Modern Cryptography

600.442

Lecture #12

Dr. Christopher Pappacena

Fall 2013

Last Time

- Fire Alarm
- Proved Goldreich-Levin Theorem
- One-way functions produce pseudorandom generators
- Pseudorandom generators produce pseudorandom functions

Constructing Pseudorandom Functions

- Fix a pseudorandom generator $G : \{0,1\}^n \to \{0,1\}^{2n}$ and write $G(s) = (G_0(s), G_1(s))$.

- Define a keyed function $F^{(1)} : \{0,1\}^n \times \{0,1\} \to \{0,1\}^n$ by $F_k^{(1)}(b) = G_b(k)$.

- Extend to $m > 1$ by setting

$$F_k^{(m)}(x_1 \dots x_m) = G_{x_m}(F_k^{(m-1)}(x_1 \dots x_{m-1}).$$

- $F_k^{(m)}$ can be viewed as a binary tree of depth m.

Theorem: The keyed function $F^{(n)}$ is a pseudorandom function.

Proof: The proof uses a hybrid argument. Define a distribution H_n^i on binary trees of depth n as follows:

- For nodes at level $j \leq i$, the values are chosen uniformly at random from $\{0,1\}^n$.

- For nodes at level $j > i$, look at the value k' of the node's parent and assign $G_0(k')$ if it is a left child and $G_1(k')$ if it is a right child.

Note that H_n^n corresponds to a random function and H_n^0 corresponds to $F^{(n)}$ with a uniformly-chosen key.

A Tale of Two Distinguishers

Let D be a distinguisher that can tell $F_k^{(n)}$ from a random function with advantage $\epsilon(n)$ and let $t(n)$ be the number of oracle queries that D makes.

We design a distinguisher D' which distinguishes outputs of G from random.

D' takes as input $t(n)$ strings of length $2n$ and chooses $i \to \{0, \ldots, n-1\}$ at random.

D' will run D as a subroutine and answer all oracle queries that D makes.

D' as Oracle

D' answers D's oracle queries as follows:

- On input $x_1 \ldots x_n$, D' uses $x_1 \ldots x_i$ to reach a node at level i in a binary tree.

- D' labels the left and right child of the node using one of its strings of length $2n$ and applies $G_{x_{i+1}}, \ldots, G_{x_n}$ to traverse down to the root.

- D' remembers the values of the tree it has filled in and answers consistently.

- D' only has to generate and store polynomially-many entries in its tree.

D's Advantage Becomes D''s Advantage

If D' is given $t(n)$ random strings of length $2n$, then it answers D's oracle queries with a function sampled from the distribution H_n^{i+1}.

If D' is given $t(n)$ outputs of the pseudorandom generator G, then it answers D's oracle queries with a function sampled from the distribution H_n^i.

As with the previous hybrid argument, D' succeeds in distinguishing the output of G from random with advantage $\epsilon(n)/n$.

This shows that $\epsilon(n)$ is negligible so that $F^{(n)}$ is a pseudorandom function.

Pseudorandom Permutations

We can use pseudorandom functions to build pseudorandom permutations and strong pseudorandom permutations via a Feistel network.

Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be a pseudorandom function.

Given a sequence $k = (k_1, \ldots, k_r)$ of r keys in $\{0,1\}^n$ and $x \in \{0,1\}^{2n}$, we can apply an r-round Feistel network with round functions F_{k_i} to x.

This construction yields a keyed function with key length rn and input and output length $2n$.

Feistel Networks Work

Theorem: If F is pseuodrandom, then a three-round Feistel network is a pseudorandom permutation on $2n$-bit strings with $3n$-bit keys.

Theorem: If F is pseudorandom, then a four-round Feistel network is a strong pseudorandom permutation on $2n$-bit strings with $4n$-bit keys.

Proofs are omitted. The fact that 2 rounds don't work and that 3 rounds don't give strong pseudorandom permutations are exercises in Katz and Lindell.

Recap So Far

So far we have demonstrated:

- One-way permutations imply pseudorandom generators.
- Pseudorandom generators imply pseudorandom functions.
- Pseudorandom functions imply strong pseudorandom permutations.

These are all that we need for secure private-key cryptography!

Theorem: If there exists a one-way permutation, then there exists a CCA-secure encryption scheme and a MAC that is existentially unforgeable under a chosen message attack.

One Way Functions Imply Private-Key Cryptography

In fact, it is possible to construct a pseudorandom generator from a one-way *function*.

Thus the existence of one-way functions implies the existence of secure cryptography.

Remarkably, the converse is true!

Private-Key Cryptography Implies One Way Functions

Theorem: If there exists a private-key encryption scheme with indistinguishable encryptions in the presence of an eavesdropper, then there exists a one-way function.

The proof will use the fact that messages longer than the key are allowed.

So this result does not contradict the unconditional security of the one-time pad.

Proof of the Theorem

Let $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ be a private-key encryption scheme with indistinguishable encryptions in the presence of an eavesdropper.

To allow for Enc to be a randomized algorithm we write $c = \text{Enc}_k(m, r)$ where r is a random value.

Assume that when an n bit key is used to encrypt a $2n$-bit message, then r has length $\ell(n)$.

The One Way Function

Define $f(m,k,r) = (\mathsf{Enc}_k(m,r), m)$.

- Note that f is efficiently computable beacuse $\mathsf{Enc}_k(m,r)$ is.

- We need to show that it is hard to invert.

- Intuitively, inverting f should require breaking Π. We will formalize this intuition.

The Adversary

Let $\mathcal{A}$ be a PPT algorithm that inverts f with success probability $\epsilon(n)$. We will construct an adversary $\mathcal{A}'$ for Π which uses $\mathcal{A}$ as a subroutine.

- $\mathcal{A}'$ chooses m_0 and m_1 of length $2n$ and receives the challenge ciphertext c from Π.

- $\mathcal{A}'$ gives (c, m_0) to $\mathcal{A}$ and receives back (m, k, r).

- $\mathcal{A}'$ guesses $b = 0$ if $f(m, k, r) = (c, m_0)$; otherwise $\mathcal{A}'$ outputs a random bit.

Success Probability

When $b = 0$, $\mathcal{A}$ inverts (c, m_0) with probability $\epsilon(n)$. In this case $\mathcal{A}'$ returns the correct answer. If $\mathcal{A}$ fails to invert (c, m_0), $\mathcal{A}'$ still gets the correct answer with probability 1/2. So

$$\Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\Pi,\mathcal{A}'}(n) = 1 | b = 0] = \epsilon(n) + \frac{1}{2}(1 - \epsilon(n)) = \frac{1}{2} + \frac{\epsilon(n)}{2}.$$

When $b = 1$, $\mathcal{A}$ can still successfully invert (c, m_0). This means that for some different key k' and random string r', we have $c = \mathsf{Enc}_k(m_1, r) = \mathsf{Enc}_{k'}(m_0, r')$.

How likely is this to happen?

There are only 2^n messages which encrypt to c, namely the decryptions of c with all 2^n possible keys.

Since there are 2^{2n} messages of length $2n$, c will be a valid ciphertext for m_0 with probability 2^{-n}. So,

$$\Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\Pi,\mathcal{A}'}(n) = 1 | b = 1] \geq \frac{1}{2}(1 - 2^{-n}) = \frac{1}{2} - \frac{1}{2^{n+1}}.$$

Putting this all together gives

$$\Pr[\mathsf{PrivK}^{\mathsf{eav}}_{\Pi,\mathcal{A}'}(n) = 1] \geq \frac{1}{2} + \frac{\epsilon(n)}{4} - \frac{1}{2^{n+2}}.$$

Since Π has indistinguishable encryptions in the presence of an eavesdropper, $\epsilon(n)$ is negligible and f is a one-way function.

Practical Considerations

So why do we use AES to instantiate a pseudorandom permutation, or RC4 to instantiate a pseudorandom generator?

Why not create cryptography from some NP-complete problem like the subset sum problem, or use another hard problem like factoring?

Because these would be *horribly* inefficient – running orders of magnitude more slowly!

Closing Observations

- It is also possible to prove the existence of one-way functions from secure MACs.

- We have not constructed collision-resistant hash functions from one-way functions.

- No construction is known and there is evidence that no such construction exists.

- Similarly, one-way functions do not appear to be enough to accomplish *public-key cryptography*, the topic of the second half of the course.

We have concluded the private-key portion of the course.

Next time, we will discuss public-key cryptography, which:

- Revolutionized cryptography in the 1970s.
- Gave birth to the science of modern cryptography.
- Pervades all aspects of digital life, for example enabling e-commerce.