Lattice Trapdoors, Gaussian Sampling and More Cryptography

We will work with the ℓ_∞ norm throughout these lecture notes; tighter bounds are sometimes possible with the Euclidean norm but we would like to avoid the complication of computing the exact factors in favor of simplicity and conceptual clarity.

1 Lattice Trapdoors

Recall that

$$\Lambda^\perp(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}^m \,:\, \mathbf{A}\mathbf{z} = \mathbf{0} \pmod q\}$$

is a rank-m lattice. We will define a lattice trapdoor for a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ to be a short basis for $\Lambda^\perp(\mathbf{A})$. More generally, a set of short linearly independent vectors in $\Lambda^\perp(\mathbf{A})$ suffices. More explicitly:

Definition 1. *A matrix $\mathbf{T} \in \mathbb{Z}^{m\times m}$ is a β-good lattice trapdoor for a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ if*

1. *Each column vector of $\mathbf{T}$ is in the (right) mod-q kernel of $\mathbf{A}$, namely, $\mathbf{A}\mathbf{T} = \mathbf{0} \pmod q$;*
2. *Each column vector of $\mathbf{T}$ is short, namely $\|\mathbf{T}\|_\infty \le \beta$; and*
3. *$\mathbf{T}$ has rank m over $\mathbb{R}$.*

Note that the rank of $\mathbf{T}$ over $\mathbb{Z}_q$ can be no more than $m-n$; so, at first sight, the first and the third conditions may appear to be contradictory. However, the fact that we require *the real rank* over $\mathbf{T}$ to be large is the crucial thing here. This is related to why $\Lambda^\perp(\mathbf{A})$ as a lattice has rank m, even though as a linear subspace of $\mathbb{Z}_q^m$ has rank only $m-n$. Another way to look at $\mathbf{T}$ is that each of its columns is a homogenous SIS solution with respect to $\mathbf{A}$.

What good is such a trapdoor? We will demonstrate (in Section 3) its usefulness by showing that it can be used to solve both LWE and (inhomogenous) SIS with respect to $\mathbf{A}$.

2 Trapdoor Sampling

2.1 Leftover Hash Lemma

We will use the following form of the leftover hash lemma.

Lemma 2. *Let P be a probability distribution over $\mathbb{Z}^m$. The following two distributions have statistical distance at most ε as long as $H_\infty(P) \ge n\log q + 2\log(1/\varepsilon)$:*

$$(\mathbf{A}, \mathbf{A}\mathbf{e} \pmod q) \approx (\mathbf{A}, \mathbf{u})$$

where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$ is uniformly random, $\mathbf{e} \leftarrow X$ is drawn from the probability distribution P and $\mathbf{u} \leftarrow \mathbb{Z}_q^n$ is uniformly random. Here, $H_\infty(P)$ refers to the min-entropy of P.

For a proof, we refer the reader to these lecture notes.

2.2 Sampling a Random A with a Single Trapdoor Vector

Ajtai in 1996 gave us a procedure to sample a (statistically close to) uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ together with a single short vector $\mathbf{t} \in \mathbb{Z}^m$ such that $\mathbf{At} = \mathbf{0} \pmod q$. We begin our journey into trapdoors by describing this simple procedure.

1. Pick a uniformly random matrix $\mathbf{A}' \in \mathbb{Z}_q^{n \times (m-1)}$.
2. Pick a uniformly random vector $\mathbf{t}' \in \{0,1\}^{m-1}$.
3. Define
$$\mathbf{A} = [\mathbf{A}' \| -\mathbf{A}'\mathbf{t}' \pmod q] \quad \text{and} \quad \mathbf{t} = \begin{bmatrix} \mathbf{t}' \\ 1 \end{bmatrix}$$
as the matrix and trapdoor vector, respectively.

It is clear that $\mathbf{t}$ is a short vector in the right kernel of $\mathbf{A}$ mod q. It remains to show that $\mathbf{A}$ is close to uniformly random, which reduces to showing that $\mathbf{A}'\mathbf{t}$ is close to uniform given $\mathbf{A}'$. This follows directly from the leftover hash lemma assuming that $m \geq n \log q + \lambda$.

More generally, if we let $\|\mathbf{t}\|_\infty \leq B$, then we need $m \geq n \log q / \log B + \lambda$.

2.3 Ajtai-MP Trapdoor Sampling

Now, one can try to extend the above procedure to sample $\mathbf{A}$ together with more and more short vectors until you reach m (hopefully) linearly independent vectors and then we have a trapdoor! However, this naïve idea fails to work. Indeed, letting $m^* := n \log q + \lambda$, we can generate a close to uniform matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times (m^* + \ell)}$ together with ℓ trapdoor vectors (We leave it as an exercise to the reader to figure out how.) However, this will never "catch up" as the number of trapdoor vectors (ℓ) always remains short of the rank ($m^* + \ell$).

We start with the observation that an "inhomogenous trapdoor" (a notion that we will define in a minute) will let us achieve our goals of solving LWE and SIS just as well. An inhomogenous trapdoor $\mathbf{T} \in \mathbb{Z}^{m \times n \log q}$ is a matrix with short columns such that $\mathbf{AT} = \mathbf{G} \pmod q$ where $\mathbf{G}$ is a *gadget matrix* defined as below:
$$\mathbf{g} := \begin{bmatrix} 1 & 2 & 4 & \dots & 2^{\lceil \log q \rceil - 1} \end{bmatrix} \in \mathbb{Z}_q^{1 \times \lceil \log q \rceil} \text{ and } \mathbf{G} := \mathbf{I} \otimes \mathbf{g}$$
where $\mathbf{I}$ is the $n \times n$ identity matrix. In other words, $\mathbf{G}$ is the block diagonal $n \times (n\lceil \log q \rceil)$ matrix with $\mathbf{g}$ in each of its diagonal blocks.

Why does this suffice to solve LWE and SIS? Let's do LWE first. Given $\mathbf{b}^T = \mathbf{s}^T\mathbf{A} + \mathbf{e}^T \pmod q$, we do
$$\mathbf{b}^T\mathbf{T} = (\mathbf{s}^T\mathbf{A} + \mathbf{e}^T)\mathbf{T} = \mathbf{s}^T\mathbf{G} + \mathbf{e}^T\mathbf{T} \pmod q$$
In other words, we just transformed an LWE sample relative to $\mathbf{A}$ into an LWE sample relative to $\mathbf{G}$, with a slight increase in error. Now, if we have a trapdoor (in the sense of Definition 1) for $\mathbf{G}$ (and we will show in a few minutes that we do indeed have such a trapdoor), we can solve LWE!

To solve SIS w.r.t. $\mathbf{G}$, we simply note that given a vector $\mathbf{v} \in \mathbb{Z}_q^n$, it is easy to compute a bit vector $\mathbf{e}' \in \{0,1\}^{m^*}$ such that $\mathbf{Ge}' = \mathbf{v} \pmod q$. Indeed $\mathbf{e}'$ is simply the $n\lceil \log q \rceil$-dimensional bit-vector that consists of the bit representations of each element in $\mathbf{v}$.

Trapdoor for G: The case of $q = 2^k$. We invite the reader to think about this a bit before reading on. Let us first construct a trapdoor $\mathbf{T_g} \in \mathbb{Z}^{\lceil \log q \rceil \times \lceil \log q \rceil}$. We will then see that $\mathbf{T_G} = \mathbf{I} \otimes \mathbf{T_g}$. Indeed,

$$\mathbf{G} \cdot \mathbf{T_G} = (\mathbf{I} \otimes \mathbf{g}) \cdot (\mathbf{I} \otimes \mathbf{T_g}) = \mathbf{I} \otimes (\mathbf{g}\mathbf{T_g}) = \mathbf{0} \pmod{q}$$

Here is the trapdoor for $\mathbf{g}$:

$$\mathbf{T_g} = \begin{bmatrix} 2 & & & & & \\ -1 & 2 & & & & \\ & -1 & \dots & & & \\ & & \ddots & & & \\ & & & & 2 & \\ & & & & -1 & 2 \end{bmatrix}$$

Let us check.

- $\mathbf{T_g}$ has short columns. Indeed $\|\mathbf{T_g}\|_\infty = 2$.
- $\mathbf{g}\mathbf{T_g} = 0 \pmod{q}$.
- The determinant of $\mathbf{T_g}$ is $q = 2^k$. Therefore, it has full rank over $\mathbb{R}$. It decidedly *does not* have full rank over $\mathbb{Z}_q$ since its determinant is 0 mod q. (And this had better be the case!)

Trapdoor for G: The general case. As before, let us construct a trapdoor $\mathbf{T_g} \in \mathbb{Z}^{\lceil \log q \rceil \times \lceil \log q \rceil}$. We will then see that $\mathbf{T_G} = \mathbf{I} \otimes \mathbf{T_g}$. Here is the trapdoor for $\mathbf{g}$:

$$\mathbf{T_g} = \begin{bmatrix} 2 & & & & & | \\ -1 & 2 & & & & | \\ & -1 & \dots & & & | \\ & & \ddots & & \mathsf{bits}(q) & | \\ & & & 2 & & | \\ & & & -1 & & | \end{bmatrix}$$

The only difference is in the last column which is now the bit representation of the modulus q. Checking that this is indeed a trapdoor for $\mathbf{g}$ is left as an exercise. (Hint: for the full rank property, prove that the determinant of this matrix is q.)

Sampling A together with an Inhomogenous Trapdoor. Sample a uniformly random $\mathbf{B} \in \mathbb{Z}_q^{n \times m^*}$ where $m^* = n \log q + \lambda$ (as before). Set

$$\mathbf{A} = [\mathbf{B} \| \mathbf{B}\mathbf{R} + \mathbf{G}] \quad (\text{over } \mathbb{Z}_q)$$

where $\mathbf{R} \in \mathbb{Z}_q^{m^* \times m}$ is a uniformly random 0-1 matrix. Notice that

$$\mathbf{A} \cdot \begin{bmatrix} -\mathbf{R} \\ \mathbf{I} \end{bmatrix} = \mathbf{G} \pmod{q}$$

and since $\|\mathbf{R}\|_\infty \leq 1$, we have an inhomogenous trapdoor! Furthermore, $\mathbf{A}$ is close to random by leftover hash lemma (as before).

Now, one could stop here and directly use the inhomogenous trapdoor to solve LWE and SIS but we will go one step further and show how to get a trapdoor for $\mathbf{A}$.

Sampling A with a Trapdoor, Finally. First of all, we have

$$[\mathbf{B}\|\mathbf{BR}+\mathbf{G}]\cdot\begin{bmatrix}-\mathbf{R}\\ \mathbf{I}\end{bmatrix}=\mathbf{G}$$

Thus,

$$[\mathbf{B}\|\mathbf{BR}+\mathbf{G}]\cdot\begin{bmatrix}\mathbf{I} & -\mathbf{R}\\ \mathbf{0} & \mathbf{I}\end{bmatrix}=[\mathbf{B}\|\mathbf{G}]$$

Finally, multiplying this on the right by $\begin{bmatrix}\mathbf{I} & \mathbf{0}\\ -\mathbf{G}^{-}(\mathbf{B}) & \mathbf{T}_{\mathbf{G}}\end{bmatrix}$, we get

$$[\mathbf{B}\|\mathbf{BR}+\mathbf{G}]\cdot\underbrace{\begin{bmatrix}\mathbf{I} & -\mathbf{R}\\ \mathbf{0} & \mathbf{I}\end{bmatrix}\cdot\begin{bmatrix}\mathbf{I} & \mathbf{0}\\ -\mathbf{G}^{-}(\mathbf{B}) & \mathbf{T}_{\mathbf{G}}\end{bmatrix}}_{=\mathbf{T}_{\mathbf{A}}}=[\mathbf{B}\|\mathbf{G}]\cdot\begin{bmatrix}\mathbf{I} & \mathbf{0}\\ -\mathbf{G}^{-}(\mathbf{B}) & \mathbf{T}_{\mathbf{G}}\end{bmatrix}=\mathbf{0}\pmod q$$

Thus, the lattice trapdoor

$$\mathbf{T}_{\mathbf{A}}=\begin{bmatrix}\mathbf{I}+\mathbf{R}\mathbf{G}^{-}(\mathbf{B}) & -\mathbf{R}\mathbf{T}_{\mathbf{G}}\\ -\mathbf{G}^{-}(\mathbf{B}) & \mathbf{T}_{\mathbf{G}}\end{bmatrix}$$

We already saw that $\mathbf{A}\mathbf{T}_{\mathbf{A}}=\mathbf{0}\pmod q$. The ℓ_∞ norm of $\mathbf{T}_{\mathbf{A}}$ is $O(m)$. Finally, since $\mathbf{T}_{\mathbf{A}}$ is a product of two full-rank matrices, it is full-rank as well. (It has determinant q^n.)

3 Trapdoor Functions

Definition 3. *A family of functions*[1] $\mathcal{F}_n=\{f_i:\{0,1\}^n\to\{0,1\}^m\}$ *for some* $m=m(n)$ *is called a trapdoor function family if it comes with the following three associated polynomial-time algorithms.*

- *A probabilistic function generation algorithm that, on input* 1^n*, outputs an index* i *of a function* f_i *in the family as well as a trapdoor* t_i*.*
- *A deterministic evaluation algorithm that, on input* i *and* $x\in\{0,1\}^n$*, outputs* y*. We need that* $y=f_i(x)$*.*
- *A deterministic inversion algorithm that, on input* i, t_i *and* $y\in\{0,1\}^m$*, outputs* $x\in\{0,1\}^n$ *or a special symbol* $\perp$*. We require that if* $y\in\mathsf{Image}(f_i)$*, then* x *is* an *inverse, namely* $f_i(x)=y$*.*

3.1 Injective Trapdoor Function

The function

$$f_{\mathbf{A}}(\mathbf{s},\mathbf{e})=\mathbf{s}^T\mathbf{A}+\mathbf{e}^T\pmod q$$

where $\mathbf{A}\in\mathbb{Z}_q^n$, $\mathbf{s}\in\mathbb{Z}_q^n$ and $\mathbf{e}\leftarrow\chi^m$ is a one-way family of functions, under LWE. Given the trapdoor $\mathbf{T}$, one inverts this as follows.

$$(\mathbf{s}^T\mathbf{A}+\mathbf{e}^T)\mathbf{T}=\mathbf{e}^T\mathbf{T}\pmod q$$

Now, since the latter quantity has absolute value at most $q/4$, it is $\mathbf{e}^T\mathbf{T}$ (over the integers). The mod-q has no effect, and this is the key observation. Now, multiplying the latter by $\mathbf{T}^{-1}$ (the inverse of $\mathbf{T}$ over the reals) recovers $\mathbf{e}$. Here, it is *very* important that $\mathbf{T}$ had full rank over the reals; otherwise, $\mathbf{T}^{-1}$ would not exist.

[1] To be precise, we should be talking about ensemble of such families one for every input length n. However, we will refrain from unnecessary notational gymnastics and will take that as understood.

3.2 Surjective Trapdoor Function

The function

$$g_{\mathbf{A}}(\mathbf{e}) = \mathbf{A}\mathbf{e} \pmod{q}$$

where $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ where $m > n \log q$ and $\mathbf{e} \in [-\beta, \beta]^m$ is a one-way family of functions as well, under SIS, where $\beta = \mathsf{poly}(m)$.

With an inhomogeneous trapdoor, we claim that we can easily invert the function. Indeed, given a vector $\mathbf{v} \in \mathbb{Z}_q^n$, and an inhomogeneous trapdoor $\mathbf{R}$, we first compute a bit vector $\mathbf{e}'$ such that $\mathbf{G}\mathbf{e}' = \mathbf{v} \bmod q$ as discussed above. Now, we claim that

$$\mathbf{e} := \mathbf{R} \cdot \mathbf{e}'$$

is a required inverse. Indeed,

$$\mathbf{A} \cdot \mathbf{R} \cdot \mathbf{e}' = \mathbf{G} \cdot \mathbf{e}' = \mathbf{v} \pmod{q}$$

Also $\|\mathbf{e}\|_\infty \leq m$ since each entry of $\mathbf{R}$ and $\mathbf{e}'$ are binary.

4 Digital Signatures

Here is a simple digital signature scheme. (For a definition of digital signatures and what we mean by a secure digital signature, see Rafael Pass and abhi shelat's book.)

- The key generation algorithm samples a function together with a trapdoor. This would be $\mathbf{A}$ and $\mathbf{T}$. The public key is $\mathbf{A}$ and the secret key is $\mathbf{T}$.
- To sign a message m, first map it into the range of the function, e.g., by hashing it. That is, compute $\mathbf{v} = H(m)$. The signature is an inverse of $\mathbf{v}$ under the function $g_{\mathbf{A}}$. That is, a short vector $\mathbf{e}$ such that $\mathbf{A}\mathbf{e} = \mathbf{v} \pmod{q}$. This is guaranteed by the surjectivity of the function $g_{\mathbf{A}}$.
- Verification, given a message m, public key $\mathbf{A}$ and signature $\mathbf{e}$, consists of checking that $\mathbf{A}\mathbf{e} = H(m) \pmod{q}$ and that $\|\mathbf{e}\|_\infty \leq m^2$.

Unforgeability (given no signature queries) reduces to SIS in the random oracle model, i.e., assuming that H is a random oracle.

However, given signatures on adversarially chosen messages (in fact, even random messages), this scheme is broken. The key issue is that there are many inverses of $H(m)$, and the particular inverse computed using a trapdoor $\mathbf{T}$ leaks information about $\mathbf{T}$. Collecting this leakage over sufficiently many (polynomially many) signature queries enables an adversary to find $\mathbf{T}$, allowing her to forge signatures at will going forward.

This is most easily seen when the inversion procedure for $g_{\mathbf{A}}$ uses the inhomogenous trapdoor. Note that given $\mathbf{v}$, an adversary can compute $\mathbf{G}^{-}(\mathbf{v}) = \mathbf{e}'$ herself. She now gets a signature

$$\sigma = \mathbf{R} \cdot \mathbf{e}'$$

which gives her one equation on the secret $\mathbf{R}$. Given about m equations, she can solve linear equations and learn $\mathbf{R}$.

The situation remains essentially as dire even if you use the trapdoor (as opposed to the inhomogenous trapdoor). Using rounding vs the nearest plane algorithm does not help either; see the paper of Nguyen

and Regev for robust attacks against this signature scheme. The fundamental difficulty seems to stem from the fact that the inversion procedure is deterministic!

To mitigate the difficulty, we need a special kind of inverter for $g_{\mathbf{A}}$. The inverter is a "pre-image sampler"; that is, it is given the trapdoor $\mathbf{T}$ and produces a "random" pre-image. More precisely, we need the following distributions to be statistically close (computational indistinguishabilty is fine, but we will achieve statistical closeness):

$$\left(\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}, \mathbf{e} \leftarrow D_{\mathbb{Z}^m,s}, \mathbf{v} := \mathbf{Ae} \pmod q\right) \approx_s \left(\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}, \mathbf{e} \leftarrow \mathsf{PreSamp}(\mathbf{A},\mathbf{T},\mathbf{v}), \mathbf{v} \leftarrow \mathbb{Z}_q^n\right)$$

That is, the following processes produce statistically close outputs: (a) first sample $\mathbf{e}$ from a discrete Gaussian, and deterministically set $\mathbf{v}$ to be $\mathbf{Ae} \pmod q$; and (b) sample $\mathbf{v}$ uniformly and use the pre-image sampler to produce an inverse of $\mathbf{v}$ under $g_{\mathbf{A}}$ that is distributed according to the right conditional distribution. This distribution happens to be the discrete Gaussian over a coset of the lattice, that is,

$$\Lambda_{\mathbf{v}}^{\perp}(\mathbf{A}) := \{\mathbf{e} \in \mathbb{Z}^m : \mathbf{Ae} = \mathbf{v} \pmod q\}$$

In fact, this not quite enough; we need a multi-sample version of this. That is,

$$\left(\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}, \{\mathbf{e}_i \leftarrow D_{\mathbb{Z}^m,s}, \mathbf{v} := \mathbf{Ae} \pmod q\}_{i=1}^{\mathsf{poly}(\lambda)}\right) \approx_s \left(\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}, \{\mathbf{e} \leftarrow \mathsf{PreSamp}(\mathbf{A},\mathbf{T},\mathbf{v}), \mathbf{v} \leftarrow \mathbb{Z}_q^n\}_{i=1}^{\mathsf{poly}(\lambda)}\right)$$

This is quite cumbersome to work with, so we propose an alternate stronger definition. That is, we require that for most $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$ and any trapdoor $\mathbf{T}$ of length bounded by ℓ and $s \gg \ell$:

$$\left(\mathbf{e} \leftarrow D_{\mathbb{Z}^m,s}, \mathbf{v} := \mathbf{Ae} \pmod q\right) \approx_s \left(\mathbf{e} \leftarrow \mathsf{PreSamp}(\mathbf{A},\mathbf{T},\mathbf{v}), \mathbf{v} \leftarrow \mathbb{Z}_q^n\right)$$

Proof of Security. With the one change that the inverter is replaced by a pre-image sampler, our signature scheme becomes secure in the random oracle model. We showed the proof in the class.

5 Discrete Gaussian Sampling

Throughout, we will deal with sampling from a zero-centered discrete Gaussian.

5.1 Naïve Sampling

Let us first consider sampling from a discrete Gaussian over the simplest possible lattice, namely the one-dimensional lattice of integers $\mathbb{Z}$. The first idea to sample from the discrete Gaussian $D_{\mathbb{Z},s}$ is to sample from a continuous Gaussian N_s with parameter s and round to the nearest integer. Unfortunately, this is not a discrete Gaussian, not even statistically close to it. This is true even if s is much larger than the smoothing parameter.

Lemma 4. *The statistical distance between $D_{\mathbb{Z},s}$ and $\mathsf{Round}(N_s)$ is at least $1/s^3$.*

Proof. First, the probability assigned to zero by $\mathsf{Round}(N_s)$ is

$$\frac{2}{s} \cdot \int_0^{1/2} e^{-\pi x^2/s^2} dx = \frac{2}{\sqrt{\pi}} \cdot \int_0^{\sqrt{\pi}/2s} e^{-t^2} dt = \frac{2}{\sqrt{\pi}} \cdot \mathsf{erf}(\sqrt{\pi}/2s) \geq \frac{2}{\sqrt{\pi}} \cdot \left(\frac{\sqrt{\pi}}{2s} - \Omega\left(\frac{\sqrt{\pi}}{2s} \right)^3 \right)$$

where the latter is due to a Taylor series approximation of the erf function and holds for a sufficiently large s. This quantity is at most

$$1/s - \Omega(1/s^3)$$

On the other hand, let's compute

$$\sum_{x \in \mathbb{Z}} \rho_s(x) = s \cdot \sum_{x \in \mathbb{Z}} \rho_{1/s}(x) = s \cdot (1 + \mathrm{negl}(\lambda))$$

if s is above the $\mathrm{negl}(\lambda)$-smoothing parameter of $\mathbb{Z}$ which is $\omega(\sqrt{\log \lambda})$.

Therefore, the probability assigned to zero by $D_{\mathbb{Z},s}$ is

$$\frac{1}{\sum_{x \in \mathbb{Z}} e^{-\pi x^2/s^2}} \approx 1/s$$

upto a negligible term.

Thus, the statistical distance between the two distributions in question is $\Omega(1/s^3)$ which is non-negligible unless s itself is super-polynomial. □

For n-dimensional lattices, this statistical distance degrades with n as well making the situation much worse.

The reader may recall that the first step of Regev's worst-case to average-case reduction was sampling from a discrete Gaussian over a lattice for which Regev used the above procedure. However, he could afford to use an exponential s which makes the statistical distance small.

5.2 Sampling Discrete Gaussians over $\mathbb{Z}$

So, how do we sample from $D_{\mathbb{Z},s}$ for polynomial s? We will show that the general method of rejection sampling works. Let $Z = [-t \cdot s, t \cdot s]$ be a sufficiently large interval, where $t = \omega(\sqrt{\log \lambda})$. We do the following:

1. Sample a random integer $z \leftarrow Z$.
2. Output z with probability $\rho_s(z) := e^{-\pi z^2/s^2}$; else go to step 1 and repeat.

First of all, we will show that the probability that $D_{\mathbb{Z},s}$ assigns to numbers outside of the interval Z is negligible.

Lemma 5. *Let $s \geq \eta_\varepsilon(\mathbb{Z})$ for some $\varepsilon = \mathrm{negl}(\lambda)$, and $t > 0$. We have*

$$\Pr_{x \leftarrow D_{\mathbb{Z},s}}[|x| > t \cdot s] \leq c \cdot e^{-\pi t^2}$$

for some absolute constant $c > 0$.

Consider the probability distribution $D'_{\mathbb{Z},s}$ which assigns probability $\rho_s(x)$ for all $x \in Z \cap \mathbb{Z}$ and 0 otherwise. The lemma above shows that $D'_{\mathbb{Z},s}$ is close to $D_{\mathbb{Z},s}$ if s is larger than the negl(λ)-smoothing parameter of $\mathbb{Z}$, namely $\omega(\sqrt{\log n})$, and $t = \omega(\sqrt{\log n})$.

It is not hard to see that the procedure above samples from the distribution $D'_{\mathbb{Z},s}$ exactly. It remains to see that it terminates in polynomial time. We show two things which we leave as an exercise: (a) the probability that z sampled in step 1 lies in $[-s, s]$ is $\Omega(1/t)$ and (b) if such a z is sampled, it is output with probability $\Omega(1)$. Put together, the expected time for termination is $O(t) = \mathsf{poly}(\lambda)$.

5.3 Klein-GPV algorithm

We demonstrate the sampler in two dimensions. The generalization to n dimensions follows quite naturally.

6 Identity-based Encryption

Let us think first about deploying a public-key encryption scheme on a large scale. We need a mechanism to maintain a directory of (ID, PK) pairs where ID is the identifying information of a person, say Alice's e-mail address or phone number, that other people use to send her a message. Then, when you wish to send an email to Alice, you look up her public key in the directory and encrypt to the public key.

The directory, which forms part of a public-key infrastructure (PKI), has to be authenticated and trusted. For example, an adversary should not be able to insert an entry of the form (ID_A, PK'_A), where she presumably knows SK'_A, into the directory.

Identity-based encryption (IBE) solves the problem of having to maintain an authenticated PKI. In an IBE:

- there is a master authority who generates a master public key MPK together with a master secret key MSK, and publishes the MPK.
- To encrypt a message μ, one needs to know MPK and the identity ID (e.g., the e-mail address) of the recipient.
- Each user goes to the master authority and receives SK_{ID} after authenticating that they indeed are the owner of ID.
- Using SK_{ID}, the user can decrypt ciphertexts encrypted to the identity ID.

Let us now define the syntax of an IBE, formalizing the discussion above.

- $\mathsf{Setup}(1^\lambda)$: is a probabilistic algorithm that generates a master public key MPK and a master secret key MSK.
- $\mathsf{Enc}(MPK, ID, \mu)$: is a probabilistic algorithm that generates a ciphertext C of a message μ (for simplicity, we will encrypt bits but that is largely irrelevant) w.r.t. identity ID.
- $\mathsf{KeyGen}(MSK, ID \in \{0,1\}^*)$: is a probabilistic algorithm that generates a secret key SK_{ID}.
- $\mathsf{Dec}(SK_{ID}, C)$: is a deterministic decryption algorithm.

You may have noticed that the master authority can decrypt *all* the ciphertexts generated in this system and is therefore *very powerful*.

Application: Access Delegation across Space. I can act as the master authority and use an IBE to delegate decryption of certain subsets of messages to other people (e.g., my administative assistant). For example, all messages are tagged with a keyword $ID = \texttt{CS294}$, and I can issue the SK_{ID} to my assistant that lets him decrypt only those messages tagged with ID.

Application: Access Delegation across Time. Imagine that I go on (virtual) vacation to Cancun and want to take my laptop. However, I am worried that it will be stolen. So, I ask folks encrypting messages to me to use an IBE and tag the messages with an ID which is the current date. This allows me to generate a small set of secret keys, corresponding to the days that I am away, which allows me to decrypt only the corresponding small subset of messages. IBE lets me enjoy my vacation worry-free!

Application: Chosen-Ciphertext Security. IBE can be used in surprisingly non-trivial ways to construct other cryptographic systems, e.g., chosen ciphertext secure public-key encryption schemes and digital signature schemes (that we will describe later in this lecture).

Constructions. The first constructions used bilinear maps on elliptic curves (Boneh-Franklin'00) and quadratic residuosity (Cocks'00). We will present the third IBE scheme from the LWE assumption (Gentry-Peikert-Vaikuntanathan'08) and several variants today. Recently, Garg and Dottling have come up with a completely different scheme that relies on Diffie-Hellman groups (no need for bilinear maps!) Following up, Brakerski-Lombardi-Segev-Vaikuntanathan came up with a scheme based on learning parity with very low noise.

6.1 Definitions of Security

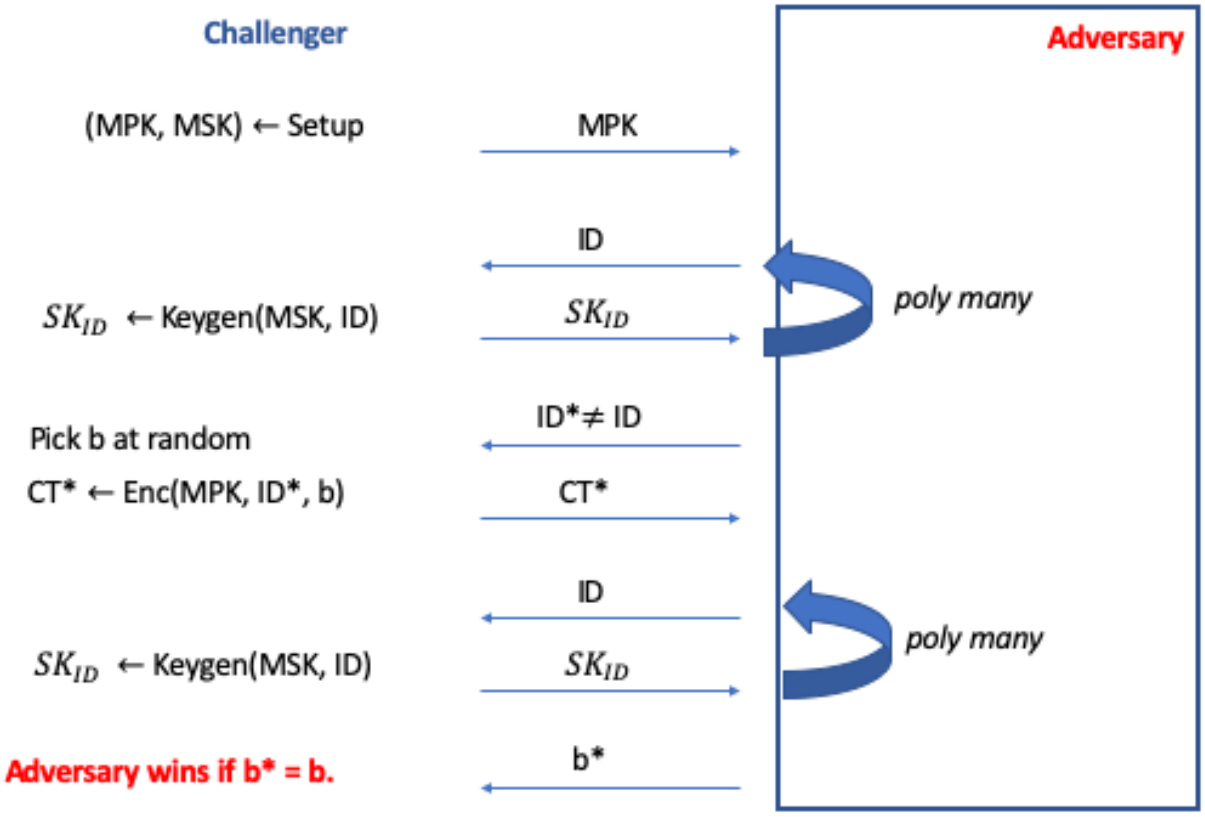
We imagine a PPT adversary that plays the following game with a challenger. This captures the requirement that encryptions relative to ID^* should be secure even to an adversary that can obtain secret keys for polynomially many *different* identities $ID \neq ID^*$. This is called the *adaptive security* or *full security* definition. The weaker *selective* security definition restricts the adversary to pick the identity it is attacking at the very beginning of the game (before it receives MPK).

Selectively secure IBE schemes can be generically proven to be fully secure under a sub-exponentially stronger assumption. Therefore, we will not attempt to optimize the strength of the assumption and focus on selective security for this lecture.

6.2 IBE=Signatures+Public-Key Encryption

Moni Naor observed that any IBE scheme gives us *for free* a digital signature scheme. **The intuition is that the identity secret key SK_{ID} can act as a signature for the "message" ID.** How so?

- It can be generated using the master secret key MSK (which will serve as the secret signing key.)
- It can be verified using the master public key MPK – indeed, encrypt a bunch of random messages using MPK and attempt to use the "signature" to decrypt. If decryption produces the correct message, accept the signature. Otherwise, reject.
- after receiving signatures SK_{ID} on polynomially many messages ID, being able to produce the "signature" on a different message ID^* constitutes a signature forgery; but being able to do that breaks



IBE security. Conversely, in a signature scheme derived from a secure IBE scheme, it should be infeasible to do that.

Indeed, turning this around, we will use the GPV signature scheme we saw in the last class as a starting point to build an IBE scheme.

7 Recap: GPV Signatures

- KeyGen(1^λ): Generate a random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and its trapdoor $\mathbf{T}$ by running TrapSamp.
- Sign(μ): first compute $\mathbf{v} = H(\mu) \in \mathbb{Z}_q^n$ where H is treated as a random oracle in the analysis. Then, use Gaussian sampling (via the GPV algorithm) to compute a Gaussian solution $\mathbf{e} \in \mathbb{Z}^m$ to the equation

$$\mathbf{Ae} = \mathbf{v} \pmod{q}$$

Let $\Lambda^\perp(\mathbf{A})$ denote the lattice

$$\{\mathbf{e} \in \mathbb{Z}^m : \mathbf{Ae} = \mathbf{0} \pmod{q}\}$$

and let $\Lambda_{\mathbf{v}}^{\perp}(\mathbf{A})$ denote a coset of $\Lambda^{\perp}(\mathbf{A})$ indexed by $\mathbf{v}$. That is,

$$\Lambda_{\mathbf{v}}^{\perp}(\mathbf{A}) = \{\mathbf{e} \in \mathbb{Z}^m \ : \ \mathbf{Ae} = \mathbf{v} \pmod{q}\}$$

Note that the distribution of $\mathbf{e}$ is $D_{\Lambda_{\mathbf{v}}^{\perp}(\mathbf{A}),\sigma}$ where $\sigma \approx \|\mathbf{T}\| \cdot \omega(\sqrt{\log n})$. (The $\omega(\sqrt{\log n})$ is so that the sampling algorithm can achieve negligible statistical distance from a true discrete Gaussian.)

- $\mathsf{Verify}(\mathbf{A}, \mathbf{e}, \mu)$: check that (1) $\mathbf{e}$ is short, that is $\|\mathbf{e}\| \leq \|T\| \cdot \omega(\sqrt{n \log n})$; and (2) $\mathbf{Ae} = H(\mu) \pmod{q}$.

The key question now is how to we build an encryption algorithm whose public key is $\mathbf{v}$ (which will be treated as $H(ID)$) and the corresponding private key is $\mathbf{e}$ as above. Indeed, we have seen precisely such a scheme in the first lecture (cf. lecture notes) called the GPV encryption scheme or more commonly, the dual-Regev encryption scheme.

But before we get there, the scheme as stated above is insecure – do you see why? Bonus points if you see how to fix it.

8 The Dual Regev Encryption Scheme

Let's recall the dual Regev scheme we saw in the first lecture (notes).

- KeyGen: the public key is an LWE matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a random vector $\mathbf{v} \in \mathbb{Z}_q^n$. The private key is a short vector $\mathbf{e}$ such that $\mathbf{Ae} = \mathbf{v} \pmod{q}$.

$$\mathsf{pk} = (\mathbf{A}, \mathbf{v}) \qquad \mathsf{sk} = \mathbf{e}$$

- $\mathsf{Enc}(\mathsf{pk}, \mu)$: pick an LWE secret $\mathbf{s} \in \mathbb{Z}_q^n$ and output

$$(\mathbf{c}_1^T, c_2) := \left(\mathbf{s}^T\mathbf{A} + \mathbf{x}^T, \mathbf{s}^T\mathbf{v} + x' + m\lfloor q/2 \rfloor\right)$$

as the ciphertext. We will call this ciphertext the dual Regev encryption of μ relative to $\mathbf{A}$ and $\mathbf{v}$.

- $\mathsf{Dec}(\mathsf{sk}, (\mathbf{c}_1^T, c_2))$: Compute

$$\tilde{\mu} := \mathsf{Round}(c_2 - \mathbf{c}_1^T\mathbf{e})$$

where $\mathsf{Round}(\alpha)$ outputs 1 if $|\alpha - q/2| \leq q/4$ and 0 otherwise.

We will leave the correctness and security as an exercise. (Alternatively, look at lecture 1.)

9 The GPV IBE Scheme

- $\mathsf{Setup}(1^\lambda)$: Pick the right $n = n(\lambda)$ for a security level of λ bits. Generate a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and its trapdoor $\mathbf{T} \in \mathbb{Z}^{m \times m}$ by running the trapdoor sampling algorithm.

$$(\mathbf{A}, \mathbf{T}) \leftarrow \mathsf{TrapSamp}(1^n)$$

(The parameters m and q are picked internally by the trapdoor sampling algorithm.) The master public key is $\mathsf{mpk} = \mathbf{A}$ and the master secret key is $\mathsf{msk} = \mathbf{T}$.

- $\mathsf{KeyGen}(\mathsf{msk}, ID)$: Compute $\mathbf{v} := H(ID) \in \mathbb{Z}_q^n$ where $H : \{0,1\}^* \to \mathbb{Z}_q^n$ is a hash function (which, in the security analysis, will be treated as a random oracle.) Generate a short vector

$$\mathbf{e} \leftarrow \mathsf{DGSamp}(\mathbf{A}, \mathbf{T}, \mathbf{v})$$

  by running the discrete Gaussian sampling algorithm. Recall that $\mathbf{Ae} = \mathbf{v} \pmod q$. Output the secret key $sk_{ID} = \mathbf{e}$.

- $\mathsf{Enc}(\mathsf{mpk}, ID, \mu)$: Run the dual Regev encryption algorithm with $\mathsf{pk} := (\mathbf{A}, \mathbf{v} = H(ID))$ and message μ and output the resulting ciphertext.

- $\mathsf{Dec}(\mathsf{sk}_{ID}, c)$: Run the dual Regev decryption algorithm with $\mathsf{sk} := \mathsf{sk}_{ID} = \mathbf{e}$.

9.1 Proof of (Full) Security

We will come up with alternate algorithms called Setup^*, KeyGen^* and Enc^* (Dec^* will be the same as Dec) which the challenger will run. Our goal will be to show that (1) the adversary cannot distinguish between the challenger running Algorithm *vs* Algorithm* and (2) Algorithms* do not need the master secret key and moreover, a challenger using Algorithm* can use a successful adversary to break LWE.

A crucial advantage of Algorithm* for the GPV scheme is that it can use the programmability of the random oracle as we will see below. We will for simplicity first create algorithms for the selective security game.

- $\mathsf{Setup}^*(ID^*, 1^\lambda)$: Sample random $\mathbf{A}^*$ which forms the MPK^* (no need for trapdoor).

- $\mathsf{Hash}^*(ID)$: Set $H(ID^*) = \mathbf{v}^*$, a random vector in $\mathbb{Z}_q^n$. For all other IDs, set $H(ID) = \mathbf{A}^*\mathbf{e}_{ID}$ where $\mathbf{e}_{ID}$ is chosen from a Gaussian. Remember $\mathbf{e}_{ID}$.

- $\mathsf{KeyGen}^*(ID)$: We know that $ID \neq ID^*$. So, we know the $\mathbf{e}_{ID}$ by construction! This is a consequence of working in the random oracle model!

- $\mathsf{Enc}^*(MPK^*, ID^*, \mu)$: return the dual Regev encryption of μ relative to $\mathbf{A}^*$ and $\mathbf{v}^*$.

The Algorithm* produce the same distribution as the original algorithms. Thus, an adversary will break the challenge ciphertext when interacting with Algorithm* just as well as with Algorithms. By embedding the dual-Regev challenge matrix $\mathbf{A}$ as the master public key and the dual-Regev public key $\mathbf{v}^*$ as the hash of ID^*, we can easily turn the IBE adversary into an attack against the dual Regev public key encryption scheme.

A Note on Full Security. Since Setup^* does not know ID^*, it guesses which of the (polynomially many) hash queries will be for ID^*. (1) any adversary that succeeds has to know $H(ID^*)$ which it can only find out by making a hash query; and (2) if the guess is correct (happens with probability $1/Q$) we can translate an IBE breaker into a dual-Regev breaker just as above.

10 The CHKP IBE Scheme

The CHKP Trick: Trapdoor Extension.
Given the trapdoor for a matrix $\mathbf{A}$, can you generate a trapdoor for $[\mathbf{A}\|\mathbf{B}]$ where $\mathbf{B}$ is an arbitrary matrix?

10.1 The Scheme

- $\mathsf{Setup}(1^\lambda)$: Pick the right $n = n(\lambda)$ for a security level of λ bits. Generate matrices

$$\mathbf{A}_{1,0}, \mathbf{A}_{1,1}, \ldots, \mathbf{A}_{\ell,0}, \mathbf{A}_{\ell,1} \in \mathbb{Z}_q^{n \times m}$$

  where ℓ is the length of the identities. The master public key is

$$\mathsf{mpk} = \left(\mathbf{A}_{i,b}\right)_{i \in [\ell], b \in \{0,1\}}, \mathbf{v}$$

  where $\mathbf{v} \in \mathbb{Z}_q^n$ is a random vector, and the master secret key is

$$\mathsf{msk} = (\mathbf{T}_{\mathbf{A}_0}, \mathbf{T}_{\mathbf{A}_1})$$

  We will never use the trapdoors for the other matrices (except in the security proof.)

- $\mathsf{KeyGen}(\mathsf{msk}, ID \in \{0,1\}^\ell)$: Let

$$\mathbf{A}_{ID} := [\mathbf{A}_{1,ID_1} || \mathbf{A}_{2,ID_2} || \ldots || \mathbf{A}_{\ell,ID_\ell}]$$

  where $ID_1, \ldots, ID_\ell$ are the bits of ID. Generate a short vector $\mathbf{e} \leftarrow \mathsf{DGSamp}(\mathbf{A}_{ID}, \mathbf{T}_{\mathbf{A}_{ID}}, \mathbf{v})$ by running the discrete Gaussian sampling algorithm. Recall that $\mathbf{A}_{ID} \cdot \mathbf{e} = \mathbf{v} \pmod q$. Output the secret key $sk_{ID} = \mathbf{e}$.

- $\mathsf{Enc}(\mathsf{mpk}, ID, \mu)$: Run the dual Regev encryption algorithm with $\mathsf{pk} := (\mathbf{A}_{ID}, \mathbf{v})$ and message μ and output the resulting ciphertext.

- $\mathsf{Dec}(\mathsf{sk}_{ID}, c)$: Run the dual Regev decryption algorithm with $\mathsf{sk} := \mathsf{sk}_{ID} = \mathbf{e}$.

10.2 Proof of (Selective) Security

As before, we will come up with alternate algorithms called $\mathsf{Setup}^*, \mathsf{KeyGen}^*$ and Enc^* (Dec^* will be the same as Dec) which the challenger will run. We will not be able to use random oracles here.

- $\mathsf{Setup}^*(ID^*, 1^\lambda)$: sample random $\mathbf{v}^*$. sample ℓ random matrices $\mathbf{B}_1, \ldots, \mathbf{B}_\ell$ and set

$$\mathbf{A}_{i,ID_i^*} = \mathbf{B}_i$$

  sample ℓ matrices $\mathbf{B}'_1, \ldots, \mathbf{B}'_\ell$ together with their trapdoors and set

$$\mathbf{A}_{i,1-ID_i^*} = \mathbf{B}'_i$$

  MPK^* consists of all the $\mathbf{A}_{i,b}$ and $\mathbf{v}^*$. MSK^* consists of the trapdoors of all $\mathbf{A}_{i,1-ID_i^*}$.

- $\mathsf{KeyGen}^*(ID)$: We know that $ID \neq ID^*$. Therefore, I know the trapdoor of the matrix

$$\mathbf{A}_{ID} := [\mathbf{A}_{1,ID_1} || \ldots || \mathbf{A}_{\ell,ID_\ell}]$$

  (do you see why?)

- $\mathsf{Enc}^*(MPK^*, ID^*, \mu)$: return the dual Regev encryption of μ relative to $\mathbf{A}_{ID^*}$ and $\mathbf{v}^*$. (note that MSK^* does not tell us anything about a trapdoor for $\mathbf{A}_{ID^*}$.)

One can also prove full security with a more sophisticated proof. In one sentence, the idea is to set up $\mathbf{A}_{i,b}$ so that Algorithm* can generate secret keys for all the Q secret key queries and yet *not* be able to generate the secret key for ID^*.

10.3 CHKP: Pros and Cons

- PLUS: the scheme is secure without resorting to the random oracle model.
- MINUS: the public parameters are rather large, namely $O(nm \log q \cdot \ell)$ as opposed to GPV where it is $O(nm \log q)$. Consequently, also ciphertexts are large.
- PLUS: While we only showed selective security, one can augment the scheme to be adaptively (fully) secure.
- PLUS: The scheme naturally extends to a hierarchical IBE scheme, described next.

10.4 A Brief Note on Hierarchical IBE

Think of hierarchies in an organization. The CEO (the master key generator) can delegate access to the VP of Engineering who can in turn delegate to programmers and so forth (but not the other way round). In a hierarchical IBE, one can generate SK_{ID} using MSK; in turn, the owner of SK_{ID} can generate $SK_{ID||ID'}$ etc.

The CHKP scheme has a natural hierarchical structure. Namely, if you know the trapdoor for $\mathbf{A}_{ID}$, you can generate a trapdoor for $\mathbf{A}_{ID||ID'} = [\mathbf{A}_{ID}||\mathbf{A}_{ID'}]$. Constructing a HIBE scheme building off of this idea is left as an exercise.

11 The ABB IBE Scheme

The ABB Trick: Punctured Trapdoors.
Given the trapdoor for a matrix $\mathbf{A}_0$, a matrix $\mathbf{R}$ with small entries, and a trapdoor for $\mathbf{G}$, can you generate a trapdoor for

$$[\mathbf{A}_0||\mathbf{A}_0\mathbf{R} + \alpha \cdot \mathbf{G}]$$

for an arbitrary integer $\alpha \neq 0 \pmod q$?

How about for $\alpha = 0 \pmod q$, that is, $[\mathbf{A}_0||\mathbf{A}_0\mathbf{R}]$?

11.1 The Scheme

- $\mathsf{Setup}(1^\lambda)$: Pick the right $n = n(\lambda)$ for a security level of λ bits. Generate matrices

  $$\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{n \times m}$$

  The master public key is

  $$\mathsf{mpk} = \mathbf{A}_0, \mathbf{A}_1, \mathbf{v}$$

  where $\mathbf{v} \in \mathbb{Z}_q^n$ is a random vector, and the master secret key is

  $$\mathsf{msk} = \mathbf{T}_{\mathbf{A}_0}$$

  We will never use the trapdoor for $\mathbf{A}_1$.

- $\mathsf{KeyGen}(\mathsf{msk}, ID \in \{0,1\}^\ell)$: Let h be a collision-resistant hash function that maps identities to $\mathbb{Z}_q^*$. Define

$$\mathbf{A}_{ID} := [\mathbf{A}_0 || \mathbf{A}_1 + h(ID) \cdot \mathbf{G}]$$

  where $\mathbf{G}$ is the gadget matrix. Note that by trapdoor extension, KeyGen knows a trapdoor for $\mathbf{A}_{ID}$ for any ID.

  Generate a short vector $\mathbf{e} \leftarrow \mathsf{DGSamp}(\mathbf{A}_{ID}, \mathbf{T}_{\mathbf{A}_{ID}}, \mathbf{v})$ by running the discrete Gaussian sampling algorithm. Recall that $\mathbf{A}_{ID} \cdot \mathbf{e} = \mathbf{v} \pmod q$. Output the secret key $sk_{ID} = \mathbf{e}$.

- $\mathsf{Enc}(\mathsf{mpk}, ID, \mu)$: Run the dual Regev encryption algorithm with $\mathsf{pk} := (\mathbf{A}_{ID}, \mathbf{v})$ and message μ and output the resulting ciphertext.

- $\mathsf{Dec}(\mathsf{sk}_{ID}, c)$: Run the dual Regev decryption algorithm with $\mathsf{sk} := \mathsf{sk}_{ID} = \mathbf{e}$.

11.2 ABB: Proof of Selective Security

As before, we will come up with a bunch of alternate algorithms called $\mathsf{Setup}^*, \mathsf{KeyGen}^*$ and Enc^* (Dec^* will be the same as Dec) which the challenger will run. We will not be able to use random oracles here either.

- $\mathsf{Setup}^*(ID^*, 1^\lambda)$: sample random $\mathbf{v}^*$. sample a random matrix $\mathbf{A}_0$ and a matrix $\mathbf{R}$ with small entries. Set

$$\mathbf{A}_1 := [\mathbf{A}_0 || \mathbf{A}_0\mathbf{R} - h(ID^*)\mathbf{G}]$$

  MPK^* consists of $\mathbf{A}_0, \mathbf{A}_1$ and $\mathbf{v}^*$. MSK^* consists of $\mathbf{R}$ (and the trapdoor for $\mathbf{G}$.)

- $\mathsf{KeyGen}^*(ID)$: We know that $ID \neq ID^*$. Therefore, I know the trapdoor of the matrix

$$\mathbf{A}_{ID} := [\mathbf{A}_0 || \mathbf{A}_1 + h(ID)\mathbf{G}] = [\mathbf{A}_0 || \mathbf{A}_0\mathbf{R} + (h(ID) - h(ID^*)\mathbf{G}]$$

  (do you see why?)

- $\mathsf{Enc}^*(MPK^*, ID^*, \mu)$: given a dual Regev encryption of μ relative to $\mathbf{A}_0$ and $\mathbf{v}^*$, compute a dual Regev encryption of μ relative to

$$\mathbf{A}_{ID^*} = [\mathbf{A}_0 || (\mathbf{A}_0\mathbf{R} - h(ID^*)\mathbf{G}) + h(ID^*)\mathbf{G}] = [\mathbf{A}_0 || \mathbf{A}_0\mathbf{R}]$$

  and $\mathbf{v}^*$. (do you see how to do this?)

11.3 ABB: Pros and Cons

- PLUS: the scheme is secure without resorting to the random oracle model.
- PLUS: the public parameters and ciphertexts are as small as GPV, namely $O(nm \log q)$.
- PLUS: Can be extended to full security.
- PLUS: Extensible to hierarchical IBE. A different ABB paper uses additional techniques to construct a "better" HIBE (where the lattice dimension stays the same regardless of the number of levels of delegation).

12 Application: Chosen Ciphertext Secure Public-key Encryption

We will now show a very simple construction of a chosen ciphertext secure (CCA2-secure) public-key encryption scheme from IBE. This is due to Canetti, Halevi and Katz [?]. In fact, here we will describe a solution for the weaker notion of CCA1-security.

But first, the definition of CCA1-security. In the CCA1 game, the adversary gets the public-key PK of the encryption scheme, and can ask to get polynomially many ciphertexts decrypted. That is, a challenger will, on input c, run $\mathsf{Dec}(SK, c)$ and return the answer to the adversary. Note that c need not be distributed like an honestly generated ciphertext, and may not even live in the range of the encryption algorithm (i.e., may not be a valid ciphertext). Eventually, the adversary gets an encryption of a random bit b under PK and is asked to guess b. CCA1 security requires that no PPT adversary can guess b with probability better than $1/2 + \mathsf{negl}(\lambda)$.

Here is the construction.

- $\mathsf{KeyGen}(1^\lambda)$: run $\mathsf{IBE.Setup}(1^\lambda)$ to get an MPK_{IBE} and an MSK_{IBE}. The public key PK of the CCA scheme is MPK_{IBE} and the secret key SK is MSK_{IBE}.
- $\mathsf{Enc}(PK, \mu)$: pick a random string ID. Run $\mathsf{IBE.Enc}(PK = MPK_{IBE}, ID, \mu)$ and output ID together with the resulting ciphertext.
- $\mathsf{Dec}(SK, (ID, c))$: use $SK = MSK_{IBE}$ to create SK_{ID} and run the IBE decryption algorithm $\mu = \mathsf{IBE.Dec}(SK_{ID}, c)$.

The CCA security proof is super simple. The intuition?

- the decryption algorithm only uses SK_{ID} (and not the MSK per se) and
- the identity in the challenge ciphertext is random and hence different w.h.p. from the (adversarially chosen) identities in all the decryption queries.

Put together, IBE security should say that breaking the security of the challenge ciphertext is hard.

13 Registration-based Encryption

We will say just a few words about RBE here. Recall from the beginning of the lecture that a major disadvantage of IBE is the power of the master key authority to decrypt all ciphertexts.

A completely orthogonal approach which does not have this problem starts from the following strawman scheme: the master public key, curated by the authority, is the concatenation of all the users' public keys... Of course, this leads us back to exactly the PKI problem we wanted to solve. However, it is possible that the authority can publish a *short digest* of the concatenation of all public keys, which is nevertheless good enough for encryption (although it should not be clear exactly how yet!)

It turns out that this idea can be brought to fruition using the methodology of deferred encryption due to Garg et al. We refer the reader to the papers [?, ?]. The construction proceeds in a completely different way from everything we saw today, and is quite inefficient. An open problem is to come up with an RBE that is as efficient as (or more efficient than!) the IBE schemes we saw here.