So far, we have examined PRBG (Pseudo Random Bit Generator) along with its two ingredients, one-way permutation and hard core bits. Secure PRBG means that it generates pseudo random bits which is indistinguishable from truly random bits. We have also studied PRF (Pseudo Random Function) and here, security means that it is indistinguishable whether PRF or random function is used to produce outputs for certain inputs. Now, we are ready to discuss PRPs (Pseudo Random Permutations) which can be constructed from PRFs. In this lecture, we will go over number of properties of PRPs and the construction of them.

1. Properties of PRPs

- A PRP is "basically" a PRF and a permutation.
  - (a) $\{F_k\}_{k \in \text{Keyspace}}$, where $F_k : \{0,1\}^l \rightarrow \{0,1\}^l$
  - (b) It is easy to evaluate both $F_k$ and $F_k^{-1}$ with given $k$.
  - (c) For a randomly chosen $k$, it is hard to distinguish $F_k$ from a random permutation on $l$-bits.

Adversary submits queries, $x_1, x_2, ..., x_q$ to an oracle $O$, whether it be $F_k$ or $\Pi$, obtaining $O(x_i)$ for each $i = 1, 2, ..., q$. Here, security means distinguishing advantage is negligible for all efficient adversaries.

- Notion of a a strong PRP - all properties are the same with the ones of a standard PRP except that the adversary has an access to $O$ and $O^{-1}$, where $O = F_k$ or $O = \Pi$.

- What does the previous statement that a PRP is "basically" a PRF mean? Is a PRP a PRF? The answer is it depends on the size of $l$. If $l$ is sufficiently small, then it is distinguishable between a random function and a random permutation with $2^l$ queries, and therefore, a PRP is not a PRF. However, for a sufficiently

L10-1
large \( l \), a random function cannot be distinguished from a random permutation, that is a PRP is a PRF. And here is the proof.

Proof. (Assume adversary makes \( q \) distinct queries.)

- Game 0: The adversary talks to \( \Pi \).
- Game 1: A little "gnome" is in a box for a consistency check. At query \( i \) at \( x_i \), he chooses \( y_i \in_R \{0, 1\}^l \). Then, he compared \( \tilde{y}_i \) to \( \tilde{y}_j \) for \( j = 1, \ldots, i - 1 \). If there’s a match, then he sets \( \tilde{y}_i \leftarrow R \{0, 1\}^l \setminus \{y_1, y_2, \ldots, y_{i-1}\} \), else, he sets \( \tilde{y}_i \leftarrow y_i \). He then outputs \( y_i \).
- Game 2: It is the same as game 1, but always set \( \tilde{y}_i \leftarrow y_i \).

Now, let us define \( S_i \) as the event that the adversary outputs 1 in game \( i \). Then,

\[
Pr[S_0] = Pr[S_1].
\]

Let’s also define event, \( F \) to be the event that some \( y_i \)'s are the same, such that

\[
S_1 \lor \neg F \iff S_2 \lor \neg F.
\]

Then, by Difference Lemma,

\[
|Pr[S_1] - Pr[S_2]| \leq Pr[F] \\
\leq \frac{q(q-1)2^{-l}}{2^q} \\
= O(\frac{q^2}{2^q}).
\]

Now, game 2 is exactly the same as the adversary talking to \( f \). Therefore, any adversary’s advantage in distinguishing a random permutation from a random function is \( O(\frac{q^2}{2^q}) \). Then, for an efficiently large, \( l \), the distinguishing advantage is negligible and a PRP is a PRF. \( \square \)

2. Construction of PRP from PRF

L10-2
• Feistel-network

Feistel-network is built by running "round" function, $H$, several times. The "round" function is defined as

$$H_f(L, R) = (R, f(R) \oplus L)$$

where $L, R \in \{0, 1\}^{l/2}$ and $f$ is a keyed function, and for our purpose, it is a pseudo random function. Specifically, $k$-round Feistel network is built by running $H$ $k$ times as illustrated in 1. As mentioned previously, $f^{(1)}, f^{(2)}, \ldots, f^{(k)}$ are keyed functions.

![Figure 1: k-round Feistel-network](image)

Is Feistel-network a permutation? It is one-to-one by the reasoning that if $(R, f(R) \oplus L) = (R', f(R') \oplus L')$, then $R = R'$ and $L = L'$. Clearly, it is also effectively invertible given keys, $f^{(1)}, f^{(2)}, \ldots, f^{(k)}$. Therefore, it is a permutation.

We are now ready to construct a PRP from 3-round Feistel-network with pseudo-random keyed functions via Ludy-Rackoff Theorem.

**Theorem 1. Ludy-Rackoff**

3-round Feistel-network with pseudo-random $f^{(1)}, f^{(2)}$, and $f^{(3)}$ is a PRP assuming $2^l$ is super-poly.

**Proof.** We will prove the theorem by showing that random permutation is indistinguishable from a random function. Let $(L_i, R_i)$ for $i = 1, 2, \ldots, q$ be the distinct queries made by the adversary. Here, we will use notations, $f, g,$ and $h$ instead of $f^{(1)}, f^{(2)},$ and $f^{(3)}$. Also, let

$$U_i = L_i \oplus f(R_i),$$

L10-3
\[ V_i = R_i \oplus g(U_i), \]
and
\[ W_i = U_i \oplus h(V_i). \]

And, the output is \((V_i, W_i)\). Let \(S_i\) be the event that output is 1 in Game \(i\).

- **Game 0**: The original game against Feiste-network with pseudo-random \(f, g,\) and \(h\).
- **Game 1**: Let \(f, g,\) and \(h\) be truly random functions instead of pseudo-random ones.
  \[ |Pr[S_0] - Pr[S_1]| \leq \epsilon_{PRF}, \]
  where \(\epsilon_{PRF}\) is negligible by the same hybrid arguments used in previous lecture’s Stretching Theorem to prove that PRBG is indistinguishable from truly random bit generator.
- **Game 2**: Replace \(g\) by a random ”responder”, that is \(g\) always outputs random bit strings even if the input is duplicated.
  * **Game 2a**: Keep a list of \(q\) random outputs with maintaining consistency. Essentially, game 1 and game 2a are the same and
  \[ Pr[S_{2a}] = Pr[S_1]. \]

  * **Game 2b**: Drop the consistency check. Let \(F\) be the event that some \(U_i\)’s collide in game 2b. Then, game 2a and game 2b are the same as long as \(F\) does not occur, that is, by Difference lemma,
  \[ |Pr[S_{2a}] - Pr[S_{2b}]| \leq Pr[F]. \]

  Now, what’s \(Pr[L_i \oplus f(R_i) = L_j \oplus f(R_j)]\) for \(i \neq j\)
  (a) Case 1: If \(R_i = R_j\), then \(L_i \neq L_j\) and \(f(R_i) = f(R_j)\). Therefore, such a \(Pr = 0\)
  (b) Case 2: If \(R_i \neq R_j\), then using ”Deferred Gratification” technique, the evaluation of \(Pr[F]\) will be held off and meanwhile we will continue with the proof, and will come back to resolve this case.
- **Game 3**: Replace all \(V_i\)’s by random independent bit strings. This replacement does not change the distribution. Let \(F'\) be the event that some \(U_i\)’s collide in game 3. Then,
  \[ Pr[S_2] = Pr[S_3] \]
  , and
  \[ Pr[F] = Pr[F']. \]

The transition from game 2 to game 3 is really only a conceptual change which rely on the fact that the adversary doesn’t have direct access to \(g\) oracle or the inside information of Feistel-network.

L10-4
- Game 4: Replace $h$ by a random "responder".
  * Game 4a: As it was done for game 2a, keep a list of $q$ random outputs with maintaining consistency. Since game 4a is basically game 3,
    \[ Pr[S_{4a}] = Pr[S_3] \]
  * Game 4b: Likewise for game 2b, drop the consistency check. Let $G$ be the event that in game 4b, some $V_i$’s collide. Then,
    \[ |Pr[S_{4a}] - Pr[S_{4b}]| \leq Pr[G] \]
    , and
    \[ |Pr[S_3] - Pr[S_4]| \leq Pr[G] \]
  since $Pr[S_{4a}] = Pr[S_3]$, and $Pr[S_{4b}] = Pr[S_4]$. Then if we define $F''$ as the event that $U_i$’s are the same in game 4, it corresponds to $F'$ in game 3, and
    \[ |Pr[F''] - Pr[F']| \leq Pr[G]. \]
  But, since $V_i$’s are independent, by Birthday Paradox argument,
    \[ Pr[G] = O\left(\frac{q^2}{2^{l/2}}\right). \]
- Game 5: Replace all $W_i$’s by random independent bit strings. Let $F'''$ be the event that some $U_i$’s collide in game 5. As in game 3, the transition from game 4 to game 5 is really a conceptual change which rely on the fact that the adversary does not have direct access to $h$ oracle. Then,
    \[ Pr[S_5] = Pr[S_4] \]
    , and
    \[ Pr[F'''] = Pr[F'']. \]
  Now, as of game 5, the original Feistel-network has transformed in the form
    \[ U_i = L_i \oplus f(R_i), \]
    \[ V_i \xleftarrow{R} \{0,1\}^{l/2}, \]
    \[ W_i \xleftarrow{R} \{0,1\}^{l/2}. \]
  And game 5 cannot be distinguished from the original Feistel-network. In another words, now that the adversary has no control over affecting the output since $f$ is not known and $V$ and $W$ are truly random, the problem is equivalent as to analyzing the game as if the adversary chooses distinct input $(L_1, R_1), (L_2, R_2), ..., (L_q, R_q)$ and chooses a random function, $f$, and try to figure out if $f(L_i, R_i) = f(L_j, R_j)$. Then, it draws us to the previous unsolved
problem of evaluating $Pr[F]$. What is $Pr[L_i \oplus f(R_i) = L_j \oplus f(R_j)]$ for some
$i \neq j$? As previously described, such a probability is zero if $R_i = R_j$. If
$R_i \neq R_j$, $f(R_i) \oplus f(R_j) = L_i \oplus L_j$ with the probability of $\frac{1}{2^l}$. When you
sum over all pairs of $(L_i, R_i)$ for all $i = 1, 2, ..., q$, the probability is $O(\frac{q^2}{2^l})$.
(Remark: it is enough to assume that $f$ is pairwise independent.)

Therefore, in conclusion, 3-round Feistel-network with pseudo-random func-
tions, $f, h$, and $g$ a PRP as long as two pairs of $(L_i, R_i)$ do not collide which
happens with the probability $O(\frac{q^2}{2^l})$ which is negligible with the sufficiently
large $l$.

\(\square\)

3. **Constructing PRP in practice** There are two kinds of PRP’s in practice, DES,
which is 16-round Feistel-network, and AES. We will study more on the topic of PRP’s
application in the next lecture.