1 Announcements

- Paper presentation sign up sheet is up. Please sign up for papers by next class.
- Lecture summaries and notes now up on course webpage

2 Recap and Overview

Previous lecture:

- Symmetric key encryption: various security defintions and constructions.

This lecture:

- Review definition of PRF, construction (and proof) of symmetric key encryption from PRF
- Block ciphers, modes of operation
- Public Key Encryption
- Digital Signature Schemes
- Additional background for readings

3 Review: Definition of PRF

Definition 1. *Let $F : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ be an efficient, length-preserving, keyed function. We say that F is a* pseudorandom function *if for all probabilistic polynomial-time distinguishers D, there exists a negligible function* neg *such that:*

$$\left|\Pr[D^{F_{\text{SK}}(\cdot)}(1^n) = 1] - \Pr[D^{f(\cdot)}(1^n) = 1]\right| \leq \text{neg}(n),$$

where $\text{SK} \leftarrow \{0,1\}^n$ *is chosen uniformly at random and f is chosen uniformly at random from the set of functions mapping n-bit strings to n-bit strings.*

4 Review: Security Against Chosen-Plaintext Attacks (CPA)

The experiment is defined for any private-key encryption scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, any adversary $\mathcal{A}$, and any value n for the security parameter:

The CPA indistinguishability experiment $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\mathcal{E}}(n)$:

1. A key SK is generated by running $\mathsf{Gen}(1^n)$.
2. The adversary $\mathcal{A}$ is given input 1^n and oracle access to $\mathsf{Enc}_{\text{SK}}(\cdot)$ and outputs a pair of messages m_0, m_1 of the same length.
3. A random bit $b \leftarrow \{0,1\}$ is chosen and then a ciphertext $C \leftarrow \mathsf{Enc}_{\text{SK}}(m_b)$ is computed and given to $\mathcal{A}$. We call C the *challenge ciphertext*.
4. The adversary $\mathcal{A}$ continues to have oracle access to $\mathsf{Enc}_{\text{SK}}(\cdot)$, and outputs a bit b'.
5. The output of the experiment is defined to be 1 if $b' = b$ and 0 otherwise. if $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\mathcal{E}}(n) = 1$, we say that $\mathcal{A}$ *succeeded.*

Definition 2. *A private-key encryption scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ has* indistinguishable encryptions under a chosen-plaintext attack *if for all probabilistic polynomial-time adversaries $\mathcal{A}$ there exists a negligible function* neg *such that*

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\mathcal{E}}(n) = 1] \leq \frac{1}{2} + \text{neg}(n),$$

where the probability is taken over the random coins used by $\mathcal{A}$, as well as the random coins used in the experiment.

5 CPA-Secure Encryption Schemes from Pseudorandom Functions

Picture:

Construction:
Let F be a pseudorandom function. Define a private-key encryption scheme for messages of length n as follows:

- Gen : on input 1^n, choose $\text{SK} \leftarrow \{0,1\}^n$ uniformly at random and output as the key.
- Enc : on input a key $\text{SK} \in \{0,1\}^n$ and a message $m \in \{0,1\}^n$, choose $r \leftarrow \{0,1\}^n$ uniformly at random and output the ciphertext
$$C := \langle r, F_{\text{SK}}(r) \oplus m \rangle.$$
- Dec : on input a key $k \in \{0,1\}^n$ and a ciphertext $c = \langle r, s \rangle$, output the plaintext message
$$m := F_{\text{SK}}(r) \oplus s.$$

Theorem 1. *If F is a pseudorandom function, then Construction is a fixed-length private-key encryption scheme for messages of length n that has indistinguishable encryptions under a chosen-plaintext attack.*

Proof. Let $\tilde{\mathcal{E}}$ be an encryption scheme that is exactly the same as $\mathcal{E}$ in Construction, except that a truly fandom function f is used in place of F_{SK}.

We claim that for every (even inefficient) adversary $\mathcal{A}$ that makes at most $q(n)$ queries to its encryption oracle, we have
$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\tilde{\mathcal{E}}}(n) = 1] \leq \frac{1}{2} + \frac{q(n)}{2^n}.$$

Let r_C denote the random string used when generating the challenge ciphertext $C = \langle r_c, f(r_c) \oplus m \rangle$. There are two subcases:

1. The value r_C is used by the encryption oracle to answer at least one of $\mathcal{A}$'s queries: In this case, $\mathcal{A}$ may easily determine which of its messages was encrypted.
   Since $\mathcal{A}$ makes at most $q(n)$ queries to its oracle and each oracle query is answered using a value r chosen uniformly at random, the probability of this event is at most $q(n)/2^n$.
2. The value r_C is never used by the encryption oracle to answer any of $\mathcal{A}$'s queries: In this case, $\mathcal{A}$ learns nothing about the value of $f(r_C)$from its interaction with the encryption oracle (since f is a truly random function). That means that the probability $\mathcal{A}$ outputs $b' = b$ is exactly $1/2$.

Let Repeat denote the event that r_C is used by the encryption oracle previously. Thus, we have

$$\begin{aligned}
\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\tilde{\mathcal{E}}}(n) = 1] &= \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\tilde{\mathcal{E}}}(n) = 1 \wedge \mathsf{Repeat}] + \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\tilde{\mathcal{E}}}(n) = 1 \wedge \overline{\mathsf{Repeat}}] \\
&\leq \Pr[\mathsf{Repeat}] + \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\tilde{\mathcal{E}}}(n) = 1 \mid \overline{\mathsf{Repeat}}] \\
&\leq \frac{q(n)}{2^n} + \frac{1}{2}.
\end{aligned}$$

Now fix some ppt adversary $\mathcal{A}$ which breaks the security of $\mathcal{E}$ and so

$$\varepsilon(n) = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\mathcal{E}}(n) = 1] - \frac{1}{2}$$

is non-negligible.

We use $\mathcal{A}$ to construct a distinguisher D for the pseudorandom function F. The distinguisher D is given oracle access to some function, and its goal is to determine whether this function is "pseudorandom" or "random".

Distinguisher D: D is given input 1^n and access to an oracle $\mathcal{O} : \{0,1\}^n \to \{0,1\}^n$.

1. Run $\mathcal{A}(1^n)$. Whenever $\mathcal{A}$ queries its encryption oracle on a message m, answer this query in the following way:
   (a) Choose $r \leftarrow \{0,1\}^n$ uniformly at random.
   (b) Query $\mathcal{O}(r)$ and obtain response s'.
   (c) Return the ciphertext $\langle r, s' \oplus m \rangle$ to $\mathcal{A}$.
2. When $\mathcal{A}$ outputs messages $m_0, m_1 \in \{0,1\}^n$ choose a random bit $b \leftarrow \{0,1\}$ and then
   (a) Choose $r \leftarrow \{0,1\}^n$ uniformly at random.
   (b) Query $\mathcal{O}(r)$ and obtain response s'.
   (c) Return the challenge ciphertext $\langle r, s' \oplus m_b \rangle$ to $\mathcal{A}$.
3. Continue answering any encryption oracle queries of $\mathcal{A}$ as before. Eventually, $\mathcal{A}'$ outputs a bit b'. Output 1 if $b' = b$, and output 0 otherwise.

Key points:

1. If D's oracle is a pseudorandom function, then the view of $\mathcal{A}$ is distributed identically to the view of $\mathcal{A}$ in $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\mathcal{E}}(n)$. Thus,
$$\Pr[D^{F_{\mathrm{SK}}(\cdot)}(1^n) = 1] = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\mathcal{E}}(n)].$$
2. If D's oracle is a random function, then the view of $\mathcal{A}$ is distributed identically to the view of $\mathcal{A}$ in $\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\tilde{\mathcal{E}}}(n)$. Thus,
$$\Pr[D^{f(\cdot)}(1^n) = 1] = \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A},\tilde{\mathcal{E}}}(n)].$$

Thus, we have that

$$\Pr[D^{F_{\mathrm{SK}}(\cdot)}(1^n) = 1] - \Pr[D^{f(\cdot)}(1^n) = 1] \geq \varepsilon(n) - \frac{q(n)}{2^n}.$$

Since we assumed $\varepsilon(\cdot)$ is non-negligible, this leads to a contradiction to the security of F_{SK}.

6 Pseudorandom Permutations and Block Ciphers

Let $F : \{0,1\}^* \to \{0,1\}^*$ be an efficient, length-preserving, keyed function. We call F a *keyed permutation* if for every SK, $F_{\mathrm{SK}}(\cdot)$ is *one-to-one*. We say that a keyed permutation is *efficient* if there is a polynomial-time algorithm computing $F_{\mathrm{SK}}(x)$ given SK, x as well as a polynomial-time algorithm computing $F^{-1}_{\mathrm{SK}}(x)$, given SK, x.

Definition, same as before, only indistinguishable from random permutation.

Proposition 1. *If F is a pseudorandom permutation then it is also a pseudorandom function.*

6.1 Modes of Operation

Introduction to Modern Cryptography, pages 96-102.

A mode of operation is a way of encrypting arbitrary-length messages using a block cipher (i.e. a pseudorandom permutation).

Note that arbitrary-length messages can be unambiguously padded to a total length that is a multiple of any desired block size by appending a 1 followed by sufficiently many 0's.

Mode 1—Electronic Code Book (ECB) mode. This is the most naive mode of operation.

Not CPA secure, does not have indistinguishable encryptions in the presence of an eavesdropper.

Mode 2—Cipher Block Chaining (CBC) mode. In this mode, a random initial vector (IV) of length n is first chosen. Then, each of the remaining ciphertext blocks is generated by applying the pseudorandom permutation to the XOR of the current plaintext block and the previous ciphertext block.

If F is a pseudorandom permutation then CBC-mode encryption is CPA-secure.
Drawback–encryption must be carried out sequentially, decryption can be done in parallel.

Mode 3—Output Feedback (OFB) mode. This mode is a way of using a block-cipher to generate a pseudorandom stream that is then XORed with the message.

Is CPA-secure if F is a pseudorandom function. Drawback–both encryption and decryption can be done in parallel. On the other hand–bulk of the computation can be done independently of the actual message in a preprocessing stage.

Mode 4—Counter (CTR) mode. *randomized counter mode*.

Randomized counter mode is CPA-secure. Encryption and decryption can be fully parallelized, it is possible to generate the psuedorandom stream ahead of time, independently of the message. "Random access" can decrypt the i-th block without decrypting anything else.

7 Public Key Encryption

Picture and intuition.

Definition 3. *A* public key encryption scheme *is a tuple of probabilistic polynomial-time algorithms* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *such that:*

1. *The* key generation algorithm Gen *takes as input the security parameter* 1^n *and outputs a pair of keys* $(\mathrm{PK}, \mathrm{SK})$. *We refer to the first of these as the* public key *and the second as the* private key. *We assume for convenience that* PK *and* SK *each have length at least* n*, and that* n *can be determined from* PK, SK.
2. *The* encryption algorithm Enc *takes as input a public key* PK *and a message* m *from some underlying plaintext space (that may depend on* PK*). It ouputs a ciphertext* C*, and we write this as* $C \leftarrow \mathsf{Enc}_{\mathrm{PK}}(m)$.
3. *The* decryption algorithm Dec *takes as input a private kye* SK *and a ciphertext* C*, and outputs a message* m *or a special symbol* $\perp$*, denoting failure. We assume without loss of generality that* Dec *is deterministic and write this as* $m := \mathsf{Dec}_{\mathrm{SK}}(C)$.

It is required that there exists a negligible function neg *such that for every* n*, every* $(\mathrm{PK}, \mathrm{SK})$ *output by* $\mathsf{Gen}(1^n)$*, and every message* m *in the appropriate underlying plaintext space, it holds that*

$$\Pr[\mathsf{Dec}_{\mathrm{SK}}(\mathsf{Enc}_{\mathrm{PK}}(m)) \neq m] \leq \mathrm{neg}(n).$$

Note: Equivalence between eavesdropping indistinguishability experiment and cpa indistinguishability experiment.

7.1 Security against Chosen-Plaintext Attacks

The experiment is defined for any public-key encryption scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, any adversary $\mathcal{A}$, and any value n for the security parameter:

The CPA indistinguishability experiment $\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathcal{E}}(n)$:

1. $\mathsf{Gen}(1^n)$ is run to obtain keys $(\mathrm{PK}, \mathrm{SK})$.
2. The adversary $\mathcal{A}$ is given PK and outputs a pair of messages m_0, m_1 of the same length.
3. A random bit $b \leftarrow \{0, 1\}$ is chosen and then a ciphertext $C \leftarrow \mathsf{Enc}_{\mathrm{PK}}(m_b)$ is computed and given to $\mathcal{A}$. We call C the *challenge ciphertext*.
4. $\mathcal{A}$ outputs a bit b'.
5. The output of the experiment is defined to be 1 if $b' = b$ and 0 otherwise.

Definition 4. *A public-key encryption scheme* $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *has* indistinguishable encryptions under a chosen-plaintext attack *if for all probabilistic polynomial-time adversaries* $\mathcal{A}$ *there exists a negligible function* neg *such that*

$$\Pr[\mathsf{PubK}^{\mathsf{cpa}}_{\mathcal{A},\mathcal{E}}(n) = 1] \leq \frac{1}{2} + \mathrm{neg}(n),$$

where the probability is taken over the random coins used by $\mathcal{A}$*, as well as the random coins used in the experiment.*

- Impossibility of perfectly-secret public key encryption.
- Insecurity of deterministic public-key encryption.
- Equivalence of single message and multiple message eavesdropping experiment.

Public key encryption schemes can be constructed from various computational assumptions such as:

- RSA assumption
- Factoring based assumptions
- Discrete-log based assumptions
- Lattice based assumptions

An interesting open problem is to construct public key encryption schemes from new assumptions, that seem to withstand quantum attacks.

7.2 Hybrid Encryption

Use the public key encryption scheme to encrypt a secret key, encrypt message using this secret key.

This is what is done in practice (e.g. SSL). The reason it is done this way is that symmetric key encryption is far more efficient than public key encryption. Thus, reduce the use of public key encryption to only a single encryption per session.

8 Digital Signatures

Picture and intuition.

Definition 5. *A* signature scheme *is a tuple of three probabilistic polynomial-time algorithms* $(\mathsf{SigGen}, \mathsf{Sign}, \mathsf{Verify})$ *satisfying the following:*

1. *The* key generation algorithm Gen *takes as input a security parameter* 1^n *and outputs a pair of keys* (PK, SK). *These are called the* public key *and the* private key*, respectively. We assume for convenience that* PK *and* SK *each have length at least* n*, and that* n *can be determined from* PK, SK.
2. *The* signing algorithm Sign *takes as input a private key* SK *and a message* $m \in \{0,1\}^*$. *It outputs a signature* σ*, denoted as* $\sigma \leftarrow \mathsf{Sign}_{\text{SK}}(m)$.
3. *The deterministic* verification algorithm Verify *takes as input a public key* PK*, a message* m*, and a signature* σ*. it ouputs a bit* b*, with* $b = 1$ *meaning* valid *and* $b = 0$ *meaning* invalid*. We write this as* $b := \mathsf{Verify}_{\text{PK}}(m, \sigma)$.

It is required that for every n*, every* (PK, SK) *output by* $\mathsf{Gen}(1^n)$*, and every* $m \in \{0,1\}^*$*, it holds that*

$$\mathsf{Verify}_{\text{PK}}(m, \mathsf{Sign}_{\text{SK}}(m)) = 1.$$

8.1 Security of signature schemes

Let $\Pi = (\mathsf{SigGen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme, and consider the following experiment for an adversary $\mathcal{A}$ and parameter n:

The signature experiment $\mathsf{Sigforge}_{\mathcal{A},\Pi}(n)$:

1. $\mathsf{Gen}(1^n)$ is run to obtain keys (PK, SK).
2. Adversary $\mathcal{A}$ is given PK and oracle access to $\mathsf{Sign}_{\text{SK}}(\cdot)$. The adversary then outputs (m, σ). Let Q denote the set of messages whose signatures were requrested by $\mathcal{A}$ during its execution.
3. The output of the experiment is defined to be 1 if and only if:
   - $\mathsf{Verify}_{\text{PK}}(m, \sigma) = 1$.
   - $m \notin Q$.

Definition 6. *A signature scheme* $\Pi = (\mathsf{SigGen}, \mathsf{Sign}, \mathsf{Verify})$ *is* existentially unforgeable under an adaptive chosen-message attack *if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there exists a negligible function* neg *such that:*

$$\Pr[\mathsf{Sigforge}_{\mathcal{A},\Pi}(n) = 1] \leq \text{neg}(n).$$

8.2 Lamport's One-Time Signature Scheme

One Way Functions. A function $f : \{0,1\}^* \to \{0,1\}^*$ is *one-way* if the following two conditions hold:

1. (Easy to compute:) There exists a polynomial-time algorithm that on input x outputs $f(x)$.
2. (Hard to invert:) For all probabilistic polynomial time algorithms $\mathcal{A}$ there exists a negligible function neg such that:
$$\Pr[\mathcal{A}(y) \in f^{-1}(y)] \leq \mathrm{neg}(n),$$
where $x \leftarrow \{0,1\}^n, y := f(x)$.

The construction. We illustrate the construction for the case of signing 3-bit messages. Let f be a one-way function. The public key consists of 6 elements $y_{1,0}, y_{1,1}, y_{2,0}, y_{2,1}, y_{3,0}, y_{3,1}$ in the range of f. The priate key constains the corresponding pre-images $x_{1,0}, x_{1,1}, x_{2,0}, x_{2,1}, x_{3,0}, x_{3,1}$.

To sign a message $m = (m_1, m_2, m_3)$, where each m_i is a single bit, the signer releases the appropriate pre-image x_{i,m_i} for $1 \leq i \leq 3$; the signature σ simply consists of the three values $(x_{1,m_1}, x_{2,m_2}, x_{3,m_3})$. Verification is carried out in the natural way: presented with the candidate signature (x_1, x_2, x_3) on the message $m = (m_1, m_2, m_3)$, accept if and only if $f(x_i) = y_{i,m_i}$ for $1 \leq i \leq 3$.

8.3 Collision Resistant Hash Functions

A *collision* in a function H is a pair of distinct inputs x and x' such that $H(x) = H(x')$; in this case we also say that x and x' *collide* under H. A function H is *collision resistant* if it is infeasible for any probabilistic polynomial time algorithm to find a collision in H. We will deal with a *family* of hash functions indexed by a "key" s. This key s is not kept secret. Rather, it is used merely to specify a particular function H_s from the family.

8.4 Many-time signature schemes

Can be constructed from collision-resistant hash functions See Introduction to Modern Cryptography pages 435-445 as well as lecture notes from Rafael Pass's Crypto class `www.cs.cornell.edu/courses/cs6830/2009fa/scribes/lecture21.pdf`.

Let $\Pi = (\mathsf{SigGen}, \mathsf{Sign}, \mathsf{Verify})$ be a one-time secure digital signature scheme for messages $\{0,1\}^n$ and $h : \{0,1\}^* \to \{0,1\}^n$ be a CRHF.

Construct one-time digital signature scheme $\Pi' = (\mathsf{SigGen}', \mathsf{Sign}', \mathsf{Verify}')$ as follows:

- SigGen': Generate a public-private key pair $(\mathsf{vksig}, \mathsf{sksig})$ for Π and sample a CRHF h.
- $\mathsf{Sign}'_{\mathsf{sksig}}(m)$: Output $\mathsf{Sign}_{\mathsf{sksig}}(h(m))$.
- $\mathsf{Verify}'_{\mathsf{vksig}}(\sigma, m)$: Output $\mathsf{Verify}_{\mathsf{vksig}}(h(m), \sigma)$.

We will show how to build a many-time signature scheme out of Π'.

Approach 1: Generate a new key pair after each signing. Keep track of the sequence of keys generated. Start with original key pair $(\mathsf{vksig}_0, \mathsf{sksig}_0)$. To sign the first message:

- Generate $(\mathsf{vksig}_1, \mathsf{sksig}_1)$
- Compute $\sigma_1 \leftarrow \mathsf{Sign}_{\mathrm{SK}_0}(m_1 || \mathsf{vksig}_1)$
- Output $\sigma'_1 = (1, \sigma_1, m_1, \mathsf{vksig}_1)$.

More generally, to sign the i-th message:

- Generate $(\mathsf{vksig}_i, \mathsf{sksig}_i)$
- Compute $\sigma_i \leftarrow \mathsf{Sign}_{\mathrm{SK}_{i-1}}(m_i || \mathsf{vksig}_i)$
- Output $\sigma'_i = (i, \sigma_i, m_i, \mathsf{vksig}_i, \sigma'_{i-1})$.

Note that we are signing messages longer than the secret key $(m_i || \mathsf{vksig}_i)$. This is why we could not use any Π (such as Lamports), but first needed to transform to Π'.

Disadvantages of this approach:

- Length of signature grows linearly with number of messages.
- Signer has to keep state and update state with every message.

Approach 2: Use each generated pair $(\mathsf{vksig}_i, \mathsf{sksig}_i)$ to sign 2 new key pairs. This forms a complete binary tree where each key signs all its children.

Picture:

Non-leaf key pairs only sign their children, leaf key-pairs only sign a message once. To sign a message m, use the m-th leaf.

- Since all key pairs (the complete binary tree of depth n) can be pre-computed, we do not need to keep state.
- Since the (hashes of the messages) are in $\{0,1\}^n$, where n is the height of the tree, we can treet the message as a number from 0 to $2^n - 1$. To sign hash value m, we use the key-pair sequence from the root to the leaf number n.
- Problem: Lots of space!!

Approach 3: Signer uses a PRF to make key generation and signing deterministic.

Signer stores 2 keys k, k' for a PRF F. When needed the values $\mathsf{vksig}_w, \mathsf{sksig}_w$ can be generated:

1. Compute $r_w = F_k(w)$.
2. Compute $(\mathsf{vksig}_w, \mathsf{sksig}_w) = \mathsf{SigGen}(1^n; r_w)$

In addition, the key k' is used similarly to generate the value r'_w to compute the signature σ_w.