CS7111: Topics in Cryptography	July 31, August 2, 7, 2019
Lecture 1-3: Learning With Errors	
Lecturer: *Shweta Agrawal*	Scribe: *Tapas Pal*

1 Introduction

What if we want to process a particular file stored on a server without letting the server know any information on the file we are looking for? If we use plain public-key encryption (PKE) to encrypt those files and keep on the server, then the server has to know the secret-key which we used to encrypt our files. This is clearly insecure, since the server can access all the files which we do not want. Else, you need to download the encrypted documents that we need every time, decrypt to process them and re-encrypt to keep them back in the server. This is cumbersome and we want to avoid this. Is there a way that we can encrypt the files and store them into the server such that the server can still process any query without knowing the secret-key and send back the result we intended to get? The cryptographic primitive of "fully homomorphic" encryption (FHE) makes this possible. FHE lets us compute any arbitrary function (of our choice) on the *encrypted* data so that we would be able to use our secret key to decrypt the "encrypted output" given by the server.

In this course, we will learn some exciting techniques in modern cryptography pertaining to computing on encrypted data and outsourcing computation. We will be introduced to some beautiful primitives like FHE, Functional Encryption (FE), Indistinguishability Obfuscation (iO) and Witness Encryption (WE). To this end, we plan to cover some basic tools and techniques in this lecture for understanding FHE.

- Introduction: from PKE to FHE
- Textbook RSA (that satisfies homomorphic property with respect to only multiplication)
- An average case hard problem: Learning With Errors (LWE)
- Private-key encryption from LWE
- Public-key encryption from LWE

Notation. For an integer $n \in \mathbb{N}$, the notation $[n]$ represents the set $\{1, \ldots, n\}$. We denote by $x \leftarrow \mathcal{D}$ the process of sampling a value x according to the distribution of $\mathcal{D}$. We consider $x \leftarrow S$ as the process of random sampling a value x according to the uniform distribution over a finite set S. We abbreviate probabilistic polynomial time as PPT. Any cryptographic scheme relies implicitly on the notion of security parameter. This parameter quantifies any PPT adversary's probability of breaking the cryptographic protocol. We will denote security parameter by λ. We denote by $\mathsf{negl} = \{\mathsf{negl}_\lambda\}_{\lambda \in \mathbb{N}}$ as the class of functions over $\mathbb{N}$ which *degrades faster than the inverse of any polynomial* in λ. In particular, $f \in \mathsf{negl}_\lambda$ iff $\forall c \in \mathbb{N}, \exists \lambda_0 \in \mathbb{N}$ such that $\forall \lambda \geq \lambda_0, f(\lambda) \leq \lambda^{-c}$. Such a function f is called a *negligible* function.

2 Public-key Encryption

Definition 1. (Public-key encryption). A *public-key encryption* scheme for a message space $\mathcal{M}$ is given as a tuple of PPT algorithms $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ as follows:

- $\mathsf{PKE.Gen}(1^\lambda) \to (\mathsf{SK}, \mathsf{PK})$: This is a randomized key generation algorithm which takes the security parameter λ as input and outputs a public key PK and a secret key SK.
- $\mathsf{PKE.Enc}(\mathsf{PK}, m) \to c$: This a randomized algorithm that takes the public key PK and a message $m \in \mathcal{M}$ and outputs a ciphertext c encrypting m.
- $\mathsf{PKE.Dec}(\mathsf{SK}, c) \in \mathcal{M} \cup \{\perp\}$: This is a *deterministic* algorithm which takes the secret key SK and a ciphertext c as input. It outputs a message $m' \in \mathcal{M}$. (In some cases, it can also output $\perp$ indicating a decryption failure.)

Correctness: $\forall \lambda \in \mathbb{N}, m \in \mathcal{M}$, we have

$$\Pr\left[\mathsf{PKE.Dec}(\mathsf{SK}, c) = m : (\mathsf{SK}, \mathsf{PK}) \leftarrow \mathsf{PKE.Gen}(1^\lambda),\ c \leftarrow \mathsf{PKE.Enc}(\mathsf{PK}, m)\right] = 1$$

over the random coin tosses of the algorithms $\mathsf{PKE.Gen}, \mathsf{PKE.Enc}$.

Definition 2. (IND-CPA). We say that a public-key encryption scheme $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ satisfies indistinguishability under chosen plaintext attacks (IND-CPA), if $\forall \lambda \in \mathbb{N}$ and every PPT adversary $\mathcal{A}$, it holds that

$$\left|\Pr\left[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{PKE}}(1^\lambda, 0) = 1\right] - \Pr\left[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{PKE}}(1^\lambda, 1) = 1\right]\right| \leq \mu(\lambda)$$

where the experiment $\mathrm{Expt}_{\mathcal{A}}^{\mathsf{PKE}}(1^\lambda, b)$ is defined in Figure 1 where $b \in \{0, 1\}$ and μ is a negligible function of λ.

1. The challenger obtains $\mathsf{PKE.Gen}(1^\lambda) \to (\mathsf{SK}, \mathsf{PK})$ and sends PK to the adversary $\mathcal{A}$.
2. $\mathcal{A}$ selects $(m_0, m_1) \in \mathcal{M}^2$ such that $|m_0| = |m_1|$ and sends (m_0, m_1) to the challenger.
3. The challenger sends $\mathsf{PKE.Enc}(\mathsf{PK}, m_b) \to c_b$ to $\mathcal{A}$.
4. The adversary $\mathcal{A}$ outputs a guess b' for b.

Figure 1: $\mathbf{Expt}_{\mathcal{A}}^{\mathsf{PKE}}(1^\lambda, b)$: IND-CPA security game for public-key encryption

Note: For homomorphic encryption there is an additional (possibly PPT) algorithm Eval which takes as input the public key PK (and sometimes a separate publicly known evaluation key, Evk), a boolean function $f : \{0, 1\}^l \to \{0, 1\}$ and ciphertexts $c_1, \ldots, c_l$. It returns a new ciphertext c^* as $\mathsf{Eval}(\mathsf{PK}, \mathsf{Evk}, f, c_1, \ldots, c_l) \to c^*$. Note that, Eval should be a public key algorithm.

Definition 3. ($\mathcal{C}$-homomorphic encryption). *An encryption scheme* $\mathsf{HE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *(with an evaluation algorithm* $\mathsf{HE.Eval}$ *as defined above) is called a* $\mathcal{C}$*-homomorphic encryption scheme for a class of functions* $\mathcal{C}$*, a message space* $\mathcal{M}$*, if* $\forall m_1, \ldots, m_l \in \mathcal{M}, \forall f \in \mathcal{C}$*, it holds that*

$$\Pr\left[\mathsf{Dec}(\mathsf{SK}, c^*) \neq f(m_1, \ldots, m_l) : \begin{array}{r} \mathsf{HE.Gen}(1^\lambda) \to (\mathsf{SK}, \mathsf{PK}, \mathsf{Evk}), \\ \mathsf{HE.Enc}(\mathsf{PK}, m_i) \to c_i, \forall i \in [l], \\ \mathsf{Eval}(\mathsf{PK}, \mathsf{Evk}, f, c_1, \ldots, c_l) \to c^* \end{array}\right] \in \mathsf{negl}_\lambda.$$

over the random coin tosses of $\mathsf{HE.Gen}, \mathsf{HE.Enc}$.

If $\mathcal{C}$ is a class of all polynomial sized circuits ($\mathsf{P/poly}$), then we say that the scheme is *fully homomorphic*. The security requirement is the same as the IND-CPA security of the PKE.

2.1 Textbook RSA: Multiplicative homomorphic

Let us first review the textbook RSA scheme. Then we show that the RSA encryption satisfies the homomorphic property with respect to multiplication.

- $\mathsf{Gen}(1^\lambda) \to (\mathsf{PK}, \mathsf{SK})$: Choose two large primes p, q and set $N = pq$. Sample $e \leftarrow \mathbb{Z}_N^*$ such that $\gcd(e, \phi(N)) = 1$. Output the public-key as $\mathsf{PK} = (e, N)$ and secret-key as $\mathsf{SK} = e^{-1} \bmod \phi(N)$. Here, ϕ is the Euler's totient function.
- $\mathsf{Enc}(\mathsf{PK}, m) \to c$: For a message $m \in \mathbb{Z}_N$, compute and return $c = m^e \bmod N$ as a ciphertext encoding m.
- $\mathsf{Dec}(\mathsf{SK}, c) = m'$: Compute and output message $m' = c^{\mathsf{SK}} \bmod N$.

Homomorphic property of RSA: Let c_1 and c_2 be the RSA encryption of two messages m_1 and m_2 respectively. We show that $c_1 \cdot c_2$ produces the RSA encryption of the message $m_1 \cdot m_2 \bmod N$:

$$\begin{aligned}\mathsf{Enc}(\mathsf{PK}, m_1) \cdot \mathsf{Enc}(\mathsf{PK}, m_2) &= m_1^e \cdot m_2^e \bmod N \\ &= (m_1 \cdot m_2)^e \bmod N \\ &= \mathsf{Enc}(\mathsf{PK}, m_1 \cdot m_2)\end{aligned}$$

Therefore, RSA encryption naturally satisfies the multiplicative homomorphic property. Similarly, one can show that the classical El-Gamal encryption scheme is additively homomorphic. Note that RSA is not IND-CPA secure as the encryption algorithm is deterministic. There are some versions of RSA where randomization techniques (for example OAEP) are employed in the encryption to make it an IND-CPA secure scheme.

Fact: All known IND-CPA versions of RSA do not satisfy homomorphism.
Hence we can note down the following problem which is open for a long time.

Prob. Construct IND-CPA multiplicative homomorphic RSA.

Therefore, it is non-trivial to construct a fully homomorphic encryption scheme from known PKEs. We discuss an average case hard problem named *Learning with Errors* (LWE) in the next section and later construct a homomorphic encryption scheme from LWE.

3 Learning With Errors

The learning with errors problem was introduced by Oded Regev [Reg09] for which he won the Gödel prize in 2018. Over this decade we have seen interesting applications of LWE in realizing many cryptographic primitives. Another advantage is that no quantum attack against LWE assumption is known till date (unlike the other problems–factoring and discrete logarithm which supplies the hardness underlying the current cryptographic schemes like RSA and ElGamal).

An learning with errors instance $\mathsf{LWE}_{n,q,\chi}$ is parameterized by an integer $n \in \mathbb{N}$, a prime modulus q and a probability distribution χ over $\mathbb{Z}_q$. For a *fixed* $\boldsymbol{s} \in \mathbb{Z}_q^n$, we define the LWE distribution and a random distribution as follows:

– *LWE distribution*: sample $\boldsymbol{a} \leftarrow \mathbb{Z}_q^n, e \leftarrow \chi$ and return $(\boldsymbol{a}, \langle \boldsymbol{a}, \boldsymbol{s} \rangle + e)$

– *Random distribution*: chose $\boldsymbol{a} \leftarrow \mathbb{Z}_q^n, r \leftarrow \mathbb{Z}_q$ and return $(\boldsymbol{a}, r)$

For $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n$, let $\mathsf{LWE}_{\boldsymbol{s}}$ be the oracle which outputs samples from the LWE distribution and Random be the oracle that outputs samples according to the random distribution defined above. We define two types of LWE problems below.

- Search Problem: The *search*-$\mathsf{LWE}_{n,q,\chi}$ is the problem to find $\boldsymbol{s}$, given access to the oracle $\mathsf{LWE}_{\boldsymbol{s}}$.
- Decision Problem: The *decision*-$\mathsf{LWE}_{n,q,\chi}$ problem is to distinguish between $\mathsf{LWE}_{\boldsymbol{s}}$ and Random.

The *search*-$\mathsf{LWE}_{n,q,\chi}$ assumption is that the *search*-$\mathsf{LWE}_{n,q,\chi}$ problem is hard and mathematically we express this as–for any PPT algorithm $\mathcal{A}$, it holds that

$$\Pr_{\boldsymbol{s}\leftarrow\mathbb{Z}_q^n}\left[\mathcal{A}^{\mathsf{LWE}_{\boldsymbol{s}}}(1^n) = \boldsymbol{s}\right] \in \mathsf{negl}_n$$

Similarly, the *decision*-$\mathsf{LWE}_{n,q,\chi}$ assumption is that the *decision*-$\mathsf{LWE}_{n,q,\chi}$ problem is hard and mathematically we represent this statement as–for any PPT distinguisher $\mathcal{D}$, it holds that

$$\left|\Pr\left[\mathcal{D}^{\mathsf{LWE}_{\boldsymbol{s}}}(1^n) = 1\right] - \Pr\left[\mathcal{D}^{\mathsf{Random}}(1^n) = 1\right]\right| \in \mathsf{negl}_n$$

3.1 LWE Assumption with Fixed Number of Samples

We can redefine the above LWE assumption with an additional parameter $m \in \mathbb{N}$ which represents the number of LWE or random samples given to the adversary (instead of the access to oracles $\mathsf{LWE}_{\boldsymbol{s}}$ or Rand) to solve the above problems. The m LWE samples received from $\mathsf{LWE}_{\boldsymbol{s}}$ (resp., Random) are written as $(\boldsymbol{A}, \boldsymbol{A}^T\boldsymbol{s} + \boldsymbol{e})$ (resp., $(\boldsymbol{A}, \boldsymbol{A}\boldsymbol{r})$) where $\boldsymbol{A} \leftarrow \mathbb{Z}_q^{n\times m}, \boldsymbol{e} \leftarrow \chi^m, \boldsymbol{r} \leftarrow \mathbb{Z}_q^m$. Therefore, we restate the above assumptions as follows:

Definition 4. (*search*-$\mathsf{LWE}_{n,m,q,\chi}$ *assumption*). For any PPT algorithm $\mathcal{A}$, it holds that

$$\Pr_{\substack{\boldsymbol{s}\leftarrow\mathbb{Z}_q^n\\ \boldsymbol{A}\leftarrow\mathbb{Z}_q^{n\times m}\\ \boldsymbol{e}\leftarrow\chi^m}}\left[\mathcal{A}(1^n, (\boldsymbol{A}, \boldsymbol{A}^T\boldsymbol{s} + \boldsymbol{e})) = \boldsymbol{s}\right] \in \mathsf{negl}_n$$

Definition 5. (*decision*-$\mathsf{LWE}_{n,m,q,\chi}$ *assumption*). For any PPT distinguisher $\mathcal{D}$ and $\boldsymbol{r} \leftarrow \mathbb{Z}_q^m$, it holds that

$$\left|\Pr_{\substack{\boldsymbol{s}\leftarrow\mathbb{Z}_q^n\\ \boldsymbol{A}\leftarrow\mathbb{Z}_q^{n\times m}\\ \boldsymbol{e}\leftarrow\chi^m}}\left[\mathcal{D}(1^n, (\boldsymbol{A}, \boldsymbol{A}^T\boldsymbol{s} + \boldsymbol{e})) = 1\right] - \Pr_{\substack{\boldsymbol{A}\leftarrow\mathbb{Z}_q^{n\times m}\\ \boldsymbol{r}\leftarrow\mathbb{Z}_q^m}}\left[\mathcal{D}(1^n, (\boldsymbol{A}, \boldsymbol{r})) = 1\right]\right| \in \mathsf{negl}_n$$

One can easily see that if the *decision*-$\mathsf{LWE}_{n,m,q,\chi}$ assumption holds, then *search*-$\mathsf{LWE}_{n,m,q,\chi}$ holds too. Interestingly, the opposite direction is also true with some change in parameter m. We note that the *search*-LWE is not suitable for cryptographic assumption as it allows an adversary to learn the secret $\boldsymbol{s}$ partially (for example, the half part of $\boldsymbol{s}$). In cryptographic construction, we want that the adversary should not get any information about the secret and the *decision*-LWE is thus preferable.

We also define another version of LWE problem where the secret $\boldsymbol{s}$ and the error $\boldsymbol{e}$ are both sampled from the same distribution χ. This version of LWE is called *short-secret*-LWE or ss-LWE.

Claim 1. *The* ss-LWE *problem is as hard as normal* LWE *problem.*

Proof. Assume we have an oracle $\mathcal{O}_{\mathsf{LWE}}$ that solves (normal) LWE problem. Given $(\boldsymbol{A}, \boldsymbol{b} = \boldsymbol{A}^T\boldsymbol{s} + \boldsymbol{e})$ for $\boldsymbol{A} \leftarrow \mathbb{Z}_q^{n\times m}$, $\boldsymbol{s} \leftarrow \chi^n$ and $\boldsymbol{e} \leftarrow \chi^m$; we need to find $\boldsymbol{b}$ using the above oracle. We sample $\boldsymbol{s}' \leftarrow \mathbb{Z}_q^n$ and define $\boldsymbol{b}' = \boldsymbol{b} + \boldsymbol{A}^T\boldsymbol{s}' = \boldsymbol{A}^T(\boldsymbol{s} + \boldsymbol{s}') + \boldsymbol{e}$. Therefore, $(\boldsymbol{A}, \boldsymbol{b}')$ becomes a normal LWE sample as $(\boldsymbol{s} + \boldsymbol{s}')$ is uniformly distributed over $\mathbb{Z}_q^n$. Now, $\mathcal{O}_{\mathsf{LWE}}(\boldsymbol{A}, \boldsymbol{b}')$ outputs $\boldsymbol{t} \in \mathbb{Z}_q^n$ and hence we output $\boldsymbol{t} - \boldsymbol{s}'$ as the solution for the $\mathsf{ss\text{-}LWE}$ instance $(\boldsymbol{A}, \boldsymbol{b})$.

Next, we assume that the oracle $\mathcal{O}_{\mathsf{ss\text{-}LWE}}$ can solve an $\mathsf{ss\text{-}LWE}$ instance. Given a (normal) LWE instance $(\boldsymbol{A}, \boldsymbol{b} = \boldsymbol{A}^T\boldsymbol{s} + \boldsymbol{e})$ for $\boldsymbol{A} \leftarrow \mathbb{Z}_q^{n\times m}$, $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n$ and $\boldsymbol{e} \leftarrow \chi^m$; we need to find $\boldsymbol{b}$ using the above oracle. We rewrite the instance as

$$\boldsymbol{A}^T = \begin{pmatrix}\boldsymbol{A}_1\\ \boldsymbol{A}_2\end{pmatrix}, \quad \boldsymbol{b} = \begin{pmatrix}\boldsymbol{A}_1\\ \boldsymbol{A}_2\end{pmatrix}\boldsymbol{s} + \begin{pmatrix}\boldsymbol{e}_1\\ \boldsymbol{e}_2\end{pmatrix} = \begin{pmatrix}\boldsymbol{b}_1\\ \boldsymbol{b}_2\end{pmatrix}$$

where $\boldsymbol{A}_1 \in \mathbb{Z}_q^{n\times n}$, $\boldsymbol{A}_2 \in \mathbb{Z}_q^{(m-n)\times n}$, $\boldsymbol{b}_1 \in \mathbb{Z}_q^n$ and $\boldsymbol{b}_2 \in \mathbb{Z}_q^{m-n}$. Since $\boldsymbol{A}$ is uniform over $\mathbb{Z}_q^{n\times m}$, the matrix $\boldsymbol{A}_1$ is invertible with overwhelming probability. Now, $\boldsymbol{b}_1 = \boldsymbol{A}_1\boldsymbol{s} + \boldsymbol{e}_1$ implies that we can implicitly set $\boldsymbol{s} = \boldsymbol{A}_1^{-1}(\boldsymbol{b}_1 - \boldsymbol{e}_1)$. We rewrite $\boldsymbol{b}_2$ as

$$\begin{aligned}\boldsymbol{A}_2\boldsymbol{s} + \boldsymbol{e}_2 &= \boldsymbol{A}_2(\boldsymbol{A}_1^{-1}(\boldsymbol{b}_1 - \boldsymbol{e}_1)) + \boldsymbol{e}_2\\ &= \boldsymbol{A}_2\boldsymbol{A}_1^{-1}\boldsymbol{b}_1 - \boldsymbol{A}_2\boldsymbol{A}_1^{-1}\boldsymbol{e}_1 + \boldsymbol{e}_2\end{aligned}$$

Therefore, $\boldsymbol{A}_2\boldsymbol{A}_1^{-1}\boldsymbol{b}_1 - \boldsymbol{b}_2 = \boldsymbol{A}_2\boldsymbol{A}_1^{-1}\boldsymbol{e}_1 - \boldsymbol{e}_2$ which implies that $(\boldsymbol{A}_2\boldsymbol{A}_1^{-1}, \boldsymbol{A}_2\boldsymbol{A}_1^{-1}\boldsymbol{b}_1 - \boldsymbol{b}_2)$ is a fresh $\mathsf{ss\text{-}LWE}$ instance (as $\boldsymbol{A}_2\boldsymbol{A}_1^{-1}$ is uniform over $\mathbb{Z}_q^{(m-n)\times n}$ since $\boldsymbol{A}^T$ is itself uniformly chosen from $\mathbb{Z}_q^{m\times n}$ and $\boldsymbol{e}_1 \leftarrow \chi^n$) and hence it can be fed to $\mathcal{O}_{\mathsf{ss\text{-}LWE}}$ to get $\boldsymbol{e}_1$. Now, we can derive $\boldsymbol{s}$ from the equation $\boldsymbol{s} = \boldsymbol{A}_1^{-1}(\boldsymbol{b}_1 - \boldsymbol{e}_1)$. □

4 Encryption Schemes from LWE

4.1 Private-key Encryption Scheme from LWE

In this subsection we describe a private-key encryption scheme $\mathsf{SKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ from LWE assumption. More precisely, we consider *decision*-$\mathsf{LWE}_{n,q,\chi}$ assumption where the error distribution χ is such that $\|\boldsymbol{e}\| \le q/4$ holds with high probability for $\boldsymbol{e} \leftarrow \chi$.

- $\mathsf{SKE.Gen}(1^n)$: Sample and output $\mathsf{SK} = \boldsymbol{s} \leftarrow \mathbb{Z}_q^n$.
- $\mathsf{SKE.Enc}(\mathsf{SK}, m)$: On input a secret-key $\mathsf{SK} = \boldsymbol{s}$ and a message $m \in \{0,1\}$, the encryption algorithm samples $\boldsymbol{a} \leftarrow \mathbb{Z}_q^n, e \leftarrow \chi$ and output $c = (\boldsymbol{a}, b = (\langle \boldsymbol{a}, \boldsymbol{s}\rangle + e + m\lfloor\frac{q}{2}\rfloor) \bmod q)$.
- $\mathsf{SKE.Dec}(\mathsf{SK}, c)$: On input a secret-key $\mathsf{SK} = \boldsymbol{s}$ and a ciphertext $c = (\boldsymbol{a}, b)$, the decryption algorithm first computes $u = b - \langle \boldsymbol{a}, \boldsymbol{s}\rangle$ and returns 0 if $|u| \le q/4$; otherwise 1.

Correctness. Note that $u = b - \langle \boldsymbol{a}, \boldsymbol{s}\rangle = e + m\lfloor\frac{q}{2}\rfloor$ implies $|u| \le q/4$ with high probability if $m = 0$ by the property of χ and $|u| > q/4$ if $m = 1$.

Security. By *decision*-$\mathsf{LWE}_{n,q,\chi}$ assumption, $(\boldsymbol{a}, \langle \boldsymbol{a}, \boldsymbol{s}\rangle + e \bmod q)$ is computationally indistinguishable from a random vector $(\boldsymbol{a}, r) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$. After adding $m\lfloor\frac{q}{2}\rfloor$ to the second component the LWE sample remains indistinguishable from a random vector $(\boldsymbol{a}, r) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$. Therefore, no PPT adversary can distinguish between the encryption of $m = 0$ and $m = 1$.

4.2 Public-key Encryption from LWE

We describe a public-key encryption scheme $\mathsf{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ from LWE assumption. For the correctness of the scheme we need to chose the error distribution χ such that $\|e\| \le \sqrt{q/4(2n+1)}$ for $e \leftarrow \chi$.

- $\mathsf{PKE.Gen}(1^n)$: Sample $\boldsymbol{A} \leftarrow \mathbb{Z}_q^{n\times n}, \boldsymbol{s}, \boldsymbol{e} \leftarrow \chi^n$ and output $\mathsf{SK} = \boldsymbol{s}$ and $\mathsf{PK} = (\boldsymbol{A}, \boldsymbol{y} = \boldsymbol{A}^T\boldsymbol{s} + \boldsymbol{e}) \in \mathbb{Z}_q^{n\times n} \times \mathbb{Z}_q^n$.
- $\mathsf{PKE.Enc}(\mathsf{PK}, m)$: On input a public-key $\mathsf{PK} = (\boldsymbol{A}, \boldsymbol{y})$ and a message $m \in \{0,1\}$, the encryption algorithm samples $\boldsymbol{r}, \boldsymbol{x} \leftarrow \chi^n, x' \leftarrow \chi$ and output $c = (\boldsymbol{c}_1, c_2) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, where $\boldsymbol{c}_1 = \boldsymbol{A}\boldsymbol{r} + \boldsymbol{x}, c_2 = \langle \boldsymbol{y}, \boldsymbol{r}\rangle + x' + m\lfloor \frac{q}{2} \rfloor$).
- $\mathsf{PKE.Dec}(\mathsf{SK}, c)$: On input a secret-key $\mathsf{SK} = \boldsymbol{s}$ and a ciphertext $c = (\boldsymbol{c}_1, c_2)$, compute $u = c_2 - \langle \boldsymbol{c}_1, \boldsymbol{s}\rangle$ and return 0, if $|u| \le q/4$; otherwise return 1.

Correctness. To re-compute u, we first note that

$$\begin{aligned}\langle \boldsymbol{c}_1, \boldsymbol{s}\rangle &= \boldsymbol{s}^T(\boldsymbol{A}r + \boldsymbol{x}) \\ &= (\boldsymbol{A}^T\boldsymbol{s})^T\boldsymbol{r} - \boldsymbol{s}^T\boldsymbol{x} \\ &= \langle \boldsymbol{A}^T\boldsymbol{s}, \boldsymbol{r}\rangle - \langle \boldsymbol{s}, \boldsymbol{x}\rangle \\ &= (\langle \boldsymbol{y}, \boldsymbol{r}\rangle - \langle \boldsymbol{e}, \boldsymbol{r}\rangle) - \langle \boldsymbol{s}, \boldsymbol{x}\rangle\end{aligned}$$

and therefore the value of $u = c_2 - \langle \boldsymbol{c}_1, \boldsymbol{s}\rangle$ is given by $\langle \boldsymbol{e}, \boldsymbol{r}\rangle - \langle \boldsymbol{s}, \boldsymbol{x}\rangle + x' + m\lfloor \frac{q}{2} \rfloor$. Due to the choice of χ we have $|\langle \boldsymbol{e}, \boldsymbol{r}\rangle - \langle \boldsymbol{s}, \boldsymbol{x}\rangle + x'| \le \frac{nq}{4(2n+1)} + \frac{nq}{4(2n+1)} + \sqrt{\frac{q}{4(2n+1)}} < \frac{q}{4}$. Hence, the correctness follows.

Security. We show that the above scheme is IND-CPA secure under the harness of $\mathsf{LWE}_{n,2(n+1),q,\chi}$ assumption. We proceed by the following hybrid experiments.

Hybrid 0. This is the same experiment as $\mathrm{Expt}_{\mathcal{A}}^{\mathsf{PKE}}(1^n, 0)$ (defined in Figure 1) where the challenger encrypts with respect to bit $b = 0$.

Hybrid 1. In this experiment the adversary receives a "fake" public-key defined as $\widetilde{\mathsf{PK}} = (\boldsymbol{A}, \boldsymbol{y}) \leftarrow \mathbb{Z}_q^{n\times n} \times \mathbb{Z}_q^n$ and ciphertexts are computed using $\widetilde{\mathsf{PK}}$.

The indistinguishability between Hybrid 0 and Hybrid 1 is preserved by the hardness of $\mathsf{ssLWE}_{n,n,q,\chi}$ assumption, and therefore by $\mathsf{LWE}_{n,2n,q,\chi}$ assumption.

Hybrid 2. Compute the challenge ciphertext as

$$\widetilde{\mathsf{Enc}}(\widetilde{\mathsf{PK}}, 0) = (\boldsymbol{a}, b)$$

where $(\boldsymbol{a}, b) \leftarrow \mathbb{Z}_q^n \times \mathbb{Z}_q$. Note that the challenge ciphertext in Hybrid 1 is actually $(n+1)$ ssLWE samples:

$$(\boldsymbol{A}\boldsymbol{r} + \boldsymbol{x}, \langle \boldsymbol{y}, \boldsymbol{r}\rangle + x')$$

where $(\boldsymbol{A}, \boldsymbol{y}) \leftarrow \mathbb{Z}_q^{n\times n} \times \mathbb{Z}_q^n, \boldsymbol{r}, \boldsymbol{x} \leftarrow \chi^n$, and $x' \leftarrow \chi$. Therefore, by $\mathsf{ssLWE}_{n,n+1,q,\chi}$ or by $\mathsf{LWE}_{n,2n+1,q,\chi}$ assumption Hybrid 2 is computationally indistinguishable from Hybrid 1.

Hybrid 3. Compute the challenge ciphertext as

$$\widetilde{\mathsf{Enc}}(\widetilde{\mathsf{PK}}, 1) = (\boldsymbol{a}, b + \lfloor \tfrac{q}{2} \rfloor \bmod q)$$

where $(\boldsymbol{a}, b) \leftarrow \mathbb{Z}_q^n \times \mathbb{Z}_q$. Since b is chosen uniformly at random from $\mathbb{Z}_q$, Hybrid 2 and Hybrid 3 are indistinguishable from the adversary's view.

Hybrid 4. We send the challenge ciphertext as

$$\mathsf{PKE.Enc}(\widetilde{\mathsf{PK}}, 1) = (\boldsymbol{A}\boldsymbol{r} + \boldsymbol{x}, \langle \boldsymbol{y}, \boldsymbol{r}\rangle + x' + \lfloor \tfrac{q}{2} \rfloor) \in \mathbb{Z}_q^n \times \mathbb{Z}_q.$$

Observe that Hybrid 3 and Hybrid 4 are computationally indistinguishable due to $\mathsf{LWE}_{n,2n+1,q,\chi}$ assumption (as in Hybrid 2).

Hybrid 5. This is the same experiment as $\mathrm{Expt}^{\mathsf{PKE}}_{\mathcal{A}}(1^n, 1)$ (defined in Figure 1) where the challenger encrypts with respect to bit $b = 1$. Hybrid 5 is indistinguishable from Hybrid 4 by $\mathsf{LWE}_{n,2n,q,\chi}$ assumption.

Therefore, we conclude by $\mathsf{LWE}_{n,2(n+1),q,\chi}$ assumption that

$$\left|\mathsf{Pr}\left[\mathsf{Expt}^{\mathsf{PKE}}_{\mathcal{A}}(1^n, 0) = 1\right] - \mathsf{Pr}\left[\mathsf{Expt}^{\mathsf{PKE}}_{\mathcal{A}}(1^n, 1) = 1\right]\right| \in \mathsf{negl}_n$$

References

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6):34:1–34:40, 2009.