, 0)\) such that all nonzero elements are invertible is called a Field
- A subring of a field \(R \subseteq F\) is called an Integral Domain
Modules and Vector Spaces

- Let $(R, +, *, 0, 1)$ be a commutative ring

- An $R$-module is an additive group $(A, +, 0)$ with a scalar multiplication operation $(*) : R \times A \rightarrow A$ such that
  
  - $r * (s * a) = (r * s) * a$
  
  - $(r + s) * a = r * a + s * a$
  
  - $r * (a + b) = r * a + r * b$

- If $R$ is a field, then $A$ is called a Vector Space
  
  - Linear independence
  
  - Dimension
  
  - Basis
Submodules and Quotients

- Let \((A, +, 0)\) be an \(R\)-module

- An \(R\)-submodule of is
  - a subgroup \(B \subseteq A\)
  - closed under scalar multiplication: \(R \ast B \subseteq B\)

- Quotient group: \(A/B = \{[a]_B : a \in A\}\), \([a]_B = a + B\)
  - also an \(R\)-module with \(r \ast [a]_B = [r \ast a]_B\)

- Special case:
  - \(R\) is an \(R\)-module
  - \(R\)-submodules \(I \subseteq R\) are called ideals
  - \(R/I\) is also a ring with \([a] \ast [b] = [a \ast b]\)
Integral and Algebraic Numbers

- Domain $R \subseteq F$: subring of a field $F$
- $\alpha \in F$ is **algebraic** over $R$ if $m(\alpha) = 0$ for some $m(X) \in R[X]$
- $\alpha \in F$ is **integral** over $R$ if $m(\alpha) = 0$ for some **monic** $m(X) \in R[X]$

**Examples:**
- $\alpha = \sqrt{2}$ is integral over $\mathbb{Z}$ because $m(\alpha) = 0$ for $m(X) = X^2 - 2$
- $\alpha = 1/\sqrt{2}$ is algebraic over $\mathbb{Z}$ because $m(\alpha) = 0$ for $m(X) = 2X^2 - 1$, but is it not integral
Minimal Polynomial

- Field Extension \( F \subseteq E \)

- Let \( \alpha \in E \) be algebraic over \( F \)

- Ring homomorphism: \( h_\alpha : F[X] \to E \), where \( h_\alpha(p(X)) = p(\alpha) \)

- \( I = \ker(h_\alpha) \): set of polynomials \( p \) such that \( p(\alpha) = 0 \)

- \( I \subseteq F[X] \) is a non-zero ideal

- **Minimal polynomial**: smallest degree monic polynomial \( m(X) \in I \)

- \( I = F[X] \cdot m(X) \), i.e., \( p(\alpha) = 0 \) iff \( m(X)|p(X) \)
Let \( m(X) \) be the minimal polynomial of \( \alpha \)

\( m(X) \) is irreducible:

- If \( m(X) = a(X) \cdot b(X) \), then \( a(\alpha) \cdot b(\alpha) = m(\alpha) = 0 \),
- either \( a(\alpha) = 0 \) or \( b(\alpha) = 0 \).
- either \( a(X) = c \cdot m(X) \) or \( b(X) = c \cdot m(X) \)

\( F[\alpha] \equiv F[X]/m(X) \) are isomorphic

isomorphism: \( h_\alpha : F[X]/m(X) \to F[\alpha] \)
Algebraic Extensions

- Algebraic $\alpha \in E \subseteq F$
- Minimal polynomial $m(\alpha) = 0$ of degree $n = \deg(m(X))$
- $F[\alpha] \equiv F^n$ as an $F$-vector space with basis $\alpha^0, \alpha^1, \ldots, \alpha^{n-1}$:
  - $\alpha^0, \alpha^1, \ldots, \alpha^{n-1}$ are linearly independent
  - $\alpha^0, \alpha^1, \ldots, \alpha^{n-1}$ generate $F[\alpha]
Extension fields

**Theorem**

\[ F[\alpha] = F(\alpha) \text{ is a field} \]

**Proof:**

- Let \( p(\alpha) \in F[\alpha] \) for some \( p(X) \in F[\alpha], \deg(p) < n \)
- \( \gcd(p(X), m(X)) \in \{1, m(X)\} \) because \( m(X) \) is irreducible
- If \( \gcd = m(X) \), then \( p(X) = m(X) \) and \( p(\alpha) = 0 \)
- If \( \gcd = 1 \), then \( u(X)p(X) + v(X)m(X) = 1 \)
- \( u(\alpha) \cdot p(\alpha) = 1 \)
Factoring primes in Cyclotomic rings

- $\Phi_m(X) \in \mathbb{Z}[X]$: $m$th cyclotomic polynomial
- $\Phi_m(X)$ is irreducible in $\mathbb{Z}[X]$.
- Let $p$ be a prime, and assume $\gcd(m, p) = 1$.
- Question: if $\Phi_m(X)$ irreducible also in $\mathbb{Z}_p[X]$?
- Answer: no, and this is very useful.

**Question**

**Question:** What’s the factorization of $\Phi_m(X)$ modulo $p$?
Since \( \gcd(m, p) = 1 \), we have \( p \in \mathbb{Z}_m^* \)

Let \( d = o(p) \) be the order of \( p \) in \( \mathbb{Z}_m^* \)

\( p^d = 1 \mod m \), equivalently, \( m \mid (p^d - 1) \)

Let \( GF(p^d) \) be the finite field with \( p^d \) elements

The multiplicative group \( GF(p^d)^* \) is cyclic of order \( p^d - 1 \)

There is an element \( \zeta \in GF(p^d)^* \) of order \( m \)

\( \zeta^d = 1 \) in \( GF(p^d) \)

\( o(\zeta^k) = m \) for all \( k \in \mathbb{Z}_m^* \)

\( \Phi_m(X) = \prod_{k \in \mathbb{Z}_m^*} (X - \zeta^k) \) splits in \( GF(p^d) \)
The minimal polynomials of all $\zeta^k$ over $\mathbb{Z}_p$ have degree $d$

Let $l(X) \in \mathbb{Z}_p[X]$ be the minimal polynomial of $\zeta$

$\mathbb{Z}_p[\zeta] \equiv \mathbb{Z}_p[X]/l(X)$ is a field

- of size $p^{\deg(l)}$
- containing an element $\zeta$ of order $m$

$m = o(\zeta)$ divides $p^{\deg(l)} - 1 = |\mathbb{Z}_p[\zeta]^*|$

$p^{\deg(l)} = 1 \mod m$

by definition of $d = o(p)$ and $\deg(l) = d$
When $\gcd(m, p) = 1$ $\Phi_m(X) \in \mathbb{Z}_p[X]$ factors into a product of $\varphi(m)/d$ distinct degree $d = o(p \mod m)$ polynomials.

For arbitrary $m$, factorization of $\Phi_m(X)$ modulo $p$ is obtained using the following theorem.

**Theorem**

For any $m' = mp^k$ with $\gcd(m, p) = 1$,

$$\Phi_{m'}(X) = (\Phi_m(X))^\varphi(p^k) \mod p$$
Proof

- Frobenius map \( (x \mapsto x^p) : GF(p^k) \to GF(p^k) \) satisfies:
  - \( (x + y)^p = x^p + y^p \) (from binomial expansion)
  - \( a^p = a \) for \( a \in \mathbb{Z}_p \subseteq GF(p^k) \) (Lagrange)

- \( \mathbb{Z}_p[X] \) is a domain:
  - \( a(X)b(X) = a(X)c(X) \) cancels to \( b(X) = c(X) \)
Using these two properties:

- \((X^{mp^k} - 1) = (X^m - 1)^{p^k} = \prod_{d|m} \Phi_d(X)^{p^k}\)
- \((X^{mp^k} - 1) = \prod_{d|m} \prod_{i \leq k} \Phi_{dp^i}(X)\)
- So, by induction on \(m:\)
  \[
  \prod_{i \leq k} \Phi_{mp^i}(X) = \Phi_m(X)^{p^k}
  \]

- Canceling equality for \(k - 1\) from equality for \(k:\)
  \[
  \Phi_{mp^k}(X) = \Phi_m(X)^{p^k-p^{k-1}} = \Phi_m(X)^\varphi(p^k)
  \]
Factoring modulo a prime power

- $\Phi_m(X) = \prod_i F_i(X) \mod p$ with irreducible $F_i(X) \in \mathbb{Z}_p[X]$
- Lift each $F_i(X) \mod p$ to a factor $G_i(X) \mod p^k$
- $\Phi_m(X) = \prod_i G_i(X) \mod p^k$ with $F_i(X) = G_i(X) \mod p$
- $G_i(X)$ is irreducible, because any factorization $\mod p^k$ gives also a factorization $\mod p$

**Theorem**

*(Lifting)* Let $a(X)b(X) = c(X) \mod p$ with $\gcd(a(X), b(X)) = 1$. For every $k$, there are $a'(X) = a(X) \mod p$ and $b'(X) = b(X) \mod p$ such that $a'(X)b'(X) = c(X) \mod p^k$
Let $u(X), v(X)$ such that $a(X)u(X) + b(X)v(X) = 1 \mod p$
Assume $a(X)b(X) = c(X) + p^k d(X)$ by induction
Let $a'(X) = a(X) - p^k v(X)d(X)$ and $b'(X) = b(X) - p^k u(X)d(X)$
$a'(X)b'(X) \mod p^{k+1} = a(X)b(X) - p^k(a(X)u(X)+b(X)v(X))d(X) = a(X)b(X) - p^k d(X) = c(X)$