How to build Pseudorandom Permutations?: Luby-Rackoff's Construction

Debdeep Mukhopadhyay
IIT Kharagpur

Pseudo-random permutation

- A pseudorandom function is an efficient function, F: $\{0,1\}^k x \{0,1\}^n \rightarrow \{0,1\}^n$, such that no efficient algorithm A, can distinguish $F_K(.)$ from R(.) for a randomly chosen key $K \leftarrow \{0,1\}^n$ and a random function $R:\{0,1\}^n \rightarrow \{0,1\}^n$.
- This implies:

$$\boxed{A^{F_K(.)} \text{ behaves like } A^{R(.)}}$$

Pseudorandom Permutation

- It is also a permutation.
- Moreover there exists an efficient inverse, P_K^{-1}.
- A pseudorandom permutation is also a pseudorandom function.
- Strong pseudorandom permutation: No efficient algorithm A can distinguish well between <$P_K(.)$,$P_K^{-1}(.)$> from <$\Pi(.)$, $\Pi^{-1}(.)$> for a randomly chosen key and random permutation, Π.

$$\boxed{A^{P_K(.),P_K^{-1}} \text{ behaves like } A^{\Pi(.),\Pi^{-1}(.)}}$$

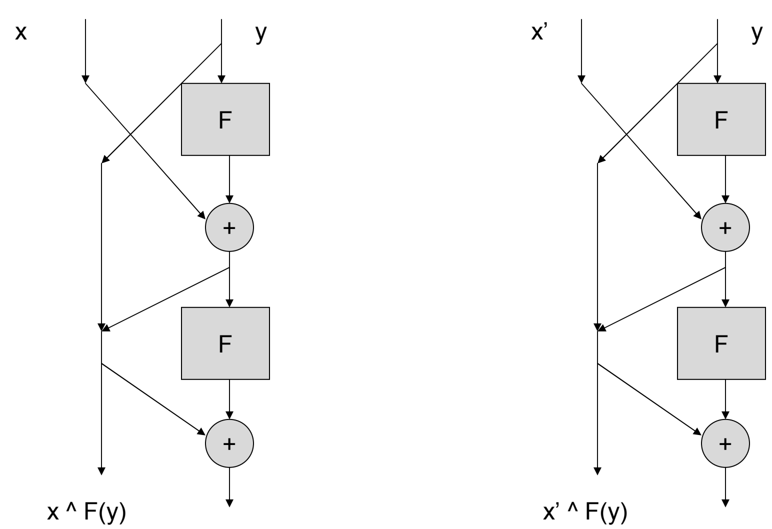
Building Pseudorandom Permutations

- We can build pseudorandom permutations from pseudorandom functions, F
- Define $\boxed{D_F(x,y) = y, F(y) \oplus x}$
- Note that this is injective and that does not depend whether F is injective or not.
- Note that D_F and D_F^{-1} are efficiently computable.
- This construction was originally due to Horst Feistel.

Is one round Pseudorandom

- No.
- Note that the output contains the right half of the input.
- This is extremely unlikely in case of a random permutation.
- So, does two rounds work?

Two Feistel Rounds



3 Rounds of DES

- 3 rounds of DES is also not pseudo-random permutation in the strong sense.
- But 4 round DES is a strong pseudorandom permutation.

Proof

Define $P_K = D_{F_{k_4}}(D_{F_{k_3}}(D_{F_{k_2}}(D_{F_{k_1}}(x))))$. Given 4 random functions,
$R = < R_1, \ldots, R_4 >$, $R_i : \{0,1\}^m \rightarrow \{0,1\}^m$.
Let, $P_R(x) = D_{R_4}(D_{R_3}(D_{R_2}(D_{R_1}(x))))$
First let us reason that: P_K and P_R are indistinguishable, as otherwise F is not pseudorandom.

Proof:

$|\Pr[A^{P_K, P_K^{-1}}() = 1] - \Pr[A^{P_R, P_R^{-1}}() = 1]| \leq 4\varepsilon$

The proof is using a hybrid argument.

Consider the following five algorithms from $\{0,1\}^{2m} \to \{0,1\}^{2m}$:

H_0 : pick random keys K_1, K_2, K_3, K_4

$H_0(.) = D_{F_{K_4}}(D_{F_{K_3}}(D_{F_{K_2}}(D_{F_{K_1}}(.))))$

H_1 : pick random keys K_2, K_3, K_4 and a random function $F_1 : \{0,1\}^m \to \{0,1\}^m$

$H_1(.) = D_{F_{K_4}}(D_{F_{K_3}}(D_{F_{K_2}}(D_{F_1}(.))))$

H_2 : pick random keys K_3, K_4 and random functions F_1 and $F_2 : \{0,1\}^m \to \{0,1\}^m$

$H_2(.) = D_{F_{K_4}}(D_{F_{K_3}}(D_{F_2}(D_{F_1}(.))))$

H_3 : pick random keys K_4 and random functions $F_1, F_2, F_3 : \{0,1\}^m \to \{0,1\}^m$

$H_3(.) = D_{F_{K_4}}(D_{F_3}(D_{F_2}(D_{F_1}(.))))$

H_4 : pick random functions $F_1, F_2, F_3, F_4 : \{0,1\}^m \to \{0,1\}^m$

$H_4(.) = D_{F_4}(D_{F_3}(D_{F_2}(D_{F_1}(.))))$

Clearly H_0 gives the first probability of using all pseudorandom and H_4 gives the construction using all random functions.

Hence, we know there exists an i for which:

$|\Pr[A^{H_i, H_i^{-1}} = 1] - \Pr[A^{H_{i+1}, H_{i+1}^{-1}} = 1]| > \varepsilon$

Define an algorithm A' using A as follows:

On the first i layers A' picks keys $K_1, \ldots, K_i$.

A' runs the pseudorandom function F using the i keys $K_1, K_2, \ldots, K_i$

On the ith layer, the oracle G is run.

For the remaining layers a random function is run.

Thus, A' operates on G and has to decide whether G is pseudorandom or random.

Note that when G is pseudorandom we have A'^{G} behaving exactly same as $A^{H_i, H_i^{-1}}$.

When G is a random function, A'^{G} behaves exactly like $A^{H_{i+1}, H_{i+1}^{-1}}$.

Thus, we have:

$|\Pr_K[A'^{F_k()} = 1] - \Pr_R[A'^{R(.)} = 1]| > \varepsilon,$

which contradicts that F is pseudorandom.

Next Step…

$$\Pr[A^{P_R, P_R^{-1}}() = 1] - \Pr[A^{\Pi, \Pi^{-1}}() = 1] \le \frac{t^2}{2^{2m}} + \frac{t^2}{2^m}$$

where $\Pi: \{0,1\}^{2m} \rightarrow \{0,1\}^{2m}$ is a random permutation.

Assume that the algorithm A is non-repeating.

Introduce one more experiment S(A) that simulates A and simulates every oracle query by providing a random answer.

[Note that the simulated answer from S() may be INCONSISTENT with a truly random permutation]

Let A be a non-repeating algorithm of complexity at most t queries. Then

$|\Pr[S(A)=1]-\Pr[A^{\Pi,\Pi^{-1}}()=1] \le \frac{t^2}{2^{2m+1}}$

Define a transcript a record of all oracle queries, $<(x_1, y_1),...(x_t, y_t)>$. The output of the algorithm is purely a function of the transcript.

Define consistent transcript T to be such that

$x_i = x_j \Leftrightarrow y_i = y_j$.

Consistent Transcripts

Also note that if the transcript is consistent, then

Pr[Tr(S)=σ|Tr(S) is consistent]

$$=\frac{2^{-2mt}}{1(1-\frac{1}{2^{2m}})...(1-\frac{t-1}{2^{2m}})}=\frac{(2^{2m}-t)!}{2^{2m}!}$$

$$\Pr[Tr(A^{\Pi,\Pi^{-1}})=\sigma]=\frac{1}{2^{2m}}\frac{1}{(2^{2m}-1)}\cdots\frac{1}{(2^{2m}-t+1)}=\frac{(2^{2m}-t)!}{2^{2m}!}$$

That is when the transcripts are consistent then the experiment S and Π cannot be distinguished.

$$
\begin{aligned}
&|\Pr[S(A)=1]-\Pr[A^{\Pi,\Pi^{-1}}()=1|\\
&=|\Pr[S(A)=1\,|\,Tr(S)\text{ is consistent}]\Pr[Tr(S)\text{ is consistent}]\\
&+\Pr[S(A)=1\,|\,Tr(S)\text{ is inconsistent}]\Pr[Tr(S)\text{ is inconsistent}]\\
&-\Pr[A^{\Pi,\Pi^{-1}}()=1]\Pr[Tr(S)\text{ is consistent}]\\
&-\Pr[A^{\Pi,\Pi^{-1}}()=1]\Pr[Tr(S)\text{ is inconsistent}]|\\
&\le|(\Pr[S(A)=1\,|\,Tr(S)\text{ is consistent}]-\Pr[A^{\Pi,\Pi^{-1}}()=1])\Pr[Tr(S)\text{ is consistent}]|\\
&+|(\Pr[S(A)=1\,|\,Tr(S)\text{ is inconsistent}]-\Pr[A^{\Pi,\Pi^{-1}}()=1])\Pr[Tr(S)\text{ is inconsistent}]|\\
&\le 0+\Pr[\text{Tr(S) is inconsistent}]\\
&\le\binom{t}{2}\frac{1}{2^{2m}}\le\frac{t^2}{2^{2m+1}}
\end{aligned}
$$

$$
\Pr[A^{P_R,P_R^{-1}}()=1]-\Pr[S(A)=1]\le\frac{t^2}{2^{2m+1}}+\frac{t^2}{2^m}
$$

Let T consist of all valid transcripts for which the algorithm A returns 1.

$$
\therefore|\Pr[A^{P_R,P_R^{-1}}()=1]-\Pr[S(A)=1]|
$$

$$
=|\sum_{\tau\in T}(\Pr[A^{P_R,P_R^{-1}}\leftarrow\tau]-\Pr[S(A)\leftarrow\tau])|
$$

Let T' $\subset$ T, consist of the consistent transcripts (consistent with a permutation).

$$
\therefore|\sum_{\tau\in T\setminus T'}(\Pr[A^{P_R,P_R^{-1}}\leftarrow\tau]-\Pr[S(A)\leftarrow\tau])|
$$

$$
=|\sum_{\tau\in T\setminus T'}\Pr[S(A)\leftarrow\tau]|\le\frac{t^2}{2}\frac{1}{2^{2m}}=\frac{t^2}{2^{2m+1}}
$$

Bounding the other part will require the details of the construction. Fix a transcript $(x_i, y_i) \in T'$. Each x_i can be written as (L_i^0, R_i^0). This gets transformed due to the 4 rounds. After the j^{th} round we have (L_i^j, R_i^j).

Functions F_1 and F_4 are said to be good for the transcript if $(R_1^1, R_2^1, ..., R_t^1)$ and $(L_1^3, L_2^3, ..., L_t^3)$ do not have any repeatitions.

What happens when $R_i^1 = R_j^1$?

$R_i^1 = L_i^0 \oplus F_1(R_i^0)$

$R_j^1 = L_j^0 \oplus F_1(R_j^0)$

$\Rightarrow 0 = L_i^0 \oplus L_j^0 \oplus F_1(R_i^0) \oplus F_1(R_j^0)$

The algorithm A is non-repeating, so (L_i^0, R_i^0) is distinct.

Note $R_i^0 \neq R_j^0$, as otherwise $L_i^0 = L_j^0$, and thus $x_i = x_j$.

Thus in the above equality the function F_1 is called at two distinct points, thus the output is randomly chosen. Thus the probability of the equality being satisfied is 2^{-m} for a given i,j pair.

$\therefore \Pr_{F_1}[\exists i, j \in [t], R_i^1 = R_j^1] \leq \frac{t^2}{2^{m+1}}$.

Likewise, $0 = R_i^4 \oplus R_j^4 \oplus F_4(L_i^4) \oplus F_4(L_j^4)$

$\therefore \Pr_{F_1}[\exists i, j \in [t], L_i^3 = L_j^3] \leq \frac{t^2}{2^{m+1}}$.

Thus, $\Pr_{F_1, F_4}[F_1, F_4 \text{ not good for transcript}] \leq \frac{t^2}{2^m}$.

Let us fix good functions F_1, F_4. We have:

$L_i^3 = R_i^2 = L_i^1 \oplus F_2(R_i^1)$

$R_i^3 = L_i^2 \oplus F_3(R_i^2) = R_i^1 \oplus F_3(L_i^3)$

Thus, $F_2(R_i^1), F_3(L_i^3) = (L_i^3 \oplus L_i^1, R_i^3 \oplus R_i^1)$

Note, $(x_i, y_i) \Leftrightarrow F_2(R_i^1), F_3(L_i^3) = (L_i^3 \oplus L_i^1, R_i^3 \oplus R_i^1)$

If we have good functions, F_1 and F_4, the values R_i^1 and L_i^3 are distinct. Thus the occurence of (x_i, y_i) is independent of i and thus the probability that a particular transcript is obtained is exactly 2^{-2mt}.

Note that this is the same as for the algorithm S(A).

Thus in this case we cannot distinguish both the algorithms and A is unable to determine whether it is interacting with S(A) or (P_R, P_R^{-1}).

$$\therefore \left| \sum_{\tau \in T'} (\Pr[A^{P_R, P_R^{-1}} \leftarrow \tau] - \Pr[S(A) \leftarrow \tau]) \right|$$

$$\leq \sum_{\tau \in T'} (\Pr[A^{P_R, P_R^{-1}} \leftarrow \tau] \mid F_1, F_4 \text{ not good for } \tau) |) \Pr[F_1, F_4 \text{ not good for } \tau]$$

$$\leq \frac{t^2}{2^m}$$

Solve

- Complete the proof