Can we use the Pseudo Random Function (PRF), discussed in last lecture, to build Symmetric Encryption (SE) schemes which have IND property?

## 1 Construction of SE scheme using PRF

A PRF accepts messages, $m$, of length $l$ and outputs $F_k(m)$. We need to use this PRF to generate the encryption scheme.

### 1.1 Possibility-1:

We want the property that given a message $m$, its cipher text should look random. We want to have a correlation between a message and its cipher, but it should be weak. Since PRF satisfies this property, we can use it directly.

The SE scheme looks like:

Key Generation: $k \xleftarrow{\$} KeyGen()$

Encryption: $c = F_k(m)$

Decryption: $m = F_k^{-1}(c)$, but $F_k^{-1}(c)$ does not exist

There are two reasons why using PRF directly does not work.

1. Decryption should be possible for any encryption scheme, but inverse is not defined for a PRF. This is because - two messages $m_1, m_2$ might get mapped to the same $y_i$.
2. Even if decryption was possible, the resulting SE is deterministic because the same message would result in the same cipher text. Any deterministic scheme is not secure against IND-CPA.
1.2 Possibility - II:

We desire an encryption method with inherent randomization and accompanied by a corresponding decryption method. The perfect secure cipher One Time Pad (OTP), produces the cipher by XORing the message with the key and decrypts the cipher by XORing the cipher with the key.

\[
c = m \oplus k \\
m = c \oplus k
\]

Taking the cue from this, we can follow a similar approach where the message is XORed with an output of a PRF, which introduces the random nature, and since the message is outside PRF, we also don't require the PRF\(^{-1}\) to get back the message from the cipher.

The SE scheme is as follows:

Key Generation: \( k \xleftarrow{\$} \text{KeyGen}() \)

Encryption: Pick a random number \( r \) from \( \{0,1\}^l \).

\[
r \xleftarrow{\$} \{0,1\}^l \\
c = m \oplus F_k(r) \\
\text{Output:} \ (r,c)
\]

Decryption: \( m = c \oplus F_k(r) \)

Even if \( r \) is made public, since \( k \) is not known, \( F_k(r) \) cannot be computed. Since \( r \) is chosen at random, the resulting encryption scheme is randomized and the same message does not result in the same cipher text each time it is encrypted.

If message, \( m \), is longer than \( l \) bits, we break it into \( l \) length blocks \((m_1, m_2, m_3, ..., m_n)\). For encrypting each of the \( i \)th message block we need a random number \( r_i \), giving cipher text \( c_i \).

Earlier only \( c \) was the exchanged between the participants, but now even \( r \) needs to be sent along with \( c \). So the final cipher text is going to be \((r_1, r_2, ..., r_n, c_1, c_2, ..., c_n)\). As
the size of transmitted text gets doubled, this method is inefficient though it is a good encryption scheme otherwise, secure against IND-CPA.

Instead of considering a separate \( r \) for encrypting each block, we might think to repeat \( r \) to reduce the length of transmitted text. But it would again give rise to deterministic encryption (same \( r \), same \( m \Rightarrow \) same \( c \)). This proposed SE is not scalable.

Now we define another primitive for which inverse exists.

## 2 Random Permutation

Definition: A mapping permutation picked at random from the family of permutations, from domain \{0, 1\}^l to range \{0, 1\}^l.

It takes messages \( m_i \), picks permutations \( y_i \) at random from the range. It maintains a table of mappings and doesn’t repeat the outputs. It is deterministic. Once we map \( m_1 \) to \( y_1 \), \( y_1 \) is removed from the set of available outputs for other \( m_i \). This allows for defining an inverse RP.

Earlier, in the case of RF, even after \( m_1 \) was mapped to output \( y_1 \), for determining the mapping for \( m_2 \) we consider the whole range again. This allows some possibility for two \( m_i \)'s to be mapped to the same output \( y_k \), making it difficult for having a inverse RF.

3 Pseudo Random Permutation

Definition: A family of permutations for a given key \( k \), picked at random - \( k \xleftarrow{\$} \mathcal{K} \), \( \pi_k \) is called a Pseudo Random Permutation (PRP) if it is a close enough approximation to the random permutations.
We have two experiments, world one has PRP-1 \((\pi_k)\) and world zero has PRP-0 (RP). The adversary looking at the \(m_i - y_i\) pairs, need to guess if it is world zero or world one. The advantage of the adversary is:

\[
ADV^{PRP}(A) = Pr(EXP^{PRP-1}(A) = 1) - Pr(EXP^{PRP-0}(A) = 1)
\]

If \(\forall A, ADV^{PRP}(A) < \epsilon\), then it is a good PRP.

We want to have a construction for SE using this PRP.

### 3.1 Construction of SE using PRP:

We have \(\pi_k\), a PRP, with us and we need to use it to generate a SE.

#### 3.1.1 Possibility - I:

Using the PRP as it is.

Key Generation: \(k \xleftarrow{s} KeyGen()\)

Encryption: \(c = \pi_k(m)\)

Decryption: \(m = \pi_k^{-1}(c)\)

Since PRP is a permutation, inverse exists, and the correctness property holds.

But it is not a good encryption scheme as it is deterministic. If there is a long message \(M\), we break it into message blocks \(m_i\), each of length \(l\). If two messages \(m_1, m_2\) are same, their cipher texts \(c_1\) and \(c_2\) are also same. Hence this SE scheme is not secure against IND-CPA.

This model is not even secure against IND. Consider two messages \(m_0\) and \(m_1\) each is of \(2l\) bits long. \(m_0\) has all 0s and \(m_1\) has initial \(l\) bits as 0s and the following \(l\) bits as 1s. In the IND setting, if these two messages are sent to the challenger, then depending on how the cipher text looks - both the \(l\) bit cipher text blocks looking same or different, we can surely identify which message was encrypted. IND-CPA property is not necessary in some scenarios, but IND is needed everywhere. Hence using \(\pi_k\) as it is, is not sufficient.

#### 3.1.2 Possibility - II:

We use PRP in Chained Block Cipher (CBC) Mode. A random number, called Initialization Vector (IV), denoted by \(c_0\), is XORed with the first message block and the result
is passed as input to the PRP. The resulting cipher text is chosen as IV for encrypting the following message, and the procedure follows.

Since a new IV is chosen for encrypting every new message M, the same message encrypted twice does not result in the same cipher text.

The new SE scheme:

Key Generation: \( k \leftarrow \text{KeyGen}() \)

Encryption: \( c_0 \leftarrow \{0, 1\}^l \), pick \( c_0 \) at random,

\[
c_i = \pi_k(m_i \oplus c_{i-1}); \text{ for } i = 1 \text{ to } n
\]

output: \( (c_0, c_1, c_2, \ldots, c_n) \)

Decryption: \( m_i = c_{i-1} \oplus \pi_k^{-1}(c_i); \text{ for } i = 1 \text{ to } n \)

Here \( c_0 \) is not kept as secret, it is transmitted along with other cipher texts. The secret is only the key \( k \).

### 3.2 Instantiation of PRP:

Block Ciphers like DES, Triple DES and AES are believed to be the practical instantiations of PRP. By considering them as such we proceed to prove other properties. (Please refer to the slides on DES).

We proceed next to proving the properties of CBC-PRP.

*Theorem:* CBC instantiated with PRP (CBC-PRP) is IND-CPA.

The proof of this theorem shall be discussed in next lecture.