Public Key Encryption

So far, we have focused on symmetric key encryption, where both parties have access to some k

- Now, we focus on when they do not

Public Key Encryption

A **public key encryption scheme** consists of three PPT algorithms:

- `Gen`: takes as input λ and outputs `(pk, sk)`
- `Enc`: takes as input `pk` and `m` and outputs a ciphertext `ct`
- `Dec`: takes as input `sk` and `ct` and outputs message `m`

The correctness guarantee is that the probability that `Dec(Enc(m, pk), sk) = m` is 1

Semantically Secure

A public key encryption scheme is **semantically secure** if for all λ and pair of messages m_0, m_1 we have:

$$(\text{pk}, \text{Enc}(\text{pk}, m_0)) \approx (\text{pk}, \text{Enc}(\text{pk}, m_1))$$

Note that adversaries have access to a lot of ciphertexts because they have the public key

Typically, classes will start with:

- El-Gamal
- RSA

But these are broken by quantum computers (rely on discrete log problem or factoring problem being hard)

- We focus on post-quantum encryption

Learning with Errors Assumption

Loosely speaking, the LWE assumption is that on a finite field, it is hard to solve noisy linear equations

LWE Assumption

The **LWE assumption** asserts that:

$$(A, sA + e) \approx (A, U)$$

where A is a $n \times m$ matrix, s is a n dim vector, e is a n dim error term sampled from some distribution χ, and U is the set of all possible n dim vectors

The idea is that we are trying to recover s and we are given a bunch of approximate linear equations relating the elements of s

- There are cases where this is clearly true (when e is uniform across all elements in the field) or we have $m < n$
- There are cases where this is clearly false (when e is 0)
- One useful case where we think this is true is when:
  - $n(\lambda) = \lambda$
  - $m(\lambda) = \text{poly}(\lambda)$
  - $q(\lambda) = \text{poly}(\lambda)$
  - χ is a small normal distribution
    - We often use a discrete Gaussian distribution restricted to $[-B, B]$
    - If q is the size of the field (i.e. if we're in $GF[q]$ which is the numbers under addition / multiplication modulo q), then we have:
      - $-B = q - B$

- $B << q$

LWE Symmetric Encryption Scheme

For encrypting messages b that are a single bit with a secret key s of length n:

$$\text{Enc}(s,b) = (a, s\cdot a + e + b\lfloor q/2 \rfloor)$$

where $a \xleftarrow{R} \mathbb{Z}_q^n$.

To decrypt messages (a,c):

$$\text{Dec}(s,(a,c)) = 0 \text{ iff } |c - s\cdot a| \leq q/4$$

- We assume e is small relative to $q/4$ so it cannot impact the answer

This scheme is **linearly homomorphic** which means for all messages b_1, b_2:

$$\text{Dec}(s, \text{Enc}(s,b_1) + \text{Enc}(s,b_2)) = b_1 \oplus b_2$$

- Intuitively, this means that we can add ciphertexts and then decrypt to get back the "sum" of the original messages
- However, the error is going to add up as we add more messages together
  - If the error term is small relative to q, we should still be able to do a lot

Proving CPA-Secure:

- The LWE assumption says that $s\cdot a + e$ cannot be distinguished, so adding a $b\lfloor q/2 \rfloor$ to it will still not be distinguishable from random data

LWE Public Encryption Scheme

We also have a public key encryption scheme for encrypting a single bit b

- $\text{Gen}(\lambda)$:
  - Let $n = \lambda$, $q = \text{poly}(\lambda)$, and $m = \Theta(n\cdot\log q)$
  - Let χ be a discrete Gaussian distribution with $B << q/4$
  - We generate $s \xrightarrow{R} \mathbb{Z}_q^n$ and $A \xrightarrow{R} \mathbb{Z}_q^{n\times m}$
  - Output

  $$\text{pk} = (A, sA + e)$$

  and $\text{s} = s$
    - We can think of the public key as a matrix B where the first n rows are A and the last row is $sA + e$
    - Note that, $(-s, 1)\cdot B = e$
- $\text{Enc}(\text{pk}, b)$:
  - Chooses a random $r \xrightarrow{R} \{0,1\}^m$
  - Outputs $B\cdot r + b * (0, \ldots, 0, \lfloor q/2 \rfloor)$
- $\text{Dec}(s, c)$:
  - Outputs 0 iff $|(-s,1)\cdot c| \leq q/4$

Correctness:

- We note that $(-s,1)\cdot(B\cdot r) = e\cdot r \leq mB$
  - The last equation follows from the fact that each element in e is at most B
  - Each element in r is 0 or 1, so we can at most have $m * B$
  - Then, mB is at most $q/4$ from our setup
- We then have that multiplying our ciphertext by $(-s,1)$ will give just:

$$mB + b * (0, \ldots, 0, \lfloor q/2 \rfloor)$$

CPA-Secure:

- Omited