CS655: Cryptography **Jan 31, 2007**

Lecture Outline for Pseudorandom functions

3 Pseudorandom Functions

Readings: Sections 3.1–3.8 of Bellare&Rogaway

3.1 Function Families

- A *function family* is a map $F : \mathcal{K} \times D \rightarrow R$.

  $\mathcal{K}$ is the *keyspace*, D the *domain*, and R the range of F.

- The function $F_K : D \rightarrow R$ is defined by $F_K(X) = F(K, X)$. We call F_K an *instance* of F.

- Usually, $\mathcal{K} = \{0,1\}^k$, $D = \{0,1\}^\ell$, and $R = \{0,1\}^L$, where k is the *key length*, ℓ the input length, and L the output length.

- $K \stackrel{\$}{\leftarrow} \mathcal{K}$ means that K is uniformly randomly chosen from $\mathcal{K}$. That $f \stackrel{\$}{\leftarrow} F$ means that f is uniformly randomly chosen from F.

- A *permutation* is a bijection (i.e., a one-to-one onto map) whose domain and range are the same set.

- A block cipher is a family of permutations, e.g., DES: $\{0,1\}^{56} \times \{0,1\}^{64} \rightarrow \{0,1\}^{64}$.

3.2 Random functions and permutations

- $\mathsf{Func}(D, R)$ is the family of all functions mapping D to R.

  $\mathsf{Perm}(D)$ is the family of all permutations on D.

- $\mathsf{Func}(\ell, L)$ is the family of all functions mapping $\{0,1\}^\ell$ to $\{0,1\}^L$;

  $\mathsf{Func}(\ell)$ is the family of all functions mapping $\{0,1\}^\ell$ to $\{0,1\}^\ell$;

  and $\mathsf{Perm}(\ell)$ is the family of all permutations on $\{0,1\}^\ell$.

- The keyspace for $\mathsf{Func}(\ell, L)$ is:

  $$\mathsf{Keys}(\mathsf{Func}(\ell, L)) = \left\{(Y_1, \ldots, Y_{2^\ell}) : Y_1, \cdots, Y_{2^\ell} \in \{0,1\}^L\right\}$$

  The size of this keyspace is $(2^L)^{2^\ell} = 2^{L2^\ell}$, and the key length is $L2^\ell$.

- A random function g mapping $\{0,1\}^\ell$ to $\{0,1\}^L$ is a random instance of $\mathsf{Func}(\ell, L)$, i.e., $g \stackrel{\$}{\leftarrow} \mathsf{Func}(\ell, L)$.

  A random function means that the function is chosen randomly. The function itself is deterministic.

- Dynamic view of a random function g, or how to implement a random function. Maintains a table of all points that have been queried. When a new point is queried, return a random answer and store it in the table. When a point in the table is queried, returned the stored value.

- Look at Example 3.3.
- The keyspace for $\mathsf{Perm}(\ell)$ is:

$$\mathsf{Keys}(\mathsf{Perm}(\ell)) = \Big\{(Y_1, \ldots, Y_{2^\ell}) \ : \ Y_1, \cdots, Y_{2^\ell} \in \{0,1\}^\ell \text{ and } Y_1, \cdots, Y_{2^\ell} \text{ are distinct}\Big\}$$

  The keyspace has size $2^\ell!$.

- How to implement a random permutation on $\{0,1\}^\ell$, i.e., a random instance of $\mathsf{Perm}(\ell)$?
- For $\pi \xleftarrow{\$} \mathsf{Perm}(\ell)$, we have (example 3.5):
  1. Fix $X, Y \in \{0,1\}^\ell$, then $\mathsf{Pr}\left[\pi(X) = Y\right] = 2^{-\ell}$.
  2. Fix $X_1 \neq X_2 \in \{0,1\}^\ell$ and $Y_1 \neq Y_2 \in \{0,1\}^\ell$, then $\mathsf{Pr}\left[\pi(X_1) = Y_1 \mid \pi(X_2) = Y_2\right] = \frac{1}{2^\ell - 1}$.

3.3 Pseudorandom functions

- A *pseudorandom* function is a family F of functions with the property that the input-output behavior of a random instance of the family is "computationally indistinguishable" from that of a random function.

  No algorithm, with blackbox access to a function, can tell whether the function is randomly drawn from F or from the family of all functions over the domain and range.

- Consider the following scenario of distinguishing the following two worlds: one with a random function (i.e., $g \xleftarrow{\$} \mathsf{Func}(D, R)$), the other with a function drawn at random from F, a function family mapping D to R.
- Consider an adversary A with oracle access to a function g. An adversary is a probabilistic algorithm (Turing Machine).
- The *prf-advantage* of an adversary A.

$$\mathbf{Adv}_F^{\mathrm{prf}}(A) = \Pr[\mathbf{Exp}_F^{\mathrm{prf\text{-}1}}(A) = 1] - \Pr[\mathbf{Exp}_F^{\mathrm{prf\text{-}0}}(A) = 1]$$

  where in $\mathbf{Exp}_F^{\mathrm{prf\text{-}1}}$, A is given a function drawn at random from F, and in $\mathsf{Exp}_F^{\mathrm{prf\text{-}0}}$, A is given a random function.

- Advantage depends on the resources. Consider three resources: (1) running time of the algorithm, (2) number of queries A makes, (3) total length of A's queries.
- We say that a function family F is a "secure" PRF if, under certain resource restrictions, no adversary has a "significant" advantage.
- An alternative way of defining the advantage: a game between a Challenger and an adversary:
  1. The Challenger chooses $b \xleftarrow{\$} \{0,1\}$, and let $g \xleftarrow{\$} \mathsf{Func}(D, R)$ if $b = 0$, and let $g \xleftarrow{\$} F$ if $b = 1$.
  2. The Challenger then interacts with the adversary A, it evaluates g for the adversary at each point the adversary queries.

3. The adversary A outputs $b' \in \{0,1\}$ and wins if $b' = b$.

The advantage is defined to be $|\Pr[A \text{ wins }] - 0.5|$.

Question: How is this advantage related to $\mathbf{Adv}_F^{\mathrm{prf}}(A)$*?*

3.4 Pseudorandom permutations

- PRP under CPA: Given a family F of permutations on D, consider an adversary that is given oracle access to a function g, which is either an random permutation on D or a random instance of F, the adversary is asked to tell whether g is taken from F.

  Models chosen-plaintext attack against a cipher; however, the objective of the attack is to tell whether it is random.

- PFP under CCA: Similar to the CPA case, but the adversary has access to two oracles: g and g^{-1}.

  Models chosen-ciphertext (and chosen-plaintext) attack against a cipher.

- PRP-CCA implies PRP-CPA

3.5 Modeling block ciphers

- Classically, key recovery attacks are considered against block ciphers.

  Security against key recovery attack is necessary, but insufficient for block cipher security.

- Block ciphers should be secure PRP under CCA.

- **Conjecture** the following is true for any adversary $A_{t,q}$ that runs in time at most t and asks at most q queries.

$$\mathbf{Adv}_{\text{DES}}^{\text{prp-cpa}}(A_{t,q}) \leq c_1 \cdot \frac{t/T_{\text{DES}}}{2^{55}} + c_2 \cdot \frac{q}{2^{40}}$$

  First term models brute-force attack, second one models linear cryptanalysis, the best known theoretical attack.

- **Conjecture** the following is true for any adversary $B_{t,q}$ that runs in time $\leq t$ and makes $\leq q$ queries:

$$\mathbf{Adv}_{\text{AES}}^{\text{prp-cpa}}(B_{t,q}) \leq c_1 \cdot \frac{t/T_{\text{AES}}}{2^{128}} + c_2 \cdot \frac{q}{2^{128}}$$

3.6 Example Attacks

- **Example 3.10** Linear encryption.

- **Example 3.11** Given secure PRF $F : \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^L$, the function family $G : \{0,1\}^k \times \{0,1\}^\ell \to \{0,1\}^{2L}$, defined as

$$G_K(x) = F_K(x) \| F_K(\bar{x}),$$

  is not secure PRF.

3.7 Security against key recovery

- Define
$$\mathbf{Adv}_F^{\mathsf{kr}}(B) = \mathsf{Pr}\left[K \stackrel{\$}{\leftarrow} \mathsf{Keys}(F) : B^{F_K}() = K\right]$$

- **Proposition 1** ***[3.14]***Let $F : \mathcal{K} \times D \to R$ be a family of functions, and let B be a key-recovery adversary against F with running time at most t and making at most q queries, then there exists a PRF adversary A against F such that A has running time at most t plus the time for one evaluation of F and makes at most $q + 1$ queries, and

$$\mathbf{Adv}_F^{\mathsf{prf}}(A) \geq \mathbf{Adv}_F^{\mathsf{kr}}(B) - \frac{1}{|R|}$$

  Furthermore if $D = R$ and there also exists a PRP CPA adversary A against F such that

$$\mathbf{Adv}_F^{\mathsf{prf\text{-}cpa}}(A) \geq \mathbf{Adv}_F^{\mathsf{kr}}(B) - \frac{1}{|D| - q}$$

- Example of a cipher that is secure against key recovery, but is intuitively insecure.

3.8 The birthday attack

- One can distinguish a family of permutations from a family of random functions by the birthday attack. Note that the goal of the attack is not finding collisions, but rather distinguishing a permutation from a random function.

- Basic idea: a permutation never maps two values into the same. A random function may do that with probability $1/|R|$. Given a function, when one queries it about $\sqrt{|R|}$ times, one would expect to see collision. If collision doesn't occur, one can tell it is a permutation, rather than a random function.