CS 355 – Topics in Cryptography May 2, 2018

Lecture 10: Inhomogeneous SIS and the LWE Problem

Instructors: Henry Corrigan-Gibbs, Sam Kim, David J. Wu

1 Review

Last lecture, we discussed the SIS problem and leftover hash lemma.

$\mathsf{SIS}(n, m, q, B)$: Let $n, m, q, B \in \mathbb{N}$ be positive integers. For a given adversary $\mathcal{A}$, we define the following experiment:

- The challenger samples $\mathbf{A} \xleftarrow{R} \mathbb{Z}_q^{n \times m}$, and gives $\mathbf{A}$ to the adversary $\mathcal{A}$.
- The adversary $\mathcal{A}$ outputs some *non-zero* vector $\mathbf{x} \in \mathbb{Z}_q^m$.

We define $\mathcal{A}$'s advantage in solving the SIS problem for the set of parameters n, m, q, B, denoted $\mathsf{SISAdv}_{n,m,q,B}[\mathcal{A}]$, to be the probability that $\mathbf{A} \cdot \mathbf{x} = \mathbf{0} \pmod q$ and $\|\mathbf{x}\|_\infty \leq B$.

Note on notation. Recall the notation $\|\cdot\|_\infty$ from lecture. For a vector $\mathbf{x} \in \mathbb{Z}_q^m$, we say that $\|\mathbf{x}\|_\infty \leq B$ if for all $i \in [m]$, each entry of x_i of the vector $\mathbf{x}$ satisfies $x_i \in \{-B, -B+1, \ldots, 0, \ldots, B-1, B\}$. For example, $\|\mathbf{x}\|_\infty \leq 1$ means that $\mathbf{x} \in \{-1, 0, 1\}^m$.

Definition 1.1 (Universal Hash Function)**.** A hash function $H : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is *universal* if for every two distinct elements $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, we have

$$\Pr_{\mathsf{pk} \xleftarrow{R} \mathcal{K}} \left[H(\mathsf{pk}, \mathbf{x}) = H(\mathsf{pk}, \mathbf{x}')\right] = \frac{1}{|\mathcal{Y}|}.$$

The SIS hash function is, in fact, a universal hash function. Showing this is a simple exercise. We mentioned that universal hash functions are good randomness extractors. Randomness extractors are useful as a tool to *extract* some uniformly random bits from *unpredictable* sources. We formalized unpredictability with respect to the *guessing probability*.

Definition 1.2 (Guessing Probability)**.** Let $\mathcal{X}$ be a finite set and let $D_{\mathcal{X}}$ be some distribution on $\mathcal{X}$. Then, we define the guessing probability of $D_{\mathcal{X}}$ to be $\gamma = \max_{\mathbf{x}^* \in \mathcal{X}} \Pr_{\mathbf{x} \leftarrow D_{\mathcal{X}}}[\mathbf{x}^* = \mathbf{x}]$.

Then, we said that if we take a sample from a distribution $D_{\mathcal{X}}$ with small guessing probability and hash the result, we get something that looks uniformly random.

Theorem 1.3 (Simplified Leftover Hash Lemma)**.** *Let $\mathcal{X}$ and $\mathcal{Y}$ be two finite sets, and let $H : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ be a universal family of hash functions. Let $D_{\mathcal{X}}$ be some distribution on $\mathcal{X}$ with guessing probability at most γ. Then, for any (unbounded) adversary $\mathcal{A}$, we have*

$$\left| \Pr\left[\mathcal{A}(\mathsf{pk}, H(\mathsf{pk}, \mathbf{x})) = 1 \mid \mathsf{pk} \xleftarrow{R} \mathcal{K}, \mathbf{x} \leftarrow D_{\mathcal{X}}\right] - \Pr\left[\mathcal{A}(\mathsf{pk}, \mathbf{y}) = 1 \mid \mathsf{pk} \xleftarrow{R} \mathcal{K}, \mathbf{y} \xleftarrow{R} \mathcal{Y}\right] \right| = \gamma \cdot |\mathcal{Y}|.$$

The leftover hash lemma above is a significant simplification of the standard leftover hash lemma. It is much more general (and powerful) theorem, and generally, instead of formulating with respect to the distinguishing advantage of unbounded adversaries, we generally formulate it with respect to *statistical distance*. If you want to find out more about it, then refer to Section 8.10.4 in Boneh-Shoup (pg. 329).

Example. Recall that the SIS hash function is parameterized by the SIS parameter n, m, q, with associated domain and range $H : \mathbb{Z}_q^{n \times m} \times \{0,1\}^m \to \mathbb{Z}_q^n$. It is defined

$$H(\mathbf{A}, \mathbf{x}) = \mathbf{A} \cdot \mathbf{x} \mod q.$$

The leftover hash lemma says that if $m \geq 2n \log q$, then if you sample $\mathbf{A} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m}$, $\mathbf{x} \xleftarrow{\text{R}} \{0,1\}^m$, and $\mathbf{y} \xleftarrow{\text{R}} \mathbb{Z}_q^m$, we have

$$\big(\mathbf{A}, H(\mathbf{A}, \mathbf{x})\big) \quad \approx_{\mathsf{stat}} \quad \big(\mathbf{A}, \mathbf{y}\big)$$

or equivalently,

$$\big(\mathbf{A}, \mathbf{A} \cdot \mathbf{x}\big) \quad \approx_{\mathsf{stat}} \quad \big(\mathbf{A}, \mathbf{y}\big). \tag{1.1}$$

Why is this the case? The uniform distribution over $\{0,1\}^m$ has guessing probability $\gamma = \frac{1}{2^m} \leq \frac{1}{2^{2n \log q}}$. The size of the range $|\mathcal{Y}| = q^n = 2^{n \log q}$. Hence, the distinguishing advantage of any unbounded adversary is bounded by $\gamma \cdot |\mathcal{Y}| = \frac{1}{2^{n \log q}} = \frac{1}{q^n} = \text{negl}(n)$. Fact 1.1 is a very convenient fact that simplifies the security proofs in many lattice cryptosystems.

2 Inhongeneous SIS

Let's next discuss a simple variant of the SIS problem called the *inhomogenous* SIS problem ISIS.

$\mathsf{ISIS}(n, m, q, B)$: Let $n, m, q, B \in \mathbb{N}$ be positive integers. For a given adversary $\mathcal{A}$, we define the following experiment:

- The challenger samples $\mathbf{A} \xleftarrow{\text{R}} \mathbb{Z}_q^{n \times m}$, $\mathbf{y} \xleftarrow{\text{R}} \mathbb{Z}_q^n$, and gives $(\mathbf{A}, \mathbf{y})$ to the adversary $\mathcal{A}$.
- The adversary $\mathcal{A}$ outputs some vector $\mathbf{x} \in \mathbb{Z}_q^m$.

We define $\mathcal{A}$'s advantage in solving the ISIS problem for the set of parameters n, m, q, B, denoted $\mathsf{ISISAdv}_{n,m,q,B}\big[\mathcal{A}\big]$, to be the probability that $\mathbf{A} \cdot \mathbf{x} = \mathbf{y} \pmod q$ and $\|\mathbf{x}\|_\infty \leq B$.

2.1 OWF from ISIS

Recall the following function $f_{\mathbf{A}} : \{0,1\}^m \to \mathbb{Z}_q^n$ is a one-way function assuming $\mathsf{SIS}(n, m, q, B)$:

$$f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A} \cdot \mathbf{x}.$$

The way you would base the one-wayness of this function to $\mathsf{SIS}(n, m, q, B)$ is to go through $\mathsf{ISIS}(n, m, q, B)$. We can show that the function above is one-way assuming the $\mathsf{ISIS}(n, m, q, B)$. Then, we can show that $\mathsf{ISIS}(n, m, q, B)$ is hard assuming $\mathsf{SIS}(n, m, q, B)$.[1]

[1] Technically, in the homework, you will show that $\mathsf{ISIS}(n, m, q, B)$ is hard assuming $\mathsf{SIS}(n, m+1, q, B)$.

Definition 2.1 (One-Way Function)**.** We say that a function $f : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ is *one-way* if for any efficient adversary $\mathcal{A}$, for $\mathsf{pk} \xleftarrow{R} \mathcal{K}$, $\mathbf{x} \xleftarrow{R} \mathcal{X}$, setting $\mathbf{y} \leftarrow f(\mathbf{x})$,

$$\Pr[\mathcal{A}(\mathsf{pk}, \mathbf{y}) = \mathbf{x}' \wedge f(\mathsf{pk}, \mathbf{x}') = \mathbf{y}] = \mathrm{negl}(\lambda).$$

Let's see why the function $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{A} \cdot \mathbf{x}$ is one-way assuming ISIS.[2] If you compare the OWF security experiment with respect to $f_{\mathbf{A}}(\cdot)$, then basically, the adversary $\mathcal{A}$ is given $(\mathbf{A}, \mathbf{y}_0 = \mathbf{A} \cdot \mathbf{x})$, and must provide some preimage $\mathbf{x}'$ such that $\mathbf{A} \cdot \mathbf{x}' = \mathbf{y}_0$. In the ISIS experiment, the adversary $\mathcal{A}$ is provided $(\mathbf{A}, \mathbf{y}_1)$ for $\mathbf{y}_1 \xleftarrow{R} \mathbb{Z}_q^n$, and it must provide some $\mathbf{x}'$ such that $\mathbf{A} \cdot \mathbf{x}' = \mathbf{y}_1$. Here, the leftover hash lemma (Fact 1.1) says that $\mathbf{y}_0$ and $\mathbf{y}_1$ are generated statistically close. This immediately shows that for $\mathcal{A}$ to invert the function $f_{\mathbf{A}}(\cdot)$, it must solve ISIS.

2.2 Lattice Trapdoors

The one-way function $f_{\mathbf{A}}(\mathbf{x})$ is, in fact, not just a one-way function, but it can actually be used as a trapdoor function. There exists two efficient algorithms $\mathsf{TrapGen}$ and $f_{\mathbf{A}}^{-1}$ such that

$\mathsf{TrapGen}(n, m, q) \to (\mathbf{A}, \mathsf{td}_{\mathbf{A}})$: The trapdoor generation algorithm takes in matrix dimensions n, m, q and produces a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and some "trapdoor" information $\mathsf{td}_{\mathbf{A}}$.

$f_{\mathbf{A}}^{-1}(\mathsf{td}_{\mathbf{A}}, \mathbf{y}) \to \mathbf{x}$: The inversion algorithm takes in a trapdoor information $\mathsf{td}_{\mathbf{A}}$, and a vector $\mathbf{y} \in \mathbb{Z}_q^n$, and produces some $\mathbf{x} \in \mathbb{Z}_q^m$ such that $\mathbf{A} \cdot \mathbf{x} = \mathbf{y}$, and $\|\mathbf{x}\|_\infty \leq B$.

Bonus material. There are actually a number of ways of defining $\mathsf{td}_{\mathbf{A}}$ do this. Perhaps, the simplest way of doing this is what is known as $\mathbf{G}$-trapdoor. Let's assume that there exists some matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ such that the function

$$f_{\mathbf{G}}(\mathbf{x}) = \mathbf{G} \cdot \mathbf{x}$$

is efficiently invertible. The matrix $\mathbf{G}$ is often called the *gadget matrix*. Given $\mathbf{G}$, and $\mathbf{y} = \mathbf{G} \cdot \mathbf{x}$, *anyone* can efficiently find an $\mathbf{x}' \in \{0,1\}^m$ such that $\mathbf{G} \cdot \mathbf{x}' = \mathbf{y}$. How do we define $\mathbf{G}$? We will discuss more about the gadget matrix next lecture in the context of fully homomorphic encryption.

For a trapdoor function, we cannot just use $\mathbf{G}$ as the matrix $\mathbf{A}$ since this is *publicly* invertible. We want a function that is easy to invert *only if* you have some trapdoor. Therefore, we must random the matrix $\mathbf{G}$ somehow. The high level idea (very informal) way of doing this is as follows:

- $\mathsf{TrapGen}(n, 2m, q) \to (\mathbf{A}, \mathsf{td}_{\mathbf{A}})$: Sample a uniformly random matrix $\tilde{\mathbf{A}} \xleftarrow{R} \mathbb{Z}_q^{n \times m}$, and $\mathbf{R} \xleftarrow{R} \{0,1\}^{m \times m}$. Let $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ be a gadget matrix. Then, define set $\mathbf{A} = [\tilde{\mathbf{A}} | \tilde{\mathbf{A}} \cdot \mathbf{R} + \mathbf{G}] \in \mathbb{Z}_q^{n \times 2m}$, and let $\mathsf{td}_{\mathbf{A}} = \mathbf{R}$.
- $f_{\mathbf{A}}^{-1}(\mathsf{td}_{\mathbf{A}}, \mathbf{y}) \to \mathbf{x}$: Given a target $\mathbf{y}$, the inversion algorithm must find a vector $\mathbf{x} = [\mathbf{x}_0 | \mathbf{x}_1] \in \mathbb{Z}_q^{2m}$ such that $\mathbf{A} \cdot \mathbf{x} = \mathbf{y}$. It first computes a vector $\mathbf{x}^* \in \mathbb{Z}_q^m$ such that $\mathbf{G} \cdot \mathbf{x}^* = \mathbf{y}$.[3] Then, it sets $\mathbf{x}_0 = -\mathbf{R} \cdot \mathbf{x}^*$, and $\mathbf{x}_1 = \mathbf{x}^*$. It outputs $\mathbf{x} = [\mathbf{x}_0 | \mathbf{x}_1]$.

[2] Technically speaking, if $m >> n$, the fact that $f_{\mathbf{A}}(\cdot)$ is collision-resistant and is sufficiently compressing already implies one-wayness, but let's ignore this fact for now.

[3] Note that $\mathbf{G}$ is publicly invertible.

We note that the matrix $\mathbf{A} = [\tilde{\mathbf{A}}|\tilde{\mathbf{A}} \cdot \mathbf{R} + \mathbf{G}]$ is indistinguishable from a uniformly sampled matrix in $\mathbb{Z}_q^{n \times 2m}$. This is because $\tilde{\mathbf{A}}$ is sampled uniformly, and by the leftover hash lemma (again Fact 1.1), the matrix $\tilde{\mathbf{A}} \cdot \mathbf{R}$ is statistically uniform, completely randomizing the component $\tilde{\mathbf{A}} \cdot \mathbf{R} + \mathbf{G}$.

It is also easy to check that the output $\mathbf{x} = [\mathbf{x}_0|\mathbf{x}_1]$ that was produced by $f_{\mathbf{A}}^{-1}(\mathsf{td}_{\mathbf{A}}, \mathbf{y})$ satisfies $\mathbf{A} \cdot \mathbf{x} = \mathbf{y}$. Moreover, since $\mathbf{R} \in \{0,1\}^{m \times m}$, and $\mathbf{x}^* \in \{0,1\}^m$, the vector $\mathbf{x} = [\mathbf{R} \cdot \mathbf{x}^* \mid \mathbf{x}^*]$ have small entries.

So this is a high level idea of lattice trapdoors. However, the trapdoor function described above is actually **NOT SECURE**. This is because every preimage $\mathbf{x} = [\mathbf{R} \cdot \mathbf{x}^* \mid \mathbf{x}^*]$ leaks information about the secret trapdoor $\mathbf{R}$. In order to get a complete and secure trapdoor functions from lattices, we must do some extra work. If you want to find out more about lattice trapdoors, refer to the paper [1], which has a nice description of $\mathbf{G}$-trapdoors.

2.3 Digital Signatures from ISIS

Using the fact that $f_{\mathbf{A}}$ is a trapdoor function, we can construct a digital signature scheme. This is pretty much identical to the full-domain hash signatures that we saw in Lecture 2 where we used trapdoor permutations.

Let $\mathcal{M}$ be a message space. Fix a hash function (random oracle) $H : \mathcal{M} \to \mathbb{Z}_q^n$.

- $\mathsf{KeyGen}(1^\lambda)$: The key generation algorithm generates $(\mathbf{A}, \mathsf{td}_{\mathbf{A}}) \leftarrow \mathsf{TrapGen}(n, m, q)$. It sets $\mathsf{pk} = \mathbf{A}$ and $\mathsf{sk} = \mathsf{td}_{\mathbf{A}}$.
- $\mathsf{Sign}(\mathsf{sk}, \mu \in \mathcal{M})$: The signing algorithm first hashes the message $\mathbf{y} \leftarrow H(\mu)$. Then, it computes $\mathbf{x} \leftarrow f_{\mathbf{A}}^{-1}(\mathsf{td}_{\mathbf{A}}, \mathbf{y})$, and outputs the signature $\sigma = \mathbf{x} \in \mathbb{Z}_q^m$.
- $\mathsf{Verify}(\mathsf{pk}, \mu, \sigma = \mathbf{x})$: The verification algorithm hashes the message $\mathbf{y} \leftarrow H(\mu)$. Then, it checks if $\mathbf{A} \cdot \mathbf{x} = \mathbf{y}$, and $\|\mathbf{x}\|_\infty \leq B$. If both of these conditions are true, then output "accept" and otherwise, "reject".

In lecture 2, we proved the security of FDH signatures from trapdoor permutations in the random oracle model. The same proof can be adapted to show the security of the construction above. Hence, we get a digital signature scheme from the ISIS assumption and therefore, the SIS assumption.

3 Learning with Errors

$\mathsf{LWE}(n, m, q, \chi_B)$: Let $n, m, q, B \in \mathbb{N}$ be positive integers, and let χ_B be a B-bounded distribution over $\mathbb{Z}_q$. For a given adversary $\mathcal{A}$, we define the following two experiments:

Experiment b $\quad (b = 0, 1)$:

- The challenger computes

$$\mathbf{A} \xleftarrow{\text{R}} \mathbb{Z}_q^{m \times n}, \quad \mathbf{s} \xleftarrow{\text{R}} \mathbb{Z}_q^n, \quad \mathbf{e} \leftarrow \chi_B^m, \quad \mathbf{b}_0 \leftarrow \mathbf{A} \cdot \mathbf{s} + \mathbf{e}, \quad \mathbf{b}_1 \xleftarrow{\text{R}} \mathbb{Z}_q^m,$$

  and gives the tuple $(\mathbf{A}, \mathbf{b}_b)$ to the adversary.

- The adversary outputs a bit $\hat{b} \in \{0, 1\}$.

Let W_b be the event that $\mathcal{A}$ outputs 1 in Experiment b. Then, we define $\mathcal{A}$'s advantage in solving the LWE problem for the set of parameters n, m, q, χ_B to be

$$\mathsf{LWEAdv}_{n,m,q,\chi_B}[\mathcal{A}] := \Big| \Pr[W_0] - \Pr[W_1] \Big|.$$

Note on bounded distributions. We say that a distribution χ_B is a B-bounded distribution if

$$\Pr_{\mathbf{e} \leftarrow \chi_B} [\|\mathbf{e}\|_\infty \leq B] = 1.$$

As we discussed in lecture, for intuition, you can just think of χ_B as a uniform distribution over the space $\{-B, -B+1, \ldots, 0, \ldots, B\}$.

3.1 Regev Encryption

Let's construct a public key encryption scheme from the learning with errors assumption. The classical PKE scheme from LWE is *Regev encryption*, which was first introduced in [2]. The encryption scheme is defined with respect to the standard LWE parameters.

- $\mathsf{KeyGen}(1^\lambda) \to (\mathsf{sk}, \mathsf{pk})$: The key generation algorithm samples $\mathbf{A} \xleftarrow{\text{R}} \mathbb{Z}_q^{m \times n}$, $\mathbf{s} \xleftarrow{\text{R}} \mathbb{Z}_q^n$, and $\mathbf{e} \leftarrow \chi_B^m$. Then, it sets $\mathsf{sk} = \mathbf{s}$, and $\mathsf{pk} = (\mathbf{A}, \mathbf{v} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e})$.

- $\mathsf{Encrypt}(\mathsf{pk}, x \in \{0, 1\}) \to \mathsf{ct}$: The encryption algorithm samples a random binary vector $\mathbf{r} \xleftarrow{\text{R}} \{0, 1\}^m$. Then, it computes

$$\mathbf{c}_0 \leftarrow \mathbf{r}^T \mathbf{A}, \qquad c_1 \leftarrow \mathbf{r}^T \mathbf{v} + \frac{q}{2} \cdot x.$$

  It sets $\mathsf{ct} = (\mathbf{c}_0, c_1) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, and outputs ct.

- $\mathsf{Decrypt}(\mathsf{sk}, \mathsf{ct}) \to x$: The decryption algorithm computes $\tilde{x} = c_1 - \mathbf{c}_0 \cdot \mathbf{s}$. If $\|\tilde{x}\|_\infty \leq q/4$, then output $x = 0$. Otherwise, output $x = 1$.

Correctness. The decryption algorithm computes $c_1 - \mathbf{c}_0 \cdot \mathbf{s}$. Let's expand this computation out :

$$\begin{aligned} c_1 - \mathbf{c}_0 \cdot \mathbf{s} &= \mathbf{r}^T \mathbf{v} + \frac{q}{2} \cdot x - \mathbf{r}^T \mathbf{A} \cdot \mathbf{s} \\ &= \mathbf{r}^T (\mathbf{A} \cdot \mathbf{s} + \mathbf{e}) + \frac{q}{2} \cdot x - \mathbf{r}^T \mathbf{A} \cdot \mathbf{s} \\ &= \mathbf{r}^T \mathbf{A} \cdot \mathbf{s} + \mathbf{r}^T \mathbf{e} + \frac{q}{2} \cdot x - \mathbf{r}^T \mathbf{A} \cdot \mathbf{s} \\ &= \mathbf{r}^T \mathbf{e} + \frac{q}{2} \cdot x \end{aligned}$$

Note that $\mathbf{r} \in \{0,1\}^m$, and $\|\mathbf{e}\|_\infty \leq B$. Hence, $|\mathbf{r}^T \mathbf{e}| \leq mB < q/4$, and the original message $x \in \{0,1\}$ can be recovered by checking whether $c_1 - \mathbf{c}_0 \cdot \mathbf{s}$ is closer to 0 or $\frac{q}{2}$.

Security. We proceed via a sequence of hybrid experiments.

- Hyb_0: This is the real security experiment. The challenger generates the public key $\mathsf{pk} = (\mathbf{A}, \mathbf{v} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e})$ as in the real security experiment.

  For the challenge ciphertext, it samples $\mathbf{r} \xleftarrow{R} \{0,1\}^m$ as in the real scheme, and computes

  $$\mathbf{c}_0 \leftarrow \mathbf{r}^T \mathbf{A}, \qquad c_1 \leftarrow \mathbf{r}^T \mathbf{v} + \frac{q}{2} \cdot x.$$

- Hyb_1: In this experiment, the challenger generates the public key as $\mathsf{pk} = (\mathbf{A}, \mathbf{v} \xleftarrow{R} \mathbb{Z}_q^m)$. By LWE, this change is computationally indistinguishable to the adversary.

  For the challenge ciphertext, it samples $\mathbf{r} \xleftarrow{R} \{0,1\}^m$ as in the real scheme (but using the modified public key), and computes

  $$\mathbf{c}_0 \leftarrow \mathbf{r}^T \mathbf{A}, \qquad c_1 \leftarrow \mathbf{r}^T \mathbf{v} + \frac{q}{2} \cdot x.$$

- Hyb_2: In this experiment, the challenger generates the public key as $\mathsf{pk} = (\mathbf{A}, \mathbf{v} \xleftarrow{R} \mathbb{Z}_q^m)$ as in the previous hybrid.

  For the challenge ciphertext, it sets

  $$\mathbf{c}_0 \xleftarrow{R} \mathbb{Z}_q^n, \qquad c_1 \xleftarrow{R} \mathbb{Z}_q.$$

  Note that in Hyb_1, the challenger samples $\mathbf{r} \xleftarrow{R} \{0,1\}^m$ and sets $\mathbf{c}_0 \leftarrow \mathbf{r}^T \mathbf{A}$ and $c_1 \leftarrow \mathbf{r}^T \mathbf{v} + \frac{q}{2} \cdot x$. The leftover hash lemma (again Fact 1.1) says that this is (statistically) indistinguishable from sampling $\mathbf{c}_0 \xleftarrow{R} \mathbb{Z}_q^n$ and $c_0 \xleftarrow{R} \mathbb{Z}_q$. Hence, Hyb_1 and Hyb_2 are (statistically) indistinguishable by the leftover hash lemma.

In Hyb_2, the ciphertext is completely random. Therefore, the message is information theoretically hidden.

References

[1] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, 2012.

[2] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, 2005.