Alin Tomescu, CSE408
Thursday, February 17th, Lecture #6

Pseudo random permutations

What is a Pseudo Random Permutation?

A family of functions P_k is a (t, q, ε)-pseudo-random permutation if $\forall A$ running in time $\leq t$ and making $\leq q$ oracle queries then:

$$Adv\ A = |\Pr[A^{P_k} = 0 | k \leftarrow U_l] - \Pr[A^{\pi} = 0 | \pi \leftarrow Perms]| \leq \varepsilon$$

$$P_k: \{0,1\}^n \rightarrow \{0,1\}^n, k \leftarrow \{0,1\}^l$$

Remarks:

- P_k is injective
  - Therefore, since $domain\ of\ P_k = range\ of\ P_k \Rightarrow P_k$ is surjective
- This is not a bitwise-permutation. You could have $P_k(000) = 011$
- P_k is injective and surjective $\Rightarrow P_k$ is bijective $\Leftrightarrow P_k$ is invertible

From an attacker's point of view $PRPs$ and $PRFs$ look almost identical.

What is the size of the set of all PRPs $P_k: \{0,1\}^n \rightarrow \{0,1\}^n$?

$$\left|Perms_{\{0,1\}^n}\right| = 2^n!$$

There are 2^n elements in the domain, and there are $2^n!$ ways of mapping them to each other

PRPs versus PRFs theorem

PRPs and PRFs are $\left(\infty, q, \frac{q^2}{2^{n+1}}\right)$-computationally indistinguishable.

$$Funcs_{\{0,1\}^n}^{\{0,1\}^n} \underset{\frac{q^2}{2^{n+1}}}{\overset{\infty, q}{\sim}} Perms_{\{0,1\}^n}$$

Proof

Note that permutations are one-to-one, but functions might not be, and this is the only property you can use to distinguish between them.

Program R (random function)

if $T[x]$ is not undefined **then**
    return $T[x]$
else
    $T[x] \leftarrow U_n$
    return $T[x]$

Program π (random permutation)

if $T[x]$ is not undefined **then**
    return $T[x]$
else
    $y \leftarrow U_n$
    if $y \in Range(T)$
        bad = **true**
        $y \leftarrow U_n - Range(T)$
    return $T[x]$

$$Adv\ A = |\Pr[A^R = 0 | R \leftarrow funcs] - \Pr[A^\pi = 0 | \pi \leftarrow perms]| =$$

Splitting them into cases, based on whether bad is true or false…

$$\begin{aligned} Adv\ A = |&\Pr[A^R = 0 | R \leftarrow funcs \cap bad = false] \Pr[bad = false] \\ &+ \Pr[A^R = 0 | R \leftarrow funcs \cap bad = true] \Pr[bad = true] \\ &- \Pr[A^\pi = 0 | \pi \leftarrow perms \cap bad = false] \Pr[bad = false] \\ &- \Pr[A^\pi = 0 | \pi \leftarrow perms \cap bad = true] \Pr[bad = true]| \end{aligned}$$

Grouping them based on whether bad is true or false…

$$\begin{aligned} Adv\ A = |&(\Pr[A^R = 0 | R \leftarrow funcs \cap bad = false] - \Pr[A^\pi = 0 | \pi \leftarrow perms \cap bad = false]) \Pr[bad = false] \\ &+ (\Pr[A^R = 0 | R \leftarrow funcs \cap bad = true] - \Pr[A^\pi = 0 | \pi \leftarrow perms \cap bad = true]) \Pr[bad = true]| \end{aligned}$$

Applying $|a + b| \leq |a| + |b|$…

$$\begin{aligned} Adv\ A \leq\ &\Pr[bad = false]\, |\Pr[A^R = 0 | R \leftarrow funcs \cap bad = false] - \Pr[A^\pi = 0 | \pi \leftarrow perms \cap bad = false]| \\ &+ \Pr[bad = true]\, |\Pr[A^R = 0 | R \leftarrow funcs \cap bad = true] - \Pr[A^\pi = 0 | \pi \leftarrow perms \cap bad = true]| \end{aligned}$$

When bad is false, then the $\boldsymbol{R}$ and $\boldsymbol{pi}$ programs behave the same and are indistinguishable. The advantage of the attacker in this case will be 0.

$$\Pr[A^R = 0 | R \leftarrow funcs \cap bad = false] - \Pr[A^\pi = 0 | \pi \leftarrow perms \cap bad = false] = 0$$

When bad is true, then we will bound advantage of the attacker by 1, which is the maximum he can have, assuming he has some very good method of distinguishing between

$$\begin{aligned} Adv\ A &\leq \Pr[bad = true]\, |\Pr[A^R = 0 | R \leftarrow funcs \cap bad = true] - \Pr[A^\pi = 0 | \pi \leftarrow perms \cap bad = true]| \\ &\leq \Pr[bad = true] \end{aligned}$$

$$Adv\ A \leq \Pr[bad = true] \leq \frac{q^2}{2^{n+1}}$$

Why: What is the probability of getting the same number twice after picking q n-bit random numbers? (The birthday problem)

Answer: $0.3\frac{q^2}{2^n} \leq p \leq 0.5\frac{q^2}{2^n}$ (you can find a proof for this if you look up the "birthday problem")

Examples of PRPs

AES

AES is a $(t, q, \frac{t}{2^{128}})$-secure PRP

Linear transformations (bad example)

Let $A = \{n \times n\ invertible\ matrices\ over\ GF(2)\}$

Alin Tomescu, CSE408
Thursday, February 17th, Lecture #6
$P_k(x) = A_k x$, where $A_k \leftarrow A$

This is a bad example since you can feed it a special kind of input which will reveal the columns of the matrix. For instance, if $n = 3$, and $A_k = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$ then you can compute $P\left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = col_1$, $P\left(\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = col_2$, and $P\left(\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}\right) = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = col_3$ effectively obtaining the matrix and thus getting knowledge of the full behavior of P_k

Real or random security (Real versus ideal world)

Ideal world:

- Whenever Alice sends a message to Bob, she sends him $E_k(m)$ and she also sends $E_k(\$m)$ to Eve, where \$ replaces m with random bits.
  - That's to say Eve will always know that a message of a particular length was sent.

Real world:

- Whenever Alice sends a message $E_k(m)$ to Bob, Eve gets a copy of $E_k(m)$.
- Even can also provide a message m to Alice for her to send it encrypted as $E_k(m)$ to Bob.
  - Eve can now see $E_k(m)$.

In the real world, we need "indistinguishability under chosen plaintext attack", a.k.a. IND-CPA.