7. Encryptions

We introduced private-key encryption earlier, but public-key encryption is deferred to this section. This is because of the hardness assumption: private-key encryption and other primitives can be based on the existence of OWFs (One-Way Functions), but we need other assumptions to obtain public-key encryption (PKE).

Learning with Errors (LWE)

In this section, we change the representation of $\mathbb{Z}_q$ so that:

$$\mathbb{Z}_q \equiv \{-\lfloor \frac{q-1}{2} \rfloor, \ldots, 0, \ldots, \lfloor \frac{q}{2} \rfloor\}.$$

The modular arithmetic remains the same. We also define an element in $\mathbb{Z}_q$ as *small* if it is close to 0.

We consider a matrix $A \in \mathbb{Z}_q^{m\times n}$ and a vector $\vec{s} \in \mathbb{Z}_q^n$ such that $m \gg n$. Let:

$$\vec{t'} := A \cdot \vec{s} \in \mathbb{Z}_q^m.$$

Using standard linear algebra, it is efficient to solve $A \cdot \vec{x} = \vec{t'}$ (when A is full rank).

The LWE problem considers the following variant: - A and $\vec{s}$ are chosen as above, but an additional *error vector* $\vec{e} \in \mathbb{Z}_q^m$ is sampled such that $\|\vec{e}\|$ is *small* (each coordinate of $\vec{e}$ is small in $\mathbb{Z}_q$). - The goal is to find $\vec{x}, \vec{y}$ such that:

$$A \cdot \vec{x} + \vec{y} = \vec{t},$$

where:

$$\vec{t} := A \cdot \vec{s} + \vec{e}, \quad \vec{x} \in \mathbb{Z}_q^n, \quad \vec{y} \in \mathbb{Z}_q^m \text{ is } \textit{small}.$$

Why is LWE important? The LWE problem is foundational in modern cryptography and has wide-ranging applications: 1. It forms the basis of public-key encryption schemes, such as Regev's encryption. 2. It is a critical building block for Fully Homomorphic Encryption (FHE), enabling secure computation on encrypted data. 3. It underpins post-quantum cryptographic schemes designed to resist quantum attacks.

What does "small" mean? To better understand the term "small," consider a concrete example:

$$\text{Let } q = 101, \text{ then }_q = \{-50, -49, \ldots, 0, \ldots, 49, 50\}.$$

An element $\vec{e} \in_q^m$ is considered "small" if all its coordinates satisfy $|e_i| \leq 5$. For instance:

$$\vec{e} = \begin{bmatrix} 3 \\ -2 \\ 5 \end{bmatrix} \in_q^3 \quad \text{is "small."}$$

Numerical Example of LWE: Consider a small instance of the LWE problem with the following parameters:

$$q = 7, \quad m = 3, \quad n = 2.$$

Define the matrix A and vectors $\vec{s}$ and $\vec{e}$ as:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}, \quad \vec{s} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \quad \vec{e} = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}.$$

Compute:

$$\vec{t} = A \cdot \vec{s} + \vec{e} \mod 7 = \begin{bmatrix} 1 \cdot 2 + 2 \cdot 1 + 1 \\ 3 \cdot 2 + 4 \cdot 1 - 1 \\ 5 \cdot 2 + 6 \cdot 1 + 0 \end{bmatrix} \mod 7 = \begin{bmatrix} 6 \\ 3 \\ 4 \end{bmatrix}.$$

The task is to recover $\vec{s}$ and "small" $\vec{e}$ given A and $\vec{t}$.

This simple example illustrates how noise (represented by $\vec{e}$) is added to the system, complicating the recovery of $\vec{s}$.

Definition: Learning with Errors (LWE) Assumption We say the decisional $\mathsf{LWE}_{m,q,\psi}$ problem is quantum-hard if for all quantum polynomial-time distinguisher D, there is a negligible function ϵ such that for all $n \in \mathbb{N}$:

$$\Pr[A \leftarrow \mathbb{Z}_q^{m \times n}; \vec{s} \leftarrow \psi^n; \vec{e} \leftarrow \psi^m : D(A, A \cdot \vec{s} + \vec{e}) = 1]$$
$$- \Pr[A \leftarrow \mathbb{Z}_q^{m \times n}; \vec{t} \leftarrow \mathbb{Z}_q^m : D(A, \vec{t}) = 1] \le \epsilon(n),$$

where $m, q \in \mathbb{N}$ are functions of n, and ψ is an efficiently samplable distribution over $\mathbb{Z}_q$.

Discussion

Hardness Parameters. The hardness of LWE depends on the parameters m, q, ψ. For example: - When ψ is uniform over $\mathbb{Z}_q$, the two distributions are identical (but useless). - Notice that the above definition also samples $\vec{s}$ from ψ. - The two distributions differ in entropy:

$$(A, \vec{t}) \text{ consists of } mn + m \text{ uniform elements from } \mathbb{Z}_q,$$

but

$$(A, A \cdot \vec{s} + \vec{e}) \text{ is sampled from } mn \text{ uniform elements plus } (m + n) \text{ elements from } \psi.$$

Useful Parameters. A useful set of parameters is as follows: - $m(n) = \text{poly}(n) \ge n^2$: The smaller m, the harder LWE. - $q(n) = n^{\text{poly}(n)}$: q is large enough to allow encoding elements in poly(n) bits. - ψ: A distribution such that $\exists B = \text{poly}(n)$, ψ outputs $|a| \le B$ except with negligible probability. For example, ψ can be viewed as a uniform distribution. The larger B compared to q, the harder LWE.

Example: NIST Candidate CRYSTALS-Kyber

The CRYSTALS-Kyber cryptographic scheme uses LWE with the following parameters: - $n = 256k$, where $k = 3, 4, 5$. - $q = 3329$: It can be viewed as $O(n)$. - ψ: A binomial distribution with 3 or 5 fair trials, so that $|e| \le 5$

Corollary Suppose that $\gcd(q,2)=1$. If $(A, A\cdot\vec{s}+\vec{e}) \approx (A,\vec{t})$, then $(A, A\cdot\vec{s}+2\cdot\vec{e}) \approx (A,\vec{t})$.

Proof Idea: Since q and 2 are coprime, A and $2A$ are identically distributed (and $\vec{t}$ as well).

Explanation: The significance of $\gcd(d,q)=1$ lies in ensuring that modular arithmetic remains invertible under multiplication. For example: - If d and q are not coprime, the mapping could lose injectivity, potentially exposing information about $\vec{e}$. - When d and q are coprime, multiplying $\vec{e}$ by d does not destroy the randomness of the LWE problem's output distribution.

Example: Consider the following concrete setup:

$$q=7, \quad d=3, \quad \gcd(d,q)=1.$$

Let the matrix A, secret key $\vec{s}$, and error vector $\vec{e}$ be:

$$A=\begin{bmatrix}2 & 3\\ 4 & 5\end{bmatrix}, \quad \vec{s}=\begin{bmatrix}1\\2\end{bmatrix}, \quad \vec{e}=\begin{bmatrix}1\\-1\end{bmatrix}.$$

First, compute $A\cdot\vec{s}$ modulo q:

$$A\cdot\vec{s}=\begin{bmatrix}2 & 3\\ 4 & 5\end{bmatrix}\begin{bmatrix}1\\2\end{bmatrix}=\begin{bmatrix}8\\14\end{bmatrix}\equiv\begin{bmatrix}1\\0\end{bmatrix} \pmod 7.$$

Adding the error $\vec{e}$ gives:

$$A\cdot\vec{s}+\vec{e}=\begin{bmatrix}1\\0\end{bmatrix}+\begin{bmatrix}1\\-1\end{bmatrix}=\begin{bmatrix}2\\6\end{bmatrix}.$$

Now, multiply the error by $d=3$:

$$A\cdot\vec{s}+3\cdot\vec{e}=\begin{bmatrix}2\\6\end{bmatrix}+3\cdot\begin{bmatrix}1\\-1\end{bmatrix}=\begin{bmatrix}5\\3\end{bmatrix}.$$

The final result remains statistically indistinguishable due to the randomness preserved by $\gcd(d,q)=1$.

An Encryption Based on LWE

If we have the following decisional $\mathsf{LWE}_{m,q,\psi}$:

$$\{A, A\cdot\vec{s}+\vec{e}\} \approx \{A,\vec{t}\},$$

then, given $d\in\mathbb{N}$ such that $\gcd(d,q)=1$, we also have:

$$\{A, A\cdot\vec{s}+d\vec{e}\} \approx \{A,\vec{t}\},$$

by a simple reduction that multiplies the former indistinguishability by d (because A and $\vec{t}$ are both uniform over $\mathbb{Z}_q$).

Conclusion: This reduction demonstrates the flexibility of LWE-based constructions. By scaling the error vector $\vec{e}$ appropriately, we can adapt the encryption process to balance between computational efficiency and security, without violating the fundamental hardness assumptions.

Construction: Secret Key Encryption from LWE

Parameters: $m, q \in \mathbb{N}$ as a function of n, ψ a distribution over $\mathbb{Z}_q$.

Key Generation (Gen):

$$k := \vec{s} \leftarrow \psi^n$$

Encryption (Enc): For binary message $m \in \{0, 1\}$:

1. Sample $\vec{a} \leftarrow \mathbb{Z}_q^n$ and $e \leftarrow \phi$.
2. Output ciphertext:

$$c := (\vec{a}, t = \vec{a} \cdot \vec{s} + 2e + m),$$

where the arithmetic is performed in $\mathbb{Z}_q$.

Decryption (Dec): For ciphertext $c = (\vec{a}, t)$, output:

$$m' := (t - \vec{a} \cdot \vec{s} \mod 2).$$

Correctness: The decryption process computes:

$$t - \vec{a} \cdot \vec{s} = 2e + m \quad \Rightarrow \quad (t - \vec{a} \cdot \vec{s}) \mod 2 = m.$$

As long as the error e satisfies $\|e\| < q/4$, modular arithmetic will not overflow, ensuring decryption correctness.

Security: The (secret-key) CPA security can be proved by a reduction R such that whenever the adversary A of the encryption asks for an encryption, R takes the next row from its LWE input and adds m. This reduction leverages the indistinguishability of LWE distributions. Detailed proofs are provided in the references.

Additional Explanations: 1. **Parameter Selection**: - Proper parameter selection is critical for both security and performance: - q is typically chosen as a large prime or a power of 2 to ensure efficient modular arithmetic. - m must satisfy $m \gg n$ to ensure that the linear system $A \cdot \vec{s}$ is computationally hard to invert. - ψ, often modeled as a discrete Gaussian, ensures the randomness of the error vector e.

2. Correctness Analysis: - Correctness depends on the size of the noise e. The error e must remain small enough so that the term $2e$ does not wrap around modulo q. This ensures that:

$$t - \vec{a} \cdot \vec{s} = 2e + m,$$

and taking $\mod 2$ recovers m exactly.

3. Security Proof: - Security relies on the LWE assumption: that it is computationally hard to distinguish $(\vec{a}, t = \vec{a} \cdot \vec{s} + e)$ from a uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

Homomorphic Encryption

Definition: Homomorphic Encryption A (public or secret key) encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be homomorphic if the scheme provides efficient $(\mathsf{Add}, \mathsf{Mul})$ operations that satisfy the syntax and correctness below.

Addition: For any messages $m_0, m_1 \in \mathbb{Z}_2$,

$$\Pr_k \left[(c_i \leftarrow \mathsf{Enc}_k(m_i))_{i\in\{0,1\}} ; \mathsf{Dec}_k(\mathsf{Add}(c_0, c_1)) = m_0 + m_1 \right] = 1.$$

Multiplication: For any messages $m_0, m_1 \in \mathbb{Z}_2$,

$$\Pr_k \left[(c_i \leftarrow \mathsf{Enc}_k(m_i))_{i\in\{0,1\}} ; \mathsf{Dec}_k(\mathsf{Mul}(c_0, c_1)) = m_0 \cdot m_1 \right] = 1.$$

Multiply by Constant: For any messages $m_0, m_1 \in \mathbb{Z}_2$,

$$\Pr_k [c_0 \leftarrow \mathsf{Enc}_k(m_0); \mathsf{Dec}_k(\mathsf{Mul}(c_0, m_1)) = m_0 \cdot m_1] = 1.$$

Here, the message space and the arithmetic operations $(+, \cdot)$ are in $\mathbb{Z}_2$, but one may define similarly for other algebra.

Discussion: Bootstrapping in Fully Homomorphic Encryption Bootstrapping is a key technique enabling Fully Homomorphic Encryption (FHE) by addressing the noise growth inherent in repeated homomorphic operations.

Key Idea: When homomorphic operations (e.g., Add and Mul) are performed, the noise in the ciphertext increases. If the noise exceeds a certain threshold, decryption becomes impossible. Bootstrapping allows the ciphertext to be ”refreshed” by performing a homomorphic decryption and re-encryption, reducing the accumulated noise to a manageable level.

Steps in Bootstrapping: 1. Encrypt the secret key k_1 using another key k_2, producing an evaluation key $\mathsf{ek} = \mathsf{Enc}_{k_2}(k_1)$. 2. Perform a homomorphic evaluation of the decryption function using the noisy ciphertext c and the evaluation key ek:

$$c' = \mathsf{Eval}(f_{\mathsf{Dec},c}, \mathsf{ek}),$$

where $f_{\mathsf{Dec},c}(x)$ is the decryption circuit parameterized by c. 3. Output c' as the refreshed ciphertext.

Efficiency Concerns: While Gentry’s original bootstrapping implementation (2009) was computationally expensive due to deep decryption circuits, subsequent optimizations such as BGV and CKKS schemes have significantly improved efficiency.

Applications: Bootstrapping enables arbitrary-depth computations, which are essential for:

- Secure multi-party computation.
- Privacy-preserving machine learning.
- Homomorphic evaluation of complex functions (e.g., neural networks).

Historical Note: Bootstrapping was first proposed by Gentry in 2009 and remains a cornerstone of modern FHE research.

Definition: Homomorphic Encryption for Class $\mathcal{C}$ Let $\mathcal{C}$ be a class of circuits. We say the encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ is homomorphic for $\mathcal{C}$ if for any $C \in \mathcal{C}$, for any $m_1, ..., m_\ell \in \{0, 1\}$ where ℓ is the input size of C, let $k \leftarrow \mathsf{Gen}(1^n)$ be the encryption key,

$$\Pr_k \begin{bmatrix} (c_i \leftarrow \mathsf{Enc}_k(m_i))_{i \in [\ell]}; \\ \mathsf{Dec}_k(\mathsf{Eval}(C, c_1, ..., c_\ell)) = C(m_1, ..., m_\ell) \end{bmatrix} = 1.$$

Moreover, we require the output size of Eval to be *compact*, that is, bounded by $\ell' \cdot |\mathsf{Enc}_k(m_i)|$.

Additive Homomorphic Encryption Example The above secret-key encryption based on LWE has a direct homomorphic addition. To see why, consider two ciphertexts that are encrypted using the same key $\vec{s}$:

$$c_0 := (\vec{a}_0, \vec{a}_0 \cdot \vec{s} + 2e_0 + m_0), \quad c_1 := (\vec{a}_1, \vec{a}_1 \cdot \vec{s} + 2e_1 + m_1).$$

The homomorphic $\mathsf{Add}(c_0, c_1)$ is defined to be the coordinate-wise addition:

$$\begin{aligned} \mathsf{Add}(c_0, c_1) :=& (\vec{a}_0 + \vec{a}_1, (\vec{a}_0 \cdot \vec{s} + 2e_0 + m_0) + (\vec{a}_1 \cdot \vec{s} + 2e_1 + m_1)) \\ =& (\vec{a}', \vec{a}' \cdot \vec{s} + 2e' + (m_0 + m_1)), \end{aligned}$$

where $\vec{a}' = \vec{a}_0 + \vec{a}_1$ and $e' = e_0 + e_1$.

The multiplication by constant is also defined coordinate-wise as a multiplication by the plaintext m_1. Notice that the correctness holds as long as $e' \leq q/4$ before modulo q. Hence, we can perform $O(q/B)$ operations (of addition or multiplication by constant) and still obtain the correct decryption, where B and q are the LWE parameters.

Definition: Homomorphic Encryption for Class $\mathcal{C}$ Let $\mathcal{C}$ be a class of circuits. We say the encryption scheme $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ is homomorphic for $\mathcal{C}$ if for any $C \in \mathcal{C}$, for any $m_1, ..., m_\ell \in \{0, 1\}$ where ℓ is the input size of C, let $k \leftarrow \mathsf{Gen}(1^n)$ be the encryption key,

$$\Pr_k \begin{bmatrix} (c_i \leftarrow \mathsf{Enc}_k(m_i))_{i \in [\ell]}; \\ \mathsf{Dec}_k(\mathsf{Eval}(C, c_1, ..., c_\ell)) = C(m_1, ..., m_\ell) \end{bmatrix} = 1.$$

Moreover, we require the output size of Eval to be *compact*, that is, bounded by $\ell' \cdot |\mathsf{Enc}_k(m_i)|$.

Additive Homomorphic Encryption Example The above secret-key encryption based on LWE has a direct homomorphic addition. To see why, consider two ciphertexts that are encrypted using the same key $\vec{s}$:

$$c_0 := (\vec{a}_0, \vec{a}_0 \cdot \vec{s} + 2e_0 + m_0), \quad c_1 := (\vec{a}_1, \vec{a}_1 \cdot \vec{s} + 2e_1 + m_1).$$

The homomorphic $\mathsf{Add}(c_0, c_1)$ is defined to be the coordinate-wise addition:

$$\begin{aligned}\mathsf{Add}(c_0, c_1) :=& (\vec{a}_0 + \vec{a}_1, (\vec{a}_0 \cdot \vec{s} + 2e_0 + m_0) + (\vec{a}_1 \cdot \vec{s} + 2e_1 + m_1)) \\ =& (\vec{a}', \vec{a}' \cdot \vec{s} + 2e' + (m_0 + m_1)),\end{aligned}$$

where $\vec{a}' = \vec{a}_0 + \vec{a}_1$ and $e' = e_0 + e_1$.

The multiplication by constant is also defined coordinate-wise as a multiplication by the plaintext m_1. Notice that the correctness holds as long as $e' \leq q/4$ before modulo q. Hence, we can perform $O(q/B)$ operations (of addition or multiplication by constant) and still obtain the correct decryption, where B and q are the LWE parameters.

Fuzzy Correctness For some negligible function $\epsilon(n)$, we allow the decryption correctness to be probabilistic:

$$\Pr_k [\mathsf{Dec}_k(\mathsf{Add}(c_0, c_1)) = m_0 + m_1] \geq 1 - \epsilon(n),$$

and similarly for multiplication:

$$\Pr_k [\mathsf{Dec}_k(\mathsf{Mul}(c_0, c_1)) = m_0 \cdot m_1] \geq 1 - \epsilon(n).$$

This condition is particularly relevant when noise accumulates during repeated homomorphic operations. Managing this noise is crucial for ensuring decryption correctness over multiple operations, as seen in modern schemes such as BGV and CKKS.

References

- Regev, O. (2005). *On Lattices, Learning with Errors, Random Linear Codes, and Cryptography*. Available at: `https://arxiv.org/pdf/2401.03703`. (Proposed LWE, following the "learning from parity with error" problem.)
- Regev, O. (2010). *Learning with Errors: A Survey*. Available at: `https://cims.nyu.edu/~regev/papers/lwesurvey.pdf`. (Provides a comprehensive survey on LWE.)
- Chen, Z. (2024). *A Quantum Algorithm for Solving LWE for Large Parameter Ranges in Polynomial Time*. Available at: `https://eprint.iacr.org/2024/555.pdf`. (Potential breakthrough in solving LWE, pending review.)
- Brakerski, Z., and Vaikuntanathan, V. (2011). *Efficient Fully Homomorphic Encryption from (Standard) LWE*. In *Proceedings of FOCS 2011*. Available at: `https://eprint.iacr.org/2011/344.pdf`.
- Brakerski, Z., Gentry, C., and Vaikuntanathan, V. (2012). *(Leveled) Fully Homomorphic Encryption without Bootstrapping*. In *Proceedings of ITCS 2012*. Available at: `https://people.csail.mit.edu/vinodv/6892-Fall2013/BGV.pdf`.

- Gentry, C. (2009). *Fully Homomorphic Encryption Using Ideal Lattices*. In *Proceedings of STOC 2009*. Available at: `https://dl.acm.org/doi/10.1145/1536414.1536440`.
- Gentry, C., Halevi, S., and Smart, N. P. (2012). *Homomorphic Evaluation of the AES Circuit*. In *Proceedings of CRYPTO 2012*. Available at: `https://eprint.iacr.org/2012/099.pdf`.
- Lin, H., Mook, A., and Wichs, D. (2023). *Doubly Efficient Private Information Retrieval and Fully Homomorphic RAM Computation from Ring LWE*. In *Proceedings of STOC 2023*. Available at: `https://eprint.iacr.org/2022/1703.pdf`.
- Lindner, R., and Peikert, C. (2011). *Better Key Sizes (and Attacks) for LWE-Based Encryption*. In *Proceedings of CT-RSA 2011*. Available at: `https://eprint.iacr.org/2010/613.pdf`.
- Barak, B. (Princeton University). *Lecture Notes on Homomorphic Encryption*. Available at: `https://www.cs.princeton.edu/courses/archive/spring10/cos433/lec21new.pdf`.
- Rothblum, T. (2011). *Homomorphic Encryption: From Private-Key to Public-Key*. In *Proceedings of TCC 2011*. Available at: `https://www.iacr.org/archive/tcc2011/65970216/65970216.pdf`.