

CSE 207B: Applied Cryptography

Nadia Heninger

UCSD

Fall 2025 Lecture 4

Announcements

1. HW 1 is due today! Turn it in now if you haven't yet!
2. HW 2 is out, due before class in 1 week.

Last time: Block ciphers

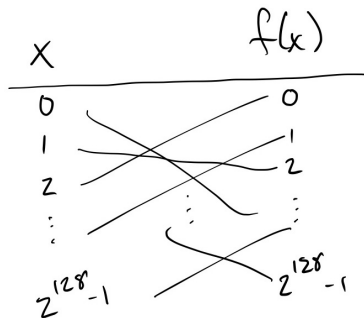
This time: Pseudorandom functions and chosen plaintext attacks

Pseudo-random functions (PRFs)

Deterministic algorithm F :

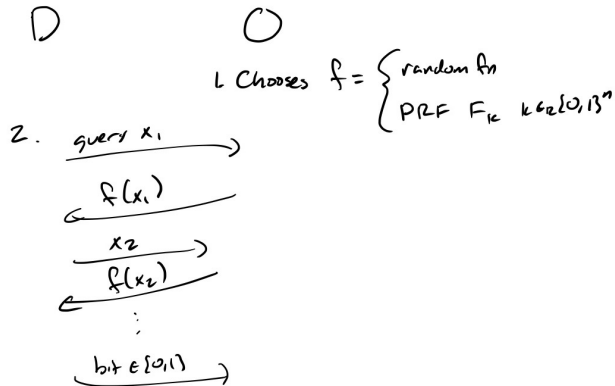
- $k \in K, x \in X, y \in Y$
- $F_k(x) = y$
- Should be computationally indistinguishable from a truly random function

Truly Random Function



In contrast to the (pseudo)random permutations from last lecture, this function is not required to be one-to-one.

Distinguishing experiment for PRFs



Definition

F_k is a secure PRF if \forall efficient D

$$|\Pr[D(F_k) = 1] - \Pr[D(\text{random fn}) = 1]| \text{ negligible}$$

Is a PRP indistinguishable from a PRF?

Consider a distinguishing experiment between functions and permutations.

Observation: The only way to distinguish between a permutation and a non-permutation function is to find a collision, inputs so that $f(x_1) = f(x_2)$.

Let $|X| = N$. By the birthday bound, the adversary will observe a collision in outputs in a PRF with constant probability after \sqrt{N} inputs.

Theorem

An adversary that makes Q queries can distinguish a random permutation from a random function with probability at most $Q^2/2N$.

Constructing PRGs from PRFs

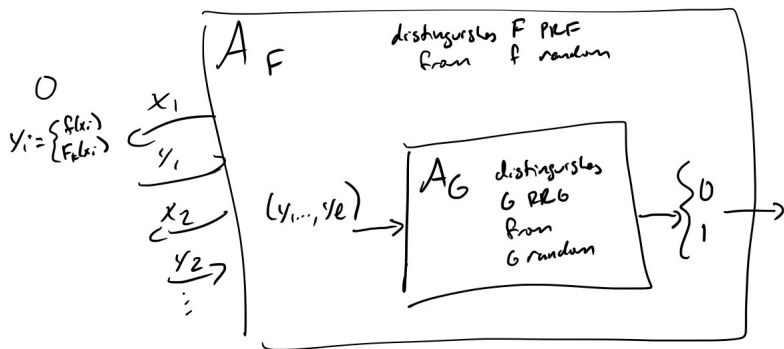
Theorem

Let x_1, \dots, x_ℓ be fixed, distinct elements, and F be a PRF.

$G(k) = (F_k(x_1), F_k(x_2), \dots, F_k(x_\ell))$ is a secure PRG.

Proof.

Assume for contradiction that adversary A_G can distinguish this PRG from random. Can construct a distinguisher A_F for the PRF.



Counter Mode

In the previous construction x_1, \dots, x_ℓ just need to be distinct elements. So just choose r and let $x_1 = r, x_2 = r + 1, \dots$

Then we can use this as a stream cipher to encrypt:

$$\text{Enc}_k(m) = (r, F_k(r) \oplus m[0], F_k(r+1) \oplus m[1], \dots, F_k(r+\ell) \oplus m[\ell])$$

$$\text{Dec}_k(c) = (F_k(r) \oplus c[0], F_k(r+1) \oplus c[1], \dots, F_k(r+\ell) \oplus c[\ell])$$

This is semantically secure.

Attack models we've seen so far:

Ciphertext-only attack

- Most restrictive attack model

Known plaintext attack

- Historical example: WWII Enigma-encoded messages from Germans ending in "Heil Hitler"
- Modern example: Observing ciphertext from someone visiting the main page of Wikipedia over HTTPS.

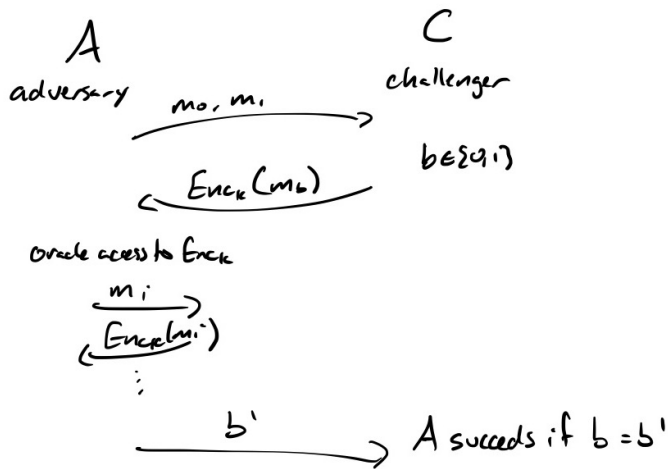
Both of these are covered by semantic security.

New attack model:

Chosen plaintext attack

- Historical example: British military would place mines in particular locations hoping Germans would send encrypted messages about that location.
- Modern example: Attacker-controlled Javascript on a web page causes victim web client to make a HTTPS connection.

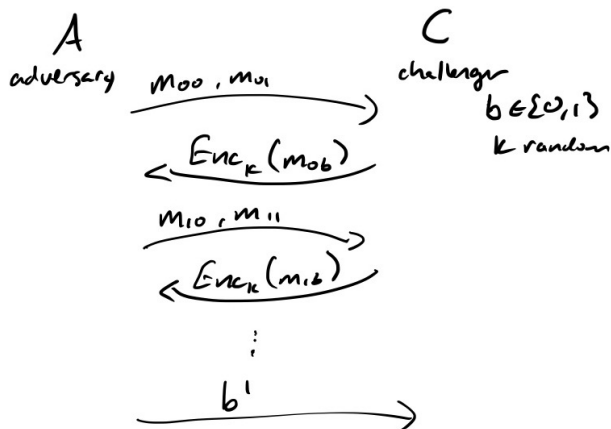
Chosen plaintext attack



Definition

Enc is CPA-secure if \forall efficient A , $\Pr[A \text{ succeeds}] \leq 1/2 + \epsilon$ for ϵ negligible

Another definition of chosen plaintext attack



Definition

Enc is CPA-secure if

$$|\Pr[A \text{ outputs } 1 \mid b = 1] - \Pr[A \text{ outputs } 1 \mid b = 0]| \text{ negligible}$$

Theorem

No deterministic cipher can be CPA-secure.

Theorem

No deterministic cipher can be CPA-secure.

Proof.

Adversary queries (m_0, m_1) then (m_0, m_0) .



Using a PRF for CPA-secure encryption

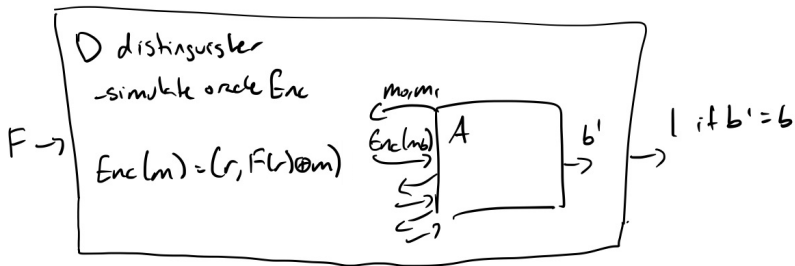
- Generate k at random.
- Encryption:
 1. Generate r uniformly at random.
 2. $\text{Enc}_k(m) = (r, F_k(r) \oplus m)$
- Decryption:
 1. Parse $c = (r, s)$
 2. $\text{Dec}_k(c) = F_k(r) \oplus s.$

Theorem

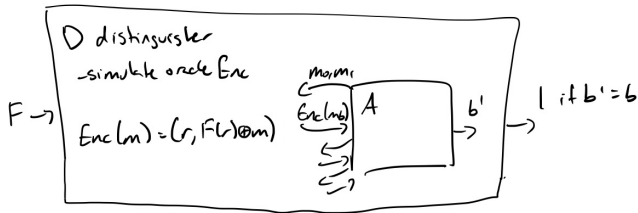
The $\text{Enc}_k(m) = (r, F_k(r) \oplus m)$ construction on the previous slide is CPA-secure.

Proof.

By contradiction. Assume adversary A can win CPA-security game #1 with non-negligible advantage, construct a PRF distinguisher.



Proof.



Assume A wins CPA game against F with advantage $d > \text{negl.}$

1. If F pseudorandom, $\Pr[A \text{ succeeds}] = 1/2 + d$
2. If F is a truly random function f : A makes Q oracle queries.
 - If nonce r_c used in challenge is repeated, A learns value of $f(r_c)$ and succeeds with probability 1.

$$\Pr[r_c \text{ repeated across oracle queries}] \leq Q/2^n$$

- If r_c not used in challenge, no information: $\Pr[\text{success}] = 1/2$

$$\Pr[A \text{ succeeds}] \leq 1/2 + Q/2^n$$

$$|\Pr[D | F_k] - \Pr[D | f]| = |1/2 + d - (1/2 + Q/2^n)| = d - Q/2^n > \text{negl.}$$

Using stream ciphers in a CPA-secure way

Augment stream cipher with an *initialization vector* or IV.

- $\text{Enc}_k(m) = (IV, G(k, IV) \oplus m)$

For this to be secure, $G(k, IV)$ needs to be pseudorandom when IV is known.

Insecure if IV is ever reused.

WEP insecurity. WEP is broken in multiple ways: it uses a 24-bit IV, which repeats with 50% probability after 5,000 packets.

Extension: Randomized Counter Mode

If we use a block cipher in counter mode with a randomized starting value r , this is CPA-secure.

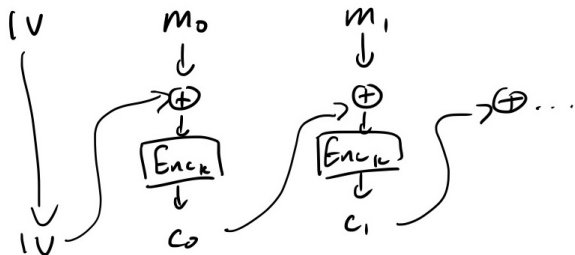
$$\text{Enc}_k(m) = (r, F_k(r) \oplus m[0], F_k(r+1) \oplus m[1], \dots, F_k(r+\ell) \oplus m[\ell])$$

$$\text{Dec}_k(c) = (F_k(r) \oplus c[0], F_k(r+1) \oplus c[1], \dots, F_k(r+\ell) \oplus c[\ell])$$

The value r is the IV.

This is an ok choice of mode of operation for AES.

Cipher block chaining (CBC) mode



1. IV has same length as block length.
2. $c_i = Enc_k(c_{i-1} \oplus m_i)$
3. Output $(IV, c_0, c_1, c_2, \dots)$.

IV should be random.

CBC mode is CPA-secure, but suffers from implementation vulnerabilities that you get to break in HW 3.

This is why people have been moving away from CBC mode.

Chosen Plaintext Attack against CBC mode

Assume: Enc_k is a secure block cipher.

Rogaway 1995: CBC mode not secure against chosen plaintext attacks if attacker can see IV or previous block before choosing m .

1. Attacker observes $c_1, c_2, \dots, c_{i-1}, c_i, \dots, c_j$.
2. Attacker wants to distinguish c_i , stream is currently at c_j .
3. Attacker guesses c_i plaintext is p .
4. Attacker causes victim to encrypt $c_j \oplus c_{i-1} \oplus p$.
5. Victim sends $\text{Enc}_k(c_j \oplus (c_j \oplus c_{i-1} \oplus p)) = \text{Enc}_k(c_{i-1} \oplus p)$
6. Attacker compares $\text{Enc}_k(c_{i-1} \oplus p)$ to c_i , match if p correct.

Chosen Plaintext Attacks against CBC mode in practice

- Dai 2002: SSH2 chains ciphertext between client, server
- SSLv3, TLS 1.0 also

128 bits is a lot to brute force in one block

- In many cases, know a lot of plaintext

How to request encryptions?

- Javascript, Java can make cookie-bearing requests across domains

Duong, Rizzo 2011: BEAST (Browser Exploit Against SSL/TLS)

- Pad plaintexts to 1 unknown text byte per block

Message padding for encryption

Q: How do you encrypt odd-length messages with a fixed-length block cipher?

A: Pad message somehow.

Message padding for encryption

Q: How do you encrypt odd-length messages with a fixed-length block cipher?

A: Pad message somehow.

Example: PKCS 7 padding.

- If message is b bytes short of 128-bit block, append b bytes $bbb \dots b$ to make block multiple of 128.
- Decryptor checks and strips off padding.

Message padding for encryption

Q: How do you encrypt odd-length messages with a fixed-length block cipher?

A: Pad message somehow.

Example: PKCS 7 padding.

- If message is b bytes short of 128-bit block, append b bytes $bbb \dots b$ to make block multiple of 128.
- Decryptor checks and strips off padding.

Q: What do you do if padding is incorrect?

A: Throw an error?

Padding oracle attacks

Vaudenay 2002: CBC mode encryption is insecure if attacker can distinguish bad from good message padding

Padding Oracle Attack

Want to decrypt $(IV, c_1, \dots, c_n) = \text{Enc}_k(m_1, \dots, m_n)$.

Have oracle that given $(IV', c'_1, \dots, c'_\ell)$ will:

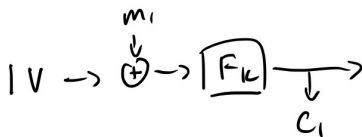
1. Compute $(m'_1, \dots, m'_\ell) = \text{Dec}_k(IV', c'_1, \dots, c'_\ell)$
2. Return

$$\begin{cases} \text{valid} & \text{if } m'_\ell \text{ ends in valid padding} \\ \text{invalid} & \text{if } m'_\ell \text{ doesn't end in valid padding} \end{cases}$$

Vaudenay CBC padding oracle attack

Input: Ciphertext $(IV, c_1, c_2, \dots, c_n)$

1. Attacker sends $(IV \oplus 00 \dots 00t, c_1) = F_k(IV \oplus m_1 \oplus 00 \dots 00t)$ to decryption padding oracle.

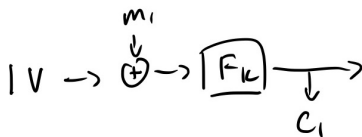


oracle returns $\left\{ \begin{array}{l} \text{valid if } m_1 \oplus 000 \dots 0t = \dots 1 \text{ (most likely)} \\ \dots 22 \\ \dots 333 \\ \vdots \\ \text{invalid otherwise} \end{array} \right.$

Vaudenay CBC padding oracle attack

Input: Ciphertext $(IV, c_1, c_2, \dots, c_n)$

1. Attacker sends $(IV \oplus 00 \dots 00t, c_1) = F_k(IV \oplus m_1 \oplus 00 \dots 00t)$ to decryption padding oracle.



oracle returns $\left\{ \begin{array}{l} \text{valid if } m_1 \oplus 000 \dots 0t = \dots 1 \text{ (most likely)} \\ \dots 22 \\ \dots 333 \\ \vdots \\ \text{invalid otherwise} \end{array} \right.$

2. Try all 256 values of t .

$$IV[0] \oplus m_1[0] \oplus t = 1$$

known unknown known known

3. Good probability of learning value

4. Iterate for successive bytes of ciphertext.

5. $256n$ query complexity to learn n bytes of plaintext.

- HW 2 is due before class in 1 week.