CS 6501 Week 12: Lattice-Based Cryptography

Recall the inhomogeneous SIS problem: given $A \xleftarrow{R} \mathbb{Z}_q^{n \times m}$ and $u \xleftarrow{R} \mathbb{Z}_q^n$, find $x \in \mathbb{Z}_q^m$ such that $Ax = y$ and $\|x\| \le \beta$

It turns out that this can actually be used as a trapdoor function. Namely, there exist efficient algorithms

- $\mathrm{TrapGen}(n, m, q, \beta) \to (A, td_A)$: On input the lattice parameters n, m, q, the trapdoor-generation algorithm outputs a matrix $A \in \mathbb{Z}_q^{n \times m}$ and a trapdoor td_A
- $f_A(x) \to y$: On input $x \in \mathbb{Z}_q^m$, computes $y = Ax \in \mathbb{Z}_q^n$
- $f_A^{-1}(td_A, y) \to x$: On input the trapdoor td_A and an element $y \in \mathbb{Z}_q^n$, the inversion algorithm outputs a value $\|x\| \le \beta$

Moreover, for a suitable choice of n, m, q, β, these algorithms satisfy the following properties:

- For all $y \in \mathbb{Z}_q^n$, $f_A^{-1}(td_A, y)$ outputs $x \in \mathbb{Z}_q^n$ such that $\|x\| \le \beta$ and $Ax = y$
- The matrix A output by TrapGen is statistically close to uniform over $\mathbb{Z}_q^{n \times m}$

Lattice trapdoors have received significant amount of study and we will not have time to study it extensively. Here, we will sketch the high-level idea behind a very useful and versatile trapdoor known as a "gadget" trapdoor

First, we define the "gadget" matrix (there are actually many possible gadget matrices — here, we use a common one sometimes called the "powers-of-two" matrix):

$$G = \begin{pmatrix} 1 & 2 & 4 & 8 & \cdots & 2^{\lfloor \log q \rfloor} & & & & & & & & & \\ & & & & & & 1 & 2 & 4 & \cdots & 2^{\lfloor \log q \rfloor} & & & & \\ & & & & & & & & & & & \ddots & & & \\ & & & & & & & & & & & & 1 & 2 & 4 & \cdots & 2^{\lfloor \log q \rfloor} \end{pmatrix} = \left(1\ 2\ 4\ \cdots\ 2^{\lfloor \log q \rfloor}\right) \otimes I_n$$

Each row of G consists of the powers of two (up to $2^{\lfloor \log q \rfloor}$). Thus, $G \in \mathbb{Z}_q^{n \times n \lfloor \log q \rfloor}$. Oftentimes, we will just write $G \in \mathbb{Z}_q^{n \times m}$ where $m > n \lfloor \log q \rfloor$. Note that we can always pad G with all-zero columns to obtain the desired dimension.

<u>Observation</u>: SIS is easy with respect to G:

$$G \cdot \begin{pmatrix} 2 \\ -1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} = 0 \in \mathbb{Z}_q^n \quad \Rightarrow \text{ norm of this vector is } 2$$

Inhomogenous SIS is also easy with respect to G: take any target vector $y \in \mathbb{Z}_q^n$. Let $y_{i,\lfloor \log q \rfloor}, \dots, y_{i,1}$ be the binary decomposition of y_i (the i^{th} component of y). Then,

$$G \cdot \underbrace{\begin{pmatrix} y_{1,1} \\ y_{1,2} \\ \vdots \\ y_{1,\lfloor \log q \rfloor} \\ y_{2,1} \\ \vdots \\ y_{2,\lfloor \log q \rfloor} \\ \vdots \\ y_{n,1} \\ \vdots \\ y_{n,\lfloor \log q \rfloor} \end{pmatrix}}_{} = \begin{pmatrix} \sum_{j=1}^{\lfloor \log q \rfloor} 2^j y_{1,j} \\ \vdots \\ \sum_{j=1}^{\lfloor \log q \rfloor} 2^j y_{n,j} \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = y$$

Observe that this is a 0/1 vector (binary valued vector), so the ℓ_∞-norm is exactly 1

We will denote this "bit-decomposition" operation by the function $G^{-1} : \mathbb{Z}_q^n \to \{0,1\}^m$

important: G^{-1} is <u>not</u> a matrix (even though G is)!

Then, for all $y \in \mathbb{Z}_q^n$, $G \cdot G^{-1}(y) = y$ and $\|G^{-1}(y)\| = 1$. Thus, both SIS and inhomogeneous SIS are easy with respect to the matrix G.

We now have a matrix with a <u>public</u> trapdoor. To construct a <u>secret</u> trapdoor function (useful for cryptographic applications), we will "hide" the gadget matrix in the matrix A, and the trapdoor will be a "short" matrix (i.e., matrix with small entries) that recovers the gadget.

More precisely, a gadget trapdoor for a matrix $A \in \mathbb{Z}_q^{n \times k}$ is a short matrix $R \in \mathbb{Z}_q^{k \times m}$ such that

$$A \cdot R = G \in \mathbb{Z}_q^{n \times m}$$

We say that R is "short" if all values are small. [We will write $\|R\|$ to refer to the largest value in R].

Suppose we know $R \in \mathbb{Z}_q^{m \times m}$ such that $AR = G$. We can then define the inversion algorithm as follows:

- $f_A^{-1}(td_A = R,\ y \in \mathbb{Z}_q^n)$: Output $x = R \cdot G^{-1}(y)$.

We check two properties:

1. $Ax = AR \cdot G^{-1}(y) = G \cdot G^{-1}(y) = y$ so x is indeed a valid pre-image
2. $\|x\| = \|R \cdot G^{-1}(y)\| \leq m \cdot \|R\| \|G^{-1}(y)\| = m \cdot \|R\|$

   Thus, if $\|R\|$ is small, then $\|x\|$ is also small (think of β as a large polynomial in n).

<u>Important note</u>: When using trapdoor functions in a setting where the adversary can see trapdoor evaluations, we actually need to <u>randomize</u> the computation of f_A^{-1}. Otherwise, we <u>leak</u> the trapdoor. But this basic scheme illustrates the main ideas...

<u>Remaining question</u>: How do we generate A together with a trapdoor (and so that A is statistically close to uniform)?

Many techniques to do so; we will look at one approach using the LHL:

Sample $\bar{A} \xleftarrow{R} \mathbb{Z}_q^{n \times m}$ and $\bar{R} \xleftarrow{R} \{0,1\}^{m \times m}$.

Set $A = [\bar{A} \mid \bar{A}\bar{R} + G] \in \mathbb{Z}_q^{n \times 2m}$

Output $A \in \mathbb{Z}_q^{n \times 2m}$, $td_A = R = \begin{bmatrix} -\bar{R} \\ I \end{bmatrix} \in \mathbb{Z}_q^{2m \times m}$

First, we have by construction that $AR = -\bar{A}\bar{R} + \bar{A}\bar{R} + G = G$, and moreover $\|R\| = 1$. It suffices to argue that A is statistically close to uniform (without the trapdoor R). This boils down to showing that $\bar{A}\bar{R} + G$ is statistically close to uniform given $\bar{A}$. We appeal to the LHL:

1. From the previous lecture, the function $f_A(x) = Ax$ is pairwise independent.
2. Thus, by the LHL, if $m \geq 3n \log q$, then Ar is statistically close to uniform in $\mathbb{Z}_q^n$ when $r \xleftarrow{R} \{0,1\}^m$.
3. Claim now follows by a hybrid argument (applied to each column of R).

Thus, given $\bar{A}$, the matrix $\bar{A}\bar{R}$ is still statistically close to uniform. Corresponding, A is statistically close to uniform.

<u>Digital signatures from lattice trapdoors</u>: We can use lattice trapdoors to obtain a digital signature scheme in the random oracle model (this is essentially an analog of RSA signatures):

- $\text{KeyGen}(1^\lambda)$: $(A, td_A) \leftarrow \text{TrapGen}(n, m, q, \beta)$ [lattice parameters n, m, q, β are based on security parameter λ]

  Output $vk = A$ and $sk = td_A$
- $\text{Sign}(sk, m)$: Output $\sigma \leftarrow f_A^{-1}(td_A, H(m))$. Here, $H : \{0,1\}^* \to \mathbb{Z}_q^n$ is modeled as a random oracle.
- $\text{Verify}(vk, m, \sigma)$: Check that $\|\sigma\| \leq \beta$ and that $f_A(\sigma) = H(m)$.

Hardness reduces to hardness of inhomogeneous SIS (similar proof as RSA-FDH). Sketch:

1. Replace A with a uniformly random matrix (as required by inhomogeneous SIS) — follows by property of TrapGen
2. Given inhomogeneous SIS challenge (A, y), set public key to A and $H(m^*) = y$ where m^* is the message the adversary forges on (guess this at beginning)

3. To simulate signing queries on a message m (without knowledge of trapdoor), first sample $x \leftarrow D_s$ and sets $H(m) = Ax$
   - Here D_s corresponds to the distribution of vectors output by the preimage-sampling algorithm f_A^{-1} [this is typically a discrete Gaussian distribution with standard deviation s, where s is chosen so that Ax is statistically close to uniform over $\mathcal{L}(A)$]
   - Thus, by programming the random oracle, we can sign arbitrary messages <u>without</u> knowledge of the trapdoor for A

<u>Summary so far</u>: - The SIS problem can be used to realize many symmetric primitives such as OWFs, CRHFs, and signatures
- Useful trick: "Concealing" a trapdoor (e.g., short matrix/basis) within a random-looking one – common theme in lattice-based cryptography.

For public-key primitives, we will rely on a very similar assumption: learning with errors (LWE), which can also be viewed as a "dual" of SIS. We introduce the assumption below:

errors are typically much smaller than q

<u>Learning with Errors (LWE)</u>: The LWE problem is defined with respect to lattice parameters n, m, q, χ, where χ is an <u>error distribution</u> over $\mathbb{Z}_q$ (oftentimes, this is a discrete Gaussian distribution over $\mathbb{Z}_q$). The $LWE_{n,m,q,\chi}$ assumption states that for a random choice $A \xleftarrow{R} \mathbb{Z}_q^{n \times m}$, $s \xleftarrow{R} \mathbb{Z}_q^n$, $e \leftarrow \chi^m$, the following two distributions are computationally indistinguishable:

$$(A, s^T A + e^T) \overset{c}{\approx} (A, r)$$

where $r \xleftarrow{R} \mathbb{Z}_q^m$.

In words, the LWE assumption says that <u>noisy</u> linear combinations of a secret vector over $\mathbb{Z}_q^n$ looks indistinguishable from random.

A few notes/observations on LWE:
- Typically, m is sufficiently large so that the LWE secret s is <u>uniquely</u> determined.
- Without the error terms, this problem is easy for $m > n$: simply use Gaussian elimination to solve for s
- Observe that if SIS is easy, then LWE is easy. Namely, if the adversary can find a short $u \in \mathbb{Z}_q^m$ such that $Au = 0$, then, the adversary can compute

$$(s^T A + e^T) u = s^T A u + e^T u = e^T u \implies \|e^T u\| \leq m \cdot \|e\| \cdot \|u\|$$

this is small (compared to q)

$r^T u$ will be uniform over $\mathbb{Z}_q$, are unlikely to be small
- We can also choose the LWE secret from the error distribution (so it is short) – can be useful for both efficiency and for functionality (this is at least as hard as LWE with secrets drawn from <u>any</u> distribution, including the uniform one)
- Can also consider search vs. decision versions of the problem (i.e., search LWE says given $(A, s^T A + e^T)$, find s). There are search-to-decision reductions for LWE

<u>LWE as a lattice problem</u>: The search version of LWE essentially asks one to find s given $s^T A + e^T$. This can be viewed as solving the "bounded-distance decoding" (BDD) problem on the q-ary lattice

$$\mathcal{L}(A^T) = \{ s \in \mathbb{Z}_q^n : A^T s \} + q\mathbb{Z}^n$$

i.e., given a point that is close to a lattice element $s \in \mathcal{L}(A^T)$, find the point s

Connections to worst-case hardness: Regev showed that for any $m = \mathrm{poly}(n)$ and modulus $q < 2^{\mathrm{poly}(n)}$ and for a discrete Gaussian noise distribution (with values bounded by β), solving $\mathrm{LWE}_{n,m,q,\chi}$ is as hard as quantumly solving GapSVP_γ on arbitrary n-dimensional lattices with approximation factor $\gamma = \tilde{O}(n \cdot q/\beta)$

↳ Long sequence of subsequent works have shown classical reductions to worst-case lattice problems (for suitable instantiations of the parameters)

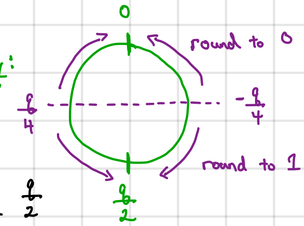
Symmetric encryption from LWE (for binary-valued messages)

Setup(1^λ): Sample $s \xleftarrow{R} \mathbb{Z}_q^n$.

Encrypt(s, μ): $\mu \in \{0,1\}$ Sample $a \xleftarrow{R} \mathbb{Z}_q^n$ and $e \leftarrow \chi$. Output $(a, s^T a + e + \mu \cdot \lfloor \frac{q}{2} \rfloor)$.

Decrypt(s, ct): Output $\underbrace{\lfloor ct_2 - s^T ct_1 \rceil_2}_{\text{"rounding operation"}}$

Visually:

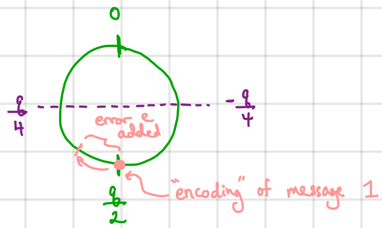


$$\lfloor x \rceil_2 = \begin{cases} 0 & \text{if } -\frac{q}{4} < x < \frac{q}{4} \\ 1 & \text{otherwise} \end{cases}$$

↳ take $x \in \mathbb{Z}_q$ to be representative between $-\frac{q}{2}$ and $\frac{q}{2}$

Correctness:
$$ct_2 - s^T ct_1 = s^T a + e + \mu \cdot \lfloor \tfrac{q}{2} \rfloor - s^T a$$
$$= \mu \cdot \lfloor \tfrac{q}{2} \rfloor + e$$

if $|e| < \frac{q}{4}$, then decryption recovers the correct bit



Security: By the LWE assumption, $(a, s^T a + e) \overset{c}{\approx} (a, r)$ where $r \xleftarrow{R} \mathbb{Z}_q$. Thus,

$$\underbrace{(a, s^T a + e)}_{\text{encryption of } 0} \overset{c}{\approx} \underbrace{(a, r) \equiv (a, r + \lfloor \tfrac{q}{2} \rfloor)}_{\text{since } r \text{ is uniform over } \mathbb{Z}_q} \overset{c}{\approx} \underbrace{(a, s^T a + e + \lfloor \tfrac{q}{2} \rfloor)}_{\text{encryption of } 1}$$

LWE   LWE

Observe: this encryption scheme is additively homomorphic (over $\mathbb{Z}_2$):

$$\begin{matrix} (a_1, & s^T a_1 + e_1 + \mu_1 \cdot \lfloor \frac{q}{2} \rfloor) \\ (a_2, & s^T a_2 + e_2 + \mu_2 \cdot \lfloor \frac{q}{2} \rfloor) \end{matrix} \Rightarrow \left(a_1 + a_2,\ s^T(a_1 + a_2) + (e_1 + e_2) + (\mu_1 + \mu_2) \cdot \lfloor \tfrac{q}{2} \rfloor\right)$$

decryption then computes

$$(\mu_1 + \mu_2) \cdot \lfloor \tfrac{q}{2} \rfloor + e_1 + e_2$$

which when rounded yields $\mu_1 + \mu_2 \pmod 2$ provided that $|e_1 + e_2 + 1| < \frac{q}{4}$

Using the results from HW3, we can obtain a public-key encryption scheme if we can "refresh" the ciphertexts

Idea: We will rely on the LHL. We will include encryptions of 0 in the public key and refresh ciphertexts by taking a subset sum of encryptions of 0:

Regev's encryption scheme

Setup(1^λ): $A \xleftarrow{R} \mathbb{Z}_q^{n \times m}$ output $pk = (A, b^T)$

$s \xleftarrow{R} \mathbb{Z}_q^n$ $b^T \leftarrow s^T A + e^T$ $sk = s$

$e \leftarrow \chi^m$ ↳ can be viewed as m encryptions of 0 under the symmetric scheme with secret key s

Encrypt(pk, μ): sample $r \xleftarrow{R} \{0,1\}^m$

output $(Ar, b^T r + \mu \cdot \lfloor \frac{q}{2} \rfloor)$

Decrypt(sk, ct): output $\lfloor ct_2 - s^T ct_1 \rceil_2$

Correctness:
$$ct_2 - s^T ct_1 = b^T r + \mu \cdot \lfloor \tfrac{q}{2} \rfloor - s^T A r = s^T A r + e^T r + \mu \cdot \lfloor \tfrac{q}{2} \rfloor - s^T A r$$
$$= \mu \cdot \lfloor \tfrac{q}{2} \rfloor + e^T r$$

if $|e^T r| < \frac{q}{4}$, then decryption succeeds (since e is small and r is binary, $e^T r$ is not large: $|e^T r| < m \|e\| \|r\| = m \|e\|$)

Security: Follows by LWE and LHL:

Hyb$_0$: Real public key

LWE

Hyb$_1$: Uniformly random public key (e.g. $b \xleftarrow{R} \mathbb{Z}_q^m$)

Hyb$_2$: Uniformly random ciphertext (e.g., $ct = (u, t)$ where $u \xleftarrow{R} \mathbb{Z}_q^m$ and $t \xleftarrow{R} \{0,1\}$)

LHL: $(\bar{A}, \bar{A}r) \overset{s}{\approx} (\bar{A}, u)$ where $\bar{A} = \left[\frac{A}{b^T}\right] \xleftarrow{R} \mathbb{Z}_q^{(n+1)\times m}$, $r \leftarrow \{0,1\}^m$, and $u \xleftarrow{R} \{0,1\}$

Encrypting multiple bits: May seem wasteful to use a vector to encrypt a <u>single</u> bit. We can consider a simple variant of Regev encryption where we <u>reuse</u> A to encrypt multiple bits:

<u>Setup$(1^\lambda, 1^\ell)$</u>: sample $A \xleftarrow{R} \mathbb{Z}_q^{n\times m}$

$S \xleftarrow{R} \mathbb{Z}_q^{n\times \ell}$ $\quad B^T \leftarrow S^T A + E^T \in \mathbb{Z}_q^{\ell\times m}$

$E \xleftarrow{R} \chi^{m\times \ell}$

ℓ secret keys concatenated together

pk: (A, B^T)

sk: S

<u>Encrypt$(pk, \mu \in \{0,1\}^\ell)$</u>: sample $r \xleftarrow{R} \{0,1\}^m$

output $(Ar, B^T r + \mu \cdot \lfloor \frac{q}{2} \rfloor)$

<u>Decrypt(sk, ct)</u>: output $\lfloor ct_2 - S^T ct_1 \rceil_2$

Correctness: As before: $ct_2 - S^T ct_1 = B^T r + \mu \cdot \lfloor \frac{q}{2} \rfloor - S^T A r = E^T r + \mu \cdot \lfloor \frac{q}{2} \rfloor$

Security: As before: by LWE, $(A, S^T A + E^T) \overset{c}{\approx} (A, R)$ where $A \xleftarrow{R} \mathbb{Z}_q^{n\times m}$, $S \xleftarrow{R} \mathbb{Z}_q^{n\times \ell}$, $E \leftarrow \chi^{m\times \ell}$, $R \xleftarrow{R} \mathbb{Z}_q^{\ell\times m}$

in particular, apply a hybrid argument and argue for each row of S (and corresponding row of $S^T A + E^T$)