Lecture 21: Private-key Encryption of Long Messages
Recall: One-time Pad

- One-time Pad was the most efficient technique to encrypt messages (Refer to Lecture 09). Any private-key encryption scheme must have secret-key that is as long as the secret-key of the one-time pad.
- It is secure even against adversaries with unbounded computation power.
- However, we need to know the length of the message that Alice wants to send to Bob. The length of the secret-key is as long as the length of the message.
Recall: One-time Pad for $n$-bit Messages

- Yesterday, Alice and Bob met to generate $sk \leftarrow \$ \{0, 1\}^n$
- Today Alice encrypts a message $m \in \{0, 1\}^n$ by computing the cipher-text $c = m \oplus sk$
- Bob can decrypt the cipher text $c$ by computing $\tilde{m} = c \oplus sk$
Last lecture we saw that if $f$ is a one-way permutation
then, using the Goldreich-Levin Hardcore Predicate, we can construct a one-bit extension pseudo-random generator $G_{n,n+1}$, where $n$ is even, using the following construction

$$G_{n,n+1}(r, x) = (r, f(x), \langle r, x \rangle),$$

where $r, x \in \{0, 1\}^{n/2}$

Given a one-bit extension PRG, we can construct arbitrary stretch pseudo-random generator $G_{n,\ell}: \{0, 1\}^n \rightarrow \{0, 1\}^\ell$
Goal

- Suppose Alice and Bob met yesterday to establish an $n$-bit secret-key
- Today we want Alice to encrypt an $\ell$-bit message, where $\ell$ is much larger than $n$ (say, $\ell = n^2$)
Instead of using a random sk in the one-time encryption we shall use a pseudorandom sk (produced from a small seed)

Gain: We shall encrypt messages that are much larger than the length of the seed

Loss: The encryption is secure only against computationally bounded adversaries
Private-key Encryption Scheme

- **Gen()**: Return $sk \leftarrow \{0, 1\}^n$ (the seed for the PRG)
- **Enc$_{sk}(m)$**: Return $c = m \oplus G_{n,\ell}(sk)$, where $\ell$ is the length of the message $m$ and $G_{n,\ell}(m)$ is a PRG
- **Dec$_{sk}(c)$**: Return $\tilde{m} = c \oplus G_{n,\ell}(sk)$

Intuition:

- Instead of the mask being a random $\ell$-bit string, we use the pseudo-random mask $G_{n,\ell}(sk)$
- Note that $\ell$ can be deduced by Bob from the length of the cipher-text, so he can compute $G_{n,\ell}$
- The scheme is secure for arbitrarily $\ell$ that is polynomial in $n$ (i.e., $\ell$ need not be known while choosing the secret key)
- A larger polynomial $\ell$ reduces the security of the scheme
How can Alice encrypt and send a second message $m'$ of length $\ell'$ tomorrow? What does Alice need to remember from today to successfully perform this encryption tomorrow?