Notes for Lecture 15

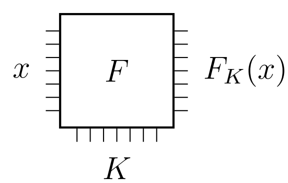
Scribed by Siu-Man Chan, posted March 12, 2009

Summary

Given one way permutations (of which discrete logarithm is a candidate), we know how to construct pseudorandom functions. Today, we are going to construct pseudorandom permutations (block ciphers) from pseudorandom functions.

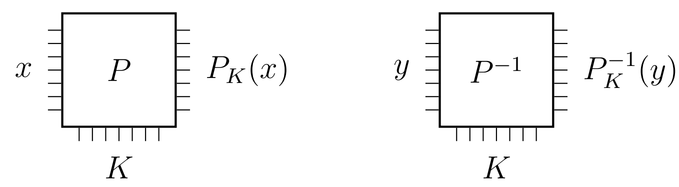
1 Pseudorandom Permutations

Recall that a pseudorandom function F is an efficient function : $\{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$, such that every efficient algorithm A cannot distinguish well $F_K(\cdot)$ from $R(\cdot)$, for a randomly chosen key $K \in \{0,1\}^k$ and a random function $R\colon \{0,1\}^n \to \{0,1\}^n$. That is, we want that $A^{F_K(\cdot)}$ behaves like $A^{R(\cdot)}$.



A pseudorandom permutation P is an efficient function : $\{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$, such that for every key K, the function P_K mapping $x \mapsto P_K(x)$ is a bijection. Moreover, we assume that given K, the mapping $x \mapsto P_K(x)$ is efficiently invertible (i.e. P_K^{-1} is efficient). The security of P states that every efficient algorithm A cannot distinguish well $\langle P_K(\cdot), P_K^{-1}(\cdot)\rangle$ from $\langle \Pi(\cdot), \Pi^{-1}(\cdot)\rangle$, for a randomly chosen key $K \in \{0,1\}^k$ and a random *permutation* $\Pi\colon \{0,1\}^n \to \{0,1\}^n$. That is, we want that $A^{P_K(\cdot), P_K^{-1}(\cdot)}$ behaves like $A^{\Pi(\cdot), \Pi^{-1}(\cdot)}$.

We note that the algorithm A is given access to both an oracle and its (supposed) inverse.



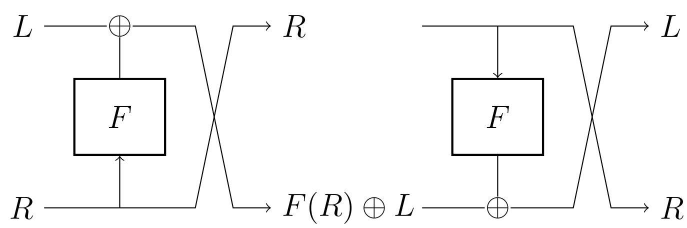
2 Feistel Permutations

Given *any* function $F\colon \{0,1\}^m \to \{0,1\}^m$, we can construct a permutation $D_F\colon \{0,1\}^{2m} \to \{0,1\}^{2m}$ using a technique named after Horst Feistel. The definition of D_F is given by

$$D_F(x, y) := y, F(y) \oplus x, \tag{1}$$

where x and y are m-bit strings. Note that this is an injective (and hence bijective) function, because its inverse is given by

$$D_F^{-1}(z, w) := F(z) \oplus w, z. \tag{2}$$



Also, note that D_F and D_F^{-1} are efficiently computable given F.

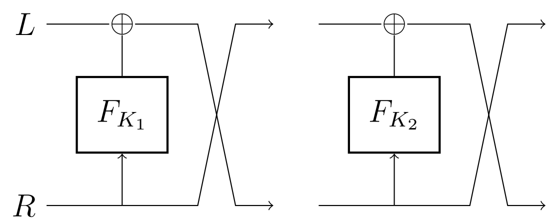
However, D_F needs not be a pseudorandom permutation even if F is a pseudorandom function, because the output of $D_F(x, y)$ must contain y, which is extremely unlikely for a truly random permutation.

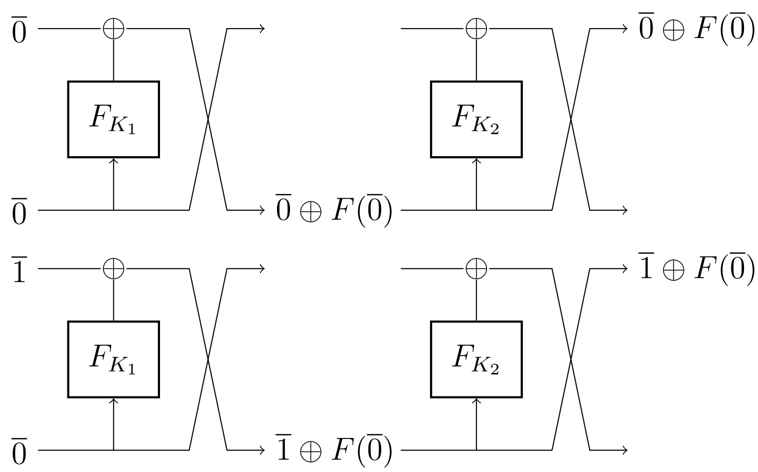
To avoid the above pitfall, we may want to repeat the construction twice. We pick two independent random keys K_1 and K_2, and compose the permutations $P(\cdot) := D_{F_{K_2}}(D_{F_{K_1}}(\cdot))$.

Indeed, the output does not always contain part of the input. However, this construction is still insecure, no matter whether F is pseudorandom or not, as the following example shows.

Here, $\overline{0}$ denotes the all-zero string of length m, $\overline{1}$ denotes the all-one string of length m, and $F(\cdot)$ is $F_{K_1}(\cdot)$. This shows that, restricting to the first half, $P(\overline{00})$ is the complement of $P(\overline{10})$, regardless of F.

What happens if we repeat the construction three times? We still do not get a pseudorandom permutation.





Exercise 1 (Not Easy) *Show that there is an efficient oracle algorithm A such that*

$$\mathop{\mathbb{P}}_{\Pi:\{0,1\}^{2m}\to\{0,1\}^{2m}}[A^{\Pi,\Pi^{-1}} = 1] = 2^{-\Omega(m)}$$

where Π is a random permutation, but for every three functions F_1, F_2, F_3, if we define $P(x) := D_{F_3}(D_{F_2}(D_{F_1}(x)))$ we have

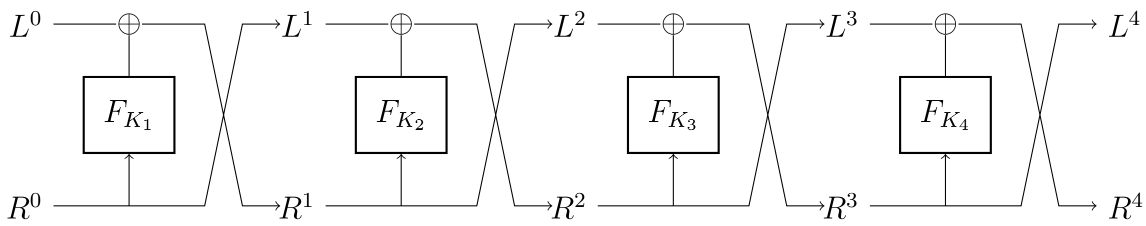
$$A^{P,P^{-1}} = 1$$

Finally, however, if we repeat the construction four times, with four independent pseudorandom functions, we get a pseudorandom permutation.

3 The Luby-Rackoff Construction

Let $F : \{0,1\}^k \times \{0,1\}^m \to \{0,1\}^m$ be a pseudorandom function, we define the following function $P : \{0,1\}^{4k} \times \{0,1\}^{2m} \to \{0,1\}^{2m}$: given a key $\overline{K} = \langle K_1, \ldots, K_4 \rangle$ and an input x,

$$P_{\overline{K}}(x) := D_{F_{K_4}}(D_{F_{K_3}}(D_{F_{K_2}}(D_{F_{K_1}}(x)))). \tag{3}$$



It is easy to construct the inverse permutation by composing their inverses backwards.

Theorem 1 (Pseudorandom Permutations from Pseudorandom Functions) *If F is a $(O(tr), \epsilon)$-secure pseudorandom function computable in time r, then P is a $(t, 4\epsilon + t^2 \cdot 2^{-m} + t^2 \cdot 2^{-2m})$ secure pseudorandom permutation.*

4 Analysis of the Luby-Rackoff Construction

Given four random functions $\overline{R} = \langle R_1, \ldots, R_4 \rangle$, $R_i : \{0,1\}^m \to \{0,1\}^m$, let $P_{\overline{R}}$ be the analog of Construction (3) using the random function R_i instead of the pseudorandom functions F_{K_i},

$$P_{\overline{R}}(x) = D_{R_4}(D_{R_3}(D_{R_2}(D_{R_1}(x)))) \tag{4}$$

We prove Theorem 1 by showing that

1. $P_{\overline{K}}$ is indistinguishable from $P_{\overline{R}}$ or else we can break the pseudorandom function
2. $P_{\overline{R}}$ is indistinguishable from a random permutation

The first part is given by the following lemma, which we prove via a standard hybrid argument.

Lemma 2 *If F is a $(O(tr), \epsilon)$-secure pseudorandom function computable in time r, then for every algorithm A of complexity $\leq t$ we have*

$$\left| \mathop{\mathbb{P}}_{\overline{K}}[A^{P_{\overline{K}}, P_{\overline{K}}^{-1}}() = 1] - \mathop{\mathbb{P}}_{\overline{R}}[A^{P_{\overline{R}}, P_{\overline{R}}^{-1}}() = 1] \right| \leq 4\epsilon \tag{5}$$

And the second part is given by the following lemma:

Lemma 3 *For every algorithm A of complexity $\leq t$ we have*

$$\left| \mathop{\mathbb{P}}_{\overline{R}}[A^{P_{\overline{R}}, P_{\overline{R}}^{-1}}() = 1] - \mathop{\mathbb{P}}_{\Pi}[A^{\Pi,\Pi^{-1}}() = 1] \right| \leq \frac{t^2}{2^{2m}} + \frac{t^2}{2^m}$$

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

We now prove Lemma 2 using a hybrid argument.

Proof: 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,
  $H_0(\cdot) := D_{F_{K_4}}(D_{F_{K_3}}(D_{F_{K_2}}(D_{F_{K_1}}(\cdot))))$;
- H_1: pick random keys K_2, K_3, K_4 and a random function $F_1\colon \{0,1\}^m \to \{0,1\}^m$,
  $H_1(\cdot) := D_{F_{K_4}}(D_{F_{K_3}}(D_{F_{K_2}}(D_{F_1}(\cdot))))$;
- H_2: pick random keys K_3, K_4 and random functions $F_1, F_2\colon \{0,1\}^m \to \{0,1\}^m$,
  $H_2(\cdot) := D_{F_{K_4}}(D_{F_{K_3}}(D_{F_2}(D_{F_1}(\cdot))))$;
- H_3: pick a random key K_4 and random functions $F_1, F_2, F_3\colon \{0,1\}^m \to \{0,1\}^m$,
  $H_3(\cdot) := D_{F_{K_4}}(D_{F_3}(D_{F_2}(D_{F_1}(\cdot))))$;
- H_4: pick random functions $F_1, F_2, F_3, F_4\colon \{0,1\}^m \to \{0,1\}^m$,
  $H_4(\cdot) := D_{F_4}(D_{F_3}(D_{F_2}(D_{F_1}(\cdot))))$.

Clearly, referring to (5), H_0 gives the first probability of using all pseudorandom functions in the construction, and H_4 gives the second probability of using all completely random functions. By triangle inequality, we know that

$$\exists i \quad \left| \mathbb{P}[A^{H_i, H_i^{-1}} = 1] - \mathbb{P}[A^{H_{i+1}, H_{i+1}^{-1}} = 1] \right| > \epsilon. \tag{6}$$

We now construct an algorithm $A'^{G(\cdot)}$ of complexity $O(tr)$ that distinguishes whether the oracle $G(\cdot)$ is $F_K(\cdot)$ or a random function $R(\cdot)$.

- The algorithm A' picks i keys $K_1, K_2, \ldots, K_i$ and initialize $4-i-1$ data structures $S_{i+2}, \ldots, S_4$ to $\emptyset$ to store pairs.
- The algorithm A' simulates $A^{O,O^{-1}}$. Whenever A queries O (or O^{-1}), the simulating algorithm A' uses the four compositions of Feistel permutations, where
  - On the first i layers, run the pseudorandom function F using the i keys $K_1, K_2, \ldots, K_i$;
  - On the i-th layer, run an oracle G;

– On the remaining $4-i-1$ layers, simulate a random function: when a new value for x is needed, use fresh randomness to generate the random function at x and store the key-value pair into the appropriate data structure; otherwise, simply return the value stored in the data structure.

When G is F_K, the algorithm A'^G behaves like $A^{H_i,H_i^{-1}}$; when G is a random function R, the algorithm A'^G behaves like $A^{H_{i+1},H_{i+1}^{-1}}$. Rewriting (6),

$$\left|\mathbb{P}_K[A'^{F_K(\cdot)}=1]-\mathbb{P}_R[A'^{R(\cdot)}=1]\right|>\epsilon,$$

and F is not $(O(tr),\epsilon)$-secure. $\square$

We say that an algorithm A is *non-repeating* if it never makes an oracle query to which it knows the answer. (That is, if A is interacting with oracles g,g^{-1} for some permutation g, then A will not ask twice for $g(x)$ for the same x, and it will not ask twice for $g^{-1}(y)$ for the same y; also, after getting the value $y=g(x)$ in an earlier query, it will not ask for $g^{-1}(y)$ later, and after getting $w=g^{-1}(z)$ it will not ask for $g(w)$ later.)

We shall prove Lemma 3 for non-repeating algorithms. The proof can be extended to arbitrary algorithms with some small changes. Alternatively, we can argue that an arbitrary algorithm can be simulated by a non-repeating algorithm of almost the same complexity in such a way that the algorithm and the simulation have the same output given any oracle permutation.

In order to prove Lemma 3 we introduce one more probabilistic experiment: we consider the probabilistic algorithm $S(A)$ that simulates $A()$ and simulates every oracle query by providing a random answer. (Note that the simulated answers in the computation of SA may be incompatible with any permutation.)

We first prove the simple fact that $S(A)$ is close to simulating what really happen when A interacts with a truly random permutation.

Lemma 4 *Let A be a non-repeating algorithm of complexity at most t. Then*

$$\left|\mathbb{P}[S(A)=1]-\mathbb{P}_\Pi[A^{\Pi,\Pi^{-1}}()=1]\right|\leq\frac{t^2}{2\cdot 2^{2m}} \tag{7}$$

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

Finally, it remains to prove:

Lemma 5 *For every non-repating algorithm A of complexity $\leq t$ we have*

$$\left|\mathbb{P}_{\overline{R}}[A^{P_{\overline{R}},P_{\overline{R}}^{-1}}()=1]-\mathbb{P}[S(A)=1]\right|\leq\frac{t^2}{2\cdot 2^{2m}}+\frac{t^2}{2^m}$$

It is clear that Lemma 3 follows Lemma 4 and Lemma 5.

We now prove Lemma 4.

$$
\begin{aligned}
& \left|\mathbb{P}[S(A)=1] - \mathbb{P}_{\Pi}[A^{\Pi,\Pi^{-1}}()=1]\right| \\
\leqslant\ & \mathbb{P}[\text{when simulating } S\text{, get answers inconsistent with any permutation}] \\
\leqslant\ & \frac{1}{2^{2m}}(1+2+\cdots+t-1) \\
=\ & \binom{t}{2}\frac{1}{2^{2m}} \\
\leqslant\ & \frac{t^2}{2\cdot 2^{2m}}.
\end{aligned}
$$

We shall prove Lemma 5 next time.