Symmetric-Key Encryption

CSE 5351: Introduction to Cryptography

Reading assignment:

- Chapter 3
- Read sections 3.1-3.2 first (skipping 3.2.2)

Negligible functions

- A nonnegative function $f: \mathbb{N} \rightarrow \mathbb{R}$ is said to be negligible if for every positive polynomial $P(n)$, there is an integer n_0 such that

$$f(n) < \frac{1}{P(n)} \quad \text{for all } n > n_0 \text{ (i.e., for sufficiently large } n\text{).}$$

- Examples: 2^{-n}, $2^{-\sqrt{n}}$, $n^{-\log n}$ are negligible functions.
- Negligible functions approach zero faster than the reciprocal of every polynomial.
- We write negl(n) to denote an unspecified negligible function.

Properties of negligible functions

- If $\text{negl}_1(n)$ and $\text{negl}_2(n)$ are negligible functions, then $\text{negl}_1(n) + \text{negl}_2(n)$ is negligible.
- If $\text{negl}(n)$ is a negligible function and $p(n)$ a polynomial, then $p(n) \cdot \text{negl}(n)$ is negligible.
- Examples: $2^{-n} + 2^{-\sqrt{n}}$ and $n^{100} n^{-\log n}$ are negligible.

Relaxing the security requirement

- In perfect indistinguishability (perfect secrecy), the adversary has
  - unlimited computing power,
  - success rate $\leq 1/2$;
  - also, message length is hidden.

- Now we relax the notion of perfect indistinguishability by
  - limiting adversaries to having poly(n) computing power,
  - allowing the success rate to be $\leq 1/2 + \text{negl}(n)$,
  - not hiding message length.

Security Parameter

- The n in the previous slide is called a security parameter, which indicates the key length.
- We will associate an encryption scheme Π with a secureity parameter n, and would like Π to be secure in the sense that any adversary with $poly(n)$ computing power can break Π with at most $negl(n)$ probability.

PPT Algorithms

- Probabilistic polynomial-time algorithms
- Polynomial-time: the running time is polynomial in input length.
- Input length is the number of bits of the input.
- What is the length of n in binary, and what is the length of 1^n?
- What is the difference between these two statements:
  - $A(n)$ is a PPT algorithm.
  - $A(1^n)$ is a PPT algorithm.

Private-key encryption scheme w. security parameter *n*

- A tuple of polynomial-time algorithms: $\Pi = (Gen, Enc, Dec)$
- Key generation algorithm Gen: On input 1^n, outputs a key $k \in \{0,1\}^n$. We write $k \leftarrow Gen(1^n)$. (n: security parameter.)
- Encryption algorithm Enc: On input a key k and a message $m \in \{0,1\}^*$, outputs a ciphertext c. We write $c \leftarrow Enc_k(m)$.
- Decryption algorithm Dec: On input a key k and a ciphertext c, Dec outputs a message m or an error symbol $\perp$.
  We write $m := Dec_k(c)$.
- Correctness requirement: for every $k \leftarrow Gen(1^n)$ and $m \in \{0,1\}^*$,
  $$Dec_k\left(Enc_k(m)\right) = m.$$
- Gen, Enc are probabilistic. Dec, deterministic.

- If message space $M = \{0,1\}^{\ell(n)}$, then $\Pi = (Gen, Enc, Dec)$ is said to be a fixed-length private-key encryption scheme for messages of length $\ell(n)$.

- If $Gen(1^n)$ simply outputs $k \leftarrow_u \{0,1\}^n$, we omit Gen and simply denote the scheme by (Enc, Dec). This is almost always the case.

Ciphertext Indistinguishability Experiment $\mathrm{PrivK}^{\mathrm{eav}}_{A,\Pi}(n)$

- Adversary: PPT eavesdropper with a single ciphertext.
- (Gen, Enc, Dec): an encryption scheme with security parameter n.
- Imagine a game played by Bob and an adversary A (Eve):
  - Eve, given input 1^n, outputs a pair of messages $m_0,\ m_1$ with $|m_0| = |m_1|$ (i.e., having the same length).
  - Bob chooses a key $k \leftarrow Gen(1^n)$ and a bit $b \leftarrow_u \{0,1\}$; computes $c \leftarrow E_k(m_b)$; and gives c to Eve.
  - Eve outputs a bit b', trying to tell whether c is an encryption of m_0 or m_1.
  - The output, $\mathrm{PrivK}^{\mathrm{eav}}_{A,\Pi}(n)$, of the experiment is 1 iff $b = b'$ (i.e., Eve succeeds.)

Ciphertext Indistinguishability against an eavesdropper

- Definition: A private-key encryption scheme has indistinguishable encryptions against an eavesdropper (or is EAV-secure) if for all probabilistic polynomial-time adversaries A, there is a negligible function $negl(n)$ such that (for all n)

$$\Pr\left[\text{PrivK}_{A,\Pi}^{\text{eav}}(n)=1\right] \le \frac{1}{2}+\text{negl}(n)$$

where the probability is taken over the randomness used by A, the randomness used by Bob to choose the key and the bit b, as well as the randomness used by Enc.

- $$\Pr\left[\text{PrivK}_{A,\Pi}^{\text{eav}}(n)=1\right]=\Pr\left[\begin{array}{l} A\left(1^n,m_0,m_1,Enc_k(m_b)\right)=b: \\ b\leftarrow_u \{0,1\},\ k\leftarrow Gen(1^n),\ m_0,m_1\leftarrow A(1^n) \end{array}\right]$$

An equivalent formulation

- For $b = 0$ or 1 (fixed), let $\mathrm{PrivK}^{\mathrm{eav}}_{A,\Pi}(n, b)$ denote the previous experiment with the fixed b used.
- Let $\mathrm{output}\left(\mathrm{PrivK}^{\mathrm{eav}}_{A,\Pi}(n, b)\right)$ denote the adversary's output.

$$\Pr\left[\mathrm{output}\left(\mathrm{PrivK}^{\mathrm{eav}}_{A,\Pi}(n, b)\right) = 1\right] = \Pr\left[\begin{array}{l} A\left(1^n, m_0, m_1, Enc_k(m_b)\right) = 1 : \\ k \leftarrow Gen(1^n),\ m_0, m_1 \leftarrow A(1^n) \end{array}\right]$$

- Theorem: A private-key encryption scheme is EAV-secure if and only if for all PPT adversaries A, there is a negligible function $negl(n)$ such that

$$\left|\Pr\left[\mathrm{output}\left(\mathrm{PrivK}^{\mathrm{eav}}_{A,\Pi}(n, 0)\right) = 1\right] - \Pr\left[\mathrm{output}\left(\mathrm{PrivK}^{\mathrm{eav}}_{A,\Pi}(n, 1)\right) = 1\right]\right|$$

$$\leq \mathrm{negl}(n).$$

- That is,

$$\left| \Pr\left[\begin{array}{l} A\left(1^n, m_0, m_1, Enc_k(m_0)\right) = 1 : \\ k \leftarrow Gen(1^n),\ m_0, m_1 \leftarrow A(1^n) \end{array} \right] - \Pr\left[\begin{array}{l} A\left(1^n, m_0, m_1, Enc_k(m_1)\right) = 1 : \\ k \leftarrow Gen(1^n),\ m_0, m_1 \leftarrow A(1^n) \end{array} \right] \right|$$

$$\leq negl(n)$$

Adversaries cannot learn any bit of the plaintext

- Let m^i denote the ith bit of m.
- If an encryption scheme is EAV-secure, then from a ciphertext $c \leftarrow Enc_k(m)$, it is infeasible for the adversary to recover m^i.
- Theorem: If a fixed-length private-key encryption scheme with $M = \{0,1\}^{\ell(n)}$ is EAV-secure, then for all PPT adversaries A and any $i \in \{1,\ldots,\ell(n)\}$, it holds:

$$\Pr\left[A\left(1^n, Enc_k(m)\right) = m^i : k \leftarrow_u \{0,1\}^n,\ m \leftarrow_u \{0,1\}^{\ell(n)} \right] \le \frac{1}{2} + \text{negl}(n).$$

Secure Encryption Schemes

- Secure: EAV-secure, CPA-secure, or CCA-secure.
- Secure private-key encryption schemes may be constructed from:
  - Pseudorandom generators
  - Pseudorandom functions
  - Pseudorandom permutations.

Pseudorandom Generators and Stream Ciphers

Encryption schemes using pseudorandom generators

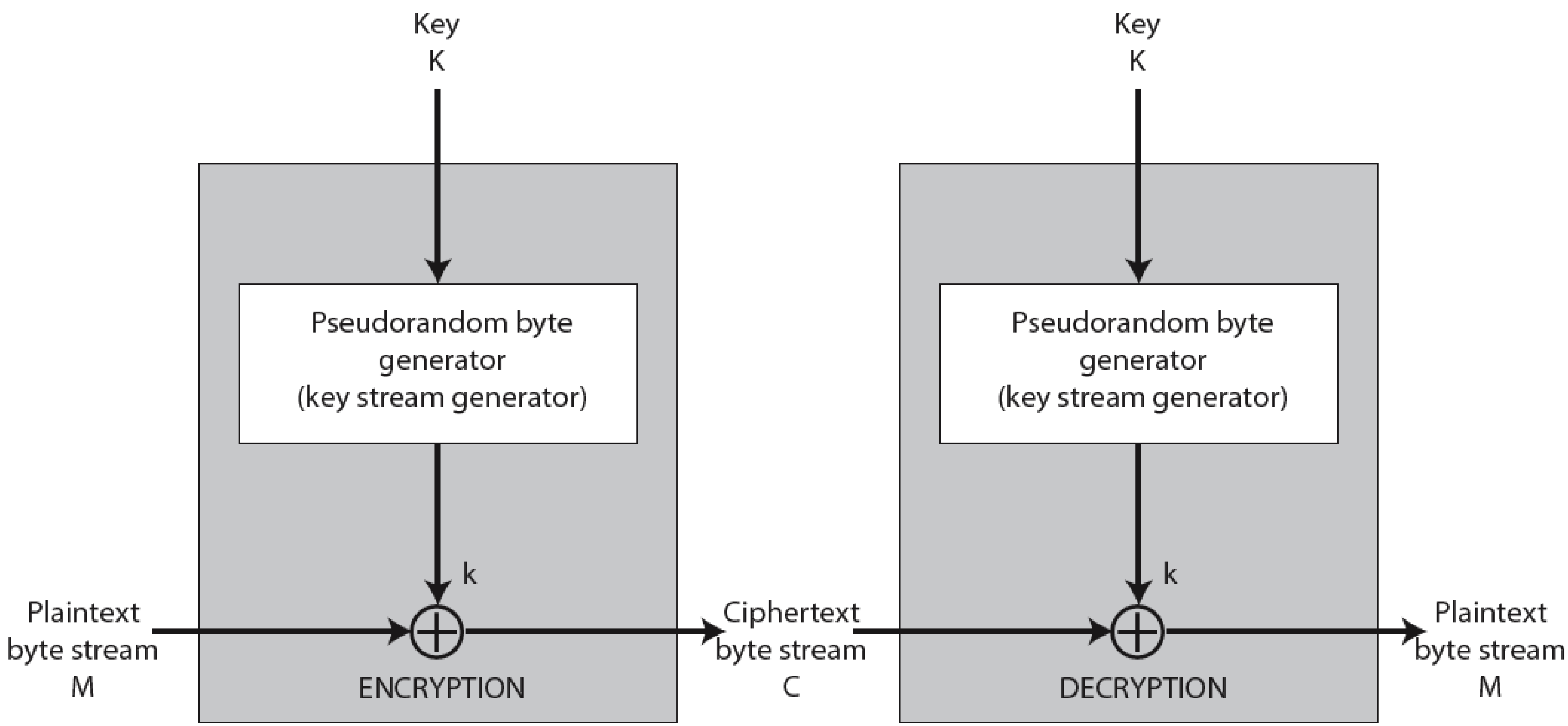
K&L: Section 3.3

Motivation

- Vernam's one-time pad scheme is perfectly secure against single-ciphertext eavesdropper.
- Drawback: it requires a random key as long as the message.
- Solution: use a short key as seed to generate a "pseudorandom" key that is as long as needed.
- This is the basic idea of stream ciphers.

Stream ciphers

- The term "stream cipher" may refer to the entire encryption scheme or just the pseudorandom generator.



What is a pseudorandom generator?

- Informally, a pseudorandom generator is an algorithm G that given a (short) truly random string s, outputs a "random-like" (i.e.,pseudorandom) string longer than s.
- Informally, a string r is "random-like" if it is hard to tell whether or not r is generated by a truly-random generator.
- Loosely speaking, two sets $A_n, B_n \subseteq \{0,1\}^n$ are said to be polynomially indistinguishable if for every polynomial distinguisher D,

$$\left| \Pr\left[D(r)=1 : r \leftarrow_u A_n\right] - \Pr\left[D(r)=1 : r \leftarrow_u B_n\right] \right| \leq \text{negl}(n)$$

- In the above, we were actually talking about the indistinguishability between two ensembles (sequences) of sets: $\left(A_n\right)_{n\in\mathbb{N}}$ and $\left(B_n\right)_{n\in\mathbb{N}}$.
- Definition: Two ensembles of sets $\left(A_n\right)_{n\in\mathbb{N}}$ and $\left(B_n\right)_{n\in\mathbb{N}}$ are polynomially indistinguishable if for every polynomial-time distinguisher D, it holds that

$$\left| \Pr\left[D(r)=1:\ r\leftarrow_u A_n\right] - \Pr\left[D(r)=1:\ r\leftarrow_u B_n\right] \right| \le \operatorname{negl}(n)$$

- Which of the following are polynomially indistinguishable?
  - $A_n=\{0,1\}^n,\ B_n=\{0,1\}^n-\left\{0^n\right\}$
  - $A_n=\{0,1\}^n,\ B_n=\left\{s\in\{0,1\}^n:\ s>2^{100} \text{ as a binary integer}\right\}$
  - $A_n=\{0,1\}^n,\ B_n=0\,||\,\{0,1\}^{n-1}$

$A_n = \{0,1\}^n$ and $B_n = \{0,1\}^n - \{0^n\}$ are polynomially indistinguishable.

$$\Pr\left[D(r)=1:\ r \leftarrow_u A_n\right] \triangleq \sum_{r \in A_n} \Pr[r] \cdot \Pr[D(r)=1]$$

$$= \frac{1}{2^n} \sum_{r \in A_n} \Pr[D(r)=1]$$

$$= \frac{1}{2^n} \Pr\left[D(0^n)=1\right] + \frac{1}{2^n} \sum_{r \in B_n} \Pr[D(r)=1]$$

$$\Pr\left[D(r)=1:\ r \leftarrow_u B_n\right] \triangleq \sum_{r \in B_n} \Pr[r] \cdot \Pr[D(r)=1]$$

$$= \frac{1}{2^n - 1} \sum_{r \in B_n} \Pr[D(r)=1]$$

$$\left|\ \Pr\left[D(r)=1:\ r \leftarrow_u A_n\right] - \Pr\left[D(r)=1:\ r \leftarrow_u B_n\right]\ \right| \le \text{negl}(n)$$

Definition of pseudorandom generator

- Let $\ell(\cdot)$ be a polynomial such that $\ell(n) > n$ for all $n > 0$.
- Let G be a deterministic polynomial-time algorithm that, for any input string $s \in \{0,1\}^n$, outputs a string $G(s) \in \{0,1\}^{\ell(n)}$.
- G is said to be a pseudorandom generator with expansion factor $\ell(\cdot)$ if for every polynomial-time distinguisher D,

$$\Big| \Pr\left[D(G(s)) = 1 : \; s \leftarrow_u \{0,1\}^n \right] - \Pr\left[D(r) = 1 : \; r \leftarrow_u \{0,1\}^{\ell(n)} \right] \Big| \leq \text{negl}(n)$$

- That is, the two ensembles $\left(A_n\right)_{n \in N}$ and $\left(B_n\right)_{n \in N}$, are polynomially indistinguishable, where $A_n = \left\{ G(s) : \; s \in \{0,1\}^n \right\}$ and $B_n = \left\{0,1\right\}^{\ell(n)}$.

Example: insecure pseudorandom generator

- Let $G(s) = s \,\|\, (s_1 \oplus \cdots \oplus s_n)$ for $s = s_1 \ldots s_n \in \{0,1\}^n$.
- Expansion factor $l(n) = n+1$.
- G is not a pseudorandom generator:
  - For $r \in \{0,1\}^{n+1}$, let $D(r) = \begin{cases} 1 & \text{if } r_1 \oplus \cdots \oplus r_n = r_{n+1} \\ 0 & \text{otherwise} \end{cases}$
  - $\Pr\left[D(G(s)) = 1 : \; s \leftarrow_u \{0,1\}^n \right] = 1$
  - $\Pr\left[D(r) = 1 : \; r \leftarrow_u \{0,1\}^{n+1} \right] = 1/2$
  - Difference between the two probabilities is not negligible.

Remarks

- A string r is said to be a random string if it is generated by a true random generator (i.e., $r \leftarrow_u \{0,1\}^\ell$, where $\ell = |r|$).
- A string r is said to be a pseudorandom string if it is generated by a pseudorandom generator.
- What if the distinguisher D has unlimited (or exponential) time?
  - Given $r \in \{0,1\}^{\ell(n)}$, let $D(r) = \begin{cases} 1 & \text{if } r = G(s) \text{ for some } s \in \{0,1\}^n \\ 0 & \text{otherwise} \end{cases}$
  - $\Pr\left[D(G(s)) = 1 :\ s \leftarrow_u \{0,1\}^n \right] = 1$
  - $\Pr\left[D(r) = 1 :\ r \leftarrow_u \{0,1\}^{\ell(n)} \right] = 2^n / 2^{\ell(n)} = 1/2^{\ell(n)-n}$
  - Difference between the two probabilities is not negligible.

Existence of pseudorandom generators

- If one-way functions exist, then pseudorandom generators exist.
- That is, pseudorandom generators can be constructed from one-way functions.
- Chapter 7 of the K&L book shows how to construct pseudorandom generators from one-way permutations.
- True pseudorandom generators are slow for applications.
- In practice, algorithms such as RC4 are used.

Existence of pseudorandom generators (basic idea)

- Let $f:\{0,1\}^n \to \{0,1\}^n$ be a one-way function.
- Let $b:\{0,1\}^n \to \{0,1\}$ be a hard-core predicate of f.
  - A boolean function defined on the domain of f.
  - Easy to compute $b(x)$ from x.
  - But hard to compute $b(x)$ from $f(x)$.
- Given seed x, let $x_0 = x$.
- Starting from x_0, apply f repeatedly:

$$x_0 \xrightarrow{f} x_1 \xrightarrow{f} x_2 \xrightarrow{f} \cdots \xrightarrow{f} x_{l(n)-1}$$

- Let $G(x) = \left(b(x_0),\ b(x_1), b(x_2),\ \ldots,\ b(x_{l(n)-1})\right)$.
- G is a pseudorandom generator with expansion factor $l(n)$.

Example: Blum-Blum-Shub pseudorandom generator

- Let $n = pq$ for two large primes $p,\ q$.
- Let $f(x) = x^2 \bmod n$. //one-way function//
- Let $b(x) =$ the least significant bit of x //hard-core predicate//

$$x_0 \xrightarrow{f} x_1 \xrightarrow{f} x_2 \xrightarrow{f} \cdots \xrightarrow{f} x_{l(n)-1}$$

- Let $G(x) = \left(b(x_0),\ b(x_1), b(x_2),\ \ldots,\ b\left(x_{l(n)-1}\right)\right)$.
- G is a pseudorandom generator with expansion factor $l(n)$.

Example: Blum-Blum-Shub pseudorandom generator

- Suppose $n = pq = 29 \times 31 = 899$.
- Suppose $x_0 = 100$.
- Then we have the sequence

$$100,\ 111,\ 634,\ 103,\ 720,\ 576,\ 45,\ 227,\ 286,\ 886,\ 169,$$

$$692,\ 596,\ 111,\ 634,\ 103,\ 720,\ \ldots$$

- The generated bits are 01010011001001010…

Encryption schemes based on pseudorandom generators

- From a pseudorandom generator with expansion factor $\ell(n)$, we can easily construct an EAV-secure $\ell(n)$-bit encryption scheme.

- G: a pseudorandom generator with expansion factor $\ell(n)$.
- Key generation: on input 1^n, outputs a key $k \leftarrow_u \{0,1\}^n$.
- Encryption: on input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^{\ell(n)}$,
  outputs the ciphertext $c := m \oplus G(k)$.
- Decryption: on input a key $k \in \{0,1\}^n$ and a ciphertext $c \in \{0,1\}^{\ell(n)}$,
  outputs the $m := c \oplus G(k)$.
- Denote this scheme by Π.

Security

Theorem. The scheme Π constructed above is EAV-secure (i.e. has indistinguishable encryptions against eavesdroppers).

Intuition:

- If encrypting with a truely random string r :

$$\left.\begin{aligned} c_0 &= m_0 \oplus r \\ c_1 &= m_1 \oplus r \end{aligned}\right\} \text{perfectly indistinguishable}$$

- If a pseudorandom string $G(s)$ is used instead:

$$\left.\begin{aligned} c_0 &= m_0 \oplus G(s) \\ c_1 &= m_1 \oplus G(s) \end{aligned}\right\} \text{polynomially indistinguishable}$$

Proof sketch

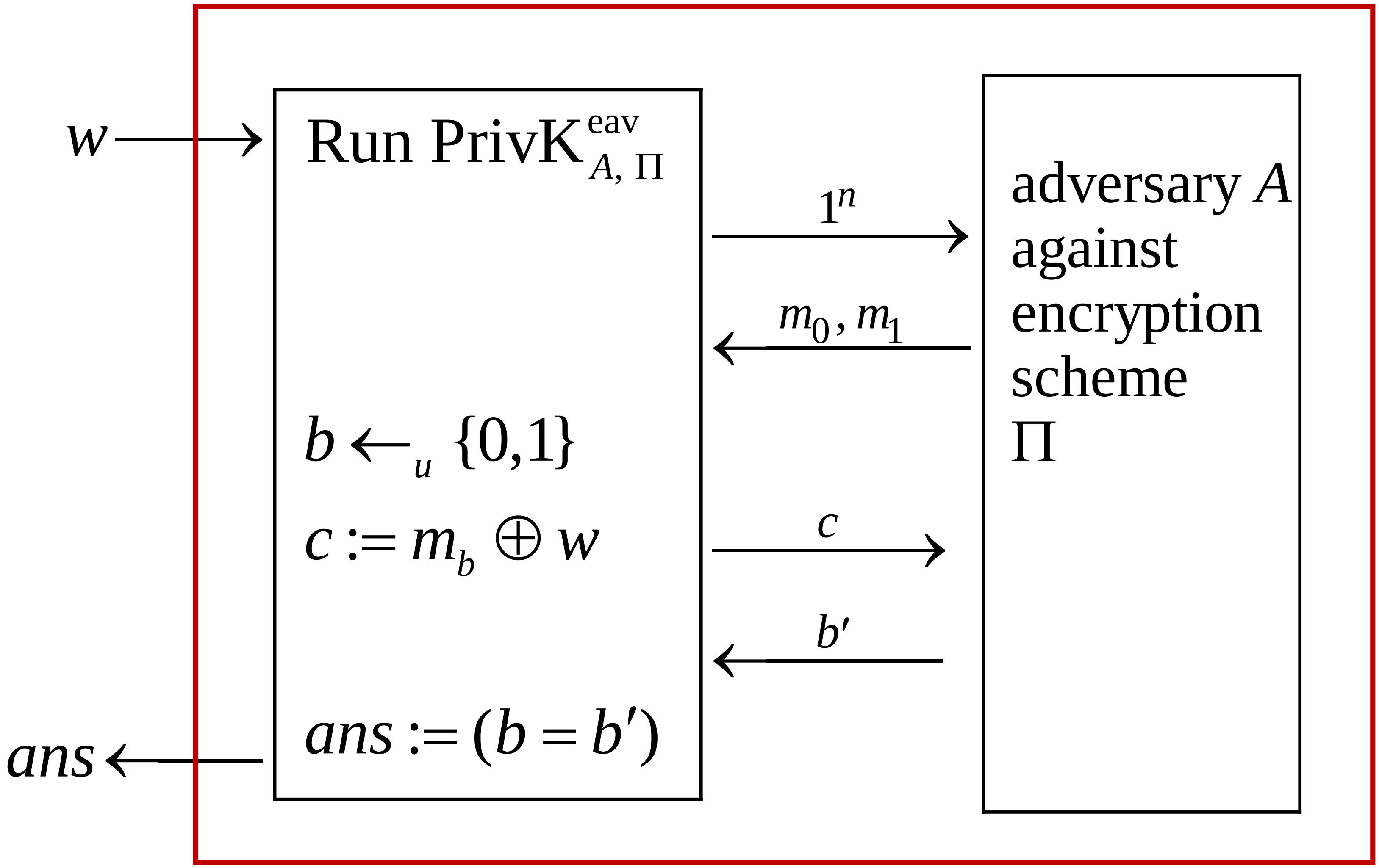
- By reduction. We will show:

Distinguishing between random strings r and pseudorandom strings $G(s)$	$\leq_P$	Breaking encryption scheme Π (distinguishing between ciphertexts c_0 and c_1)

- Notation. A $\leq_P$ B: A reduces to B in polynomial time.
- Roughly meaning that we can solve A using an algorithm for B as a subroutine. Hardness of A $\leq$ hardness of B.
- Example?

- Let A be an arbitrary PPT adversary against encryption scheme Π.
- Construct a distinguisher D:
  - D, given as input a string $w \in \{0,1\}^{l(n)}$, wants to determine whether w is random or pseudorandom.
  - D runs $\mathrm{PrivK}^{\mathrm{eav}}_{A,\Pi}(n)$ to obtain a pair of messages $m_0, m_1 \in \{0,1\}^{l(n)}$.
  - D chooses $b \leftarrow_u \{0,1\}$, sets $c := m_b \oplus w$, gives c to A, and obtains b' from A.
  - D outputs 1 if $b = b'$, and outputs 0 otherwise.

Distinguisher D



- $\Pr\left[D(w)=1:\ w \leftarrow_u \{0,1\}^{l(n)}\right] \;=\; \Pr\left[\mathrm{PrivK}^{\mathrm{eav}}_{A,\,\Pi^*}=1\right] \;=\; 1/2$

  where Π^* is Vernan's one-time pad.

- $\Pr\left[D(w)=1:\ w := G(s),\ s \leftarrow_u \{0,1\}^{n}\right] \;=\; \Pr\left[\mathrm{PrivK}^{\mathrm{eav}}_{A,\,\Pi}=1\right]$

- $$\left|\ \Pr\left[D(w)=1:\ w \leftarrow_u \{0,1\}^{l(n)}\right] - \Pr\left[D(w)=1:\ w := G(s),\ s \leftarrow_u \{0,1\}^{n}\right]\ \right| \le negl(n)$$ (Why?)

- So, $\left|\ 1/2 - \Pr\left[\mathrm{PrivK}^{\mathrm{eav}}_{A,\,\Pi}=1\right]\ \right| \le negl(n)$

  $\Rightarrow\ \Pr\left[\mathrm{PrivK}^{\mathrm{eav}}_{A,\,\Pi}=1\right] \le 1/2 + negl(n)$

  $\Rightarrow\ \Pi$ is EAV-secure

Encrypting multiple messages with a single key

- Stream ciphers require a new key for each message.
- In practice, Alice and Bob wish to share a permanent key k and use it to encrypt multiple messages. One possible strategy:
  - For each message m, generate a random string r and use $s = k \| r$ as a seed to the pseudorandom generator G.
  - Include r in the ciphertext, i.e., $c := Enc_k(m) := \left(r,\ m \oplus G(k \| r)\right)$.
  - It is probabilistic!
- Unfortunately, the resulting scheme is not necessarily EAV-secure. It requires G to be more than a pseudorandom generator for the scheme to be EAV-secure.

Using stream ciphers in a session

- At the beginning of a session, Alice and Bob agree on two keys k_1 and k_2 (called session keys).
- Alice and Bob each run $G(k_1)$ and $G(k_2)$ to get two (long enough) pseudorandom strings, say PS_1 and PS_2.
- Alice encrypts her sequence of messeges $\left(m_1, m_2, m_3, ...\right)$ as

$$\left(c_1, c_2, c_3, ...\right) := \left(\left(m_1, m_2, m_3, ...\right) \oplus PS_1\right).$$

- Bob uses PS_2 for encryption in a similar way.
- In practice, a stream cipher is designed to generate a random string of desired length bit/byte by bit/byte byte on demand.

The RC4 Stream Cipher (K&L: Section 6.1.4)

- Most popular stream cipher
- Simple and fast
- Used in many standards
- Actually not a cipher, but a (practical, approximate) pseudorandom generator. Not truely pseudorandom.
- Designed by Ron Rivest in 1987 for RSA Security, and kept as a trade secret until leaked out in 1994.

RC4

- Two vectors of bytes:
  - $S[0]$, $S[1]$, $S[2]$, …, $S[255]$
  - $T[0]$, $T[1]$, $T[2]$, …, $T[255]$
- Input Key (seed) K: variable length, 1 to 256 bytes
- Initialization:
  1. $S[i] \leftarrow i$, for $0 \le i \le 255$
  2. $T[0..255] \leftarrow K, K$, ... (until filled up)

RC4: Initial Permutation

- Initial Permutation of S:

$$j \leftarrow 0$$

$$\text{for } i \leftarrow 0 \text{ to } 255 \text{ do}$$

$$j \leftarrow (j + S[i] + T[i]) \bmod 256$$

$$\text{Swap } S[i], S[j]$$

- Idea: swapping bytes dependently of the input key.
- After this step, the input key will not be used.

RC4: Key Stream Generation

- Key stream generation:

$$i, j \leftarrow 0$$

while (true)

$$i \leftarrow (i + 1) \bmod 256$$

$$j \leftarrow (j + S[i]) \bmod 256$$

Swap $S[i], S[j]$

$$t \leftarrow (S[i] + S[j]) \bmod 256$$

output $S[t]$

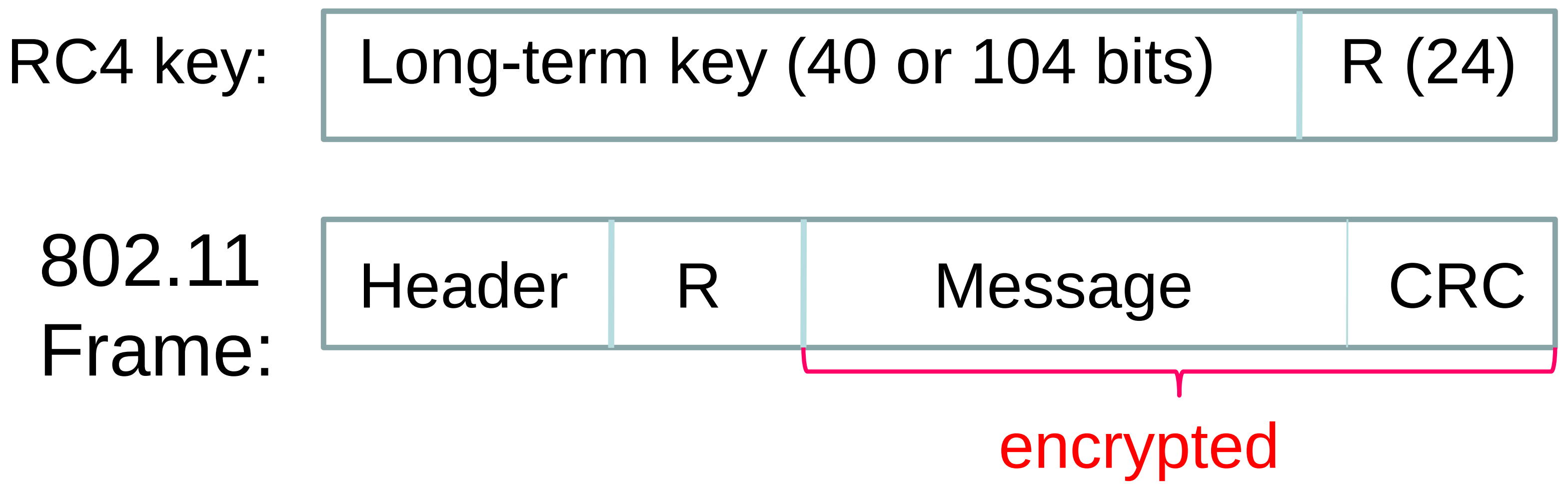
- Idea: systematically keep swapping and producing output bytes

Security of RC4

- RC4 is not a truly pseudorandom generator.
- The key stream generated by RC4 is biased.
  - The second byte is biased toward zero with high probability.
  - The first few bytes are strongly non-random and leak information about the input key.
- Defense: discard the initial *n* bytes of the keystream.
  - Called "RC4-drop[*n*-bytes]".
  - Recommended values for n = 256, 768, or 3072 bytes.
- Efforts are under way (e.g. the eSTREAM project) to develop more secure stream ciphers.

The Use of RC4 in WEP

- WEP is an RC4-based protocol for encrypting data transmitted over an IEEE 802.11 wireless LAN.
- WEP requires each packet to be encrypted with a separate RC4 key.
- The RC4 key for each packet is a concatenation of a 40-bit or 104-bit long-term key and a random 24-bit R.



WEP is not secure

- Mainly because of its way of constructing the key
- Can be cracked in a minute
- http://eprint.iacr.org/2007/120.pdf

Stronger Security Notions

K&L: Section 3.4

Different levels of security

- EAV-security (against eavedroppers, ciphertext-only-attacks)
  - one encryption
  - multiple encryptions
- CPA-security (against chosen-plaintext attacks)
  - one encryption
  - multiple encryptions
- CCA-security (against chosen-ciphertext attacks)
  - one encryption
  - multiple encryptions

Multiple-ciphertext indist. experiment $\mathrm{PrivK}^{\mathrm{mult}}_{A,\Pi}(n)$

- Adversary: eavesdropper with multiple ciphertexts
- A game between Bob and an adversary A:
  - The adversary, given input 1^n, selects two lists of messages $M_0 = (m_0^1,\ m_0^2,\ ...,\ m_0^t)$ and $M_1 = (m_1^1,\ m_1^2,\ ...,\ m_1^t)$ such that $|m_0^i| = |m_1^i|$ for all i.
  - Bob chooses a key $k \leftarrow Gen(1^n)$ and a bit $b \leftarrow_u \{0,1\}$; computes $c^i \leftarrow Enc_k(m_b^i)$ for all $i,$ and gives the challenge ciphertext list $C = (c^1,\ c^2,\ ...,\ c^t)$ to the adversary.
  - The adversary outputs a bit b'.
  - The output of the experiment is 1 iff $b = b'$.

Multiple-ciphertext indist. against an eavesdropper

- Definition: A private-key encryption scheme Π has indistinguishable multiple encryptions against an eavesdropper if for all PPT adversaries A, there is a negligible function $negl(n)$ such that (for all n)

$$\Pr\left[\mathrm{PrivK}_{A,\,\Pi}^{\mathrm{mult}}(n)=1\right] \le \frac{1}{2}+\mathrm{negl}(n)$$

  where the probability is taken over the randomness used by A, by Bob, by Gen, and by Enc.

- $$\Pr\left[\mathrm{PrivK}_{A,\,\Pi}^{\mathrm{mult}}(n)=1\right]=\Pr\left[\begin{array}{l}A(1^n,M_0,M_1,Enc_k(M_b))=b:\\ b\leftarrow_u \{0,1\},\ k\leftarrow Gen(1^n),\ M_0,M_1\leftarrow A(1^n)\end{array}\right]$$

Deterministic encryption schemes are not multiple-ciphertext indistinguishable

- Theorem: If the *Enc* of an encryption scheme $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ is deterministic, then the scheme cannot have indistinguishable multiple encryptions against an eavesdropper.

- Proof. Suppose *Enc* is deterministic.

  Let $M_0 = (0^n,\ 0^n)$ and $M_1 = (0^n,\ 1^n)$. Let the challenge ciphertext list be $C = (c_1,\ c_2)$.

  What can A say if $c_1 = c_2$ (or if $c_1 \neq c_2$)?

- For example, Vernam's one-time pad (for a fixed n) is single-ciphertext indistinguishable, but not multiple-ciphertext indistinguishable.

Chosen-Plaintext Attacks (CPA)

- The adversary is capable of adaptively obtaining samples $(m_1, c_1), \ldots, (m_t, c_t)$, where m_i is chosen by the adversary and $c_i \leftarrow Enc_k(m_i)$ for all i.
- We model such an adversary by giving it access to an encryption oracle $Enc_k(\cdot)$, viewed as a "black box" that on query m returns a ciphertext c $\leftarrow Enc_k(m)$.

$$m \rightarrow \quad \boxed{\text{Oracle } Enc_k(\cdot)}$$
$$Enc_k(m) \leftarrow$$

CPA indistinguishability experiment $\text{PrivK}^{\text{cpa}}_{A,\Pi}(n)$

1. A key $k \leftarrow Gen(1^n)$ is generated.
2. The adversary is given input 1^n and oracle access to $Enc_k(\cdot)$. It may request the oracle to encrypt messages of its choice.
3. The adversary chooses two message m_0, m_1 with $|m_0| = |m_1|$; and is given a challenge ciphertext $c \leftarrow Enc_k(m_b)$, where $b \leftarrow_u \{0,1\}$.
4. The adversary continues to have oracle access to $Enc_k(\cdot)$ and may even request the encryptions of m_0 and m_1.
5. The adversary finally outputs a bit b'.
6. The output of the experiment is 1 iff $b = b'$.

Note: The CPA here is an adaptive CPA.

CPA-security

- Definition: A private-key encryption scheme Π has indistinguishable encryptions under a chosen-plaintext attack, or is CPA-secure, if for all PPT adversaries A, there is a negligible function $negl(n)$ such that (for all n)

$$\Pr\left[\mathrm{PrivK}^{\mathrm{cpa}}_{A,\Pi}(n)=1\right] \le \frac{1}{2}+\mathrm{negl}(n)$$

where the probability is taken over the randomness used by A as well as the randomness used in the experiment.

- $$\Pr\left[\mathrm{PrivK}^{\mathrm{cpa}}_{A,\Pi}(n)=1\right]=\Pr\left[\begin{array}{l} A^{Enc_k(\cdot)}(1^n, m_0, m_1, Enc_k(m_b))=b: \\ b \leftarrow_u \{0,1\},\ k \leftarrow Gen(1^n),\ m_0, m_1 \leftarrow A(1^n) \end{array}\right]$$

CPA-security for multiple encryptions

- One approach is to model the adversary as having oracle access to $Enc_k(\cdot)$ and having it produce two message lists

  $M_0 = (m_0^1,\ m_0^2,\ \ldots,\ m_0^t)$ and $M_1 = (m_1^1,\ m_1^2,\ \ldots,\ m_1^t)$

- Alternatively, we use an oracle $\text{LR-Enc}_{k,b}(\cdot)$, where k is a key and $b \leftarrow \{0,1\}$. ($\text{LR-Enc}_{k,b}(\cdot)$ is denoted by $\text{LR}_{k,b}(\cdot)$ in the book.)

$$m_0,\ m_1 \ \rightarrow \quad \boxed{\text{Oracle LR-Enc}_{k,b}(\cdot)}$$
$$Enc_k(m_b) \ \leftarrow$$

The adversary is to guess the value of *b*.

The LR-oracle experiment $\text{PrivK}^{\text{LR-cpa}}_{A,\Pi}(n)$

1. A key $k \leftarrow Gen(1^n)$ is generated.
2. A bit $b \leftarrow_u \{0,1\}$ is chosen.
3. The adversary A is given input 1^n and oracle access to $\text{LR-Enc}_{k,b}(\cdot)$.
4. The adversary A outputs a bit b'.
5. The output of the experiment is 1 iff $b = b'$.

CPA-security for multiple encryptions

- Definition: A private-key encryption scheme Π has indistinguishable multiple encryptions under a chosen-plaintext attack, or is CPA-secure for multiple encryptions, if for all PPT adversaries A, there is a negligible function $negl(n)$ such that (for all n)

$$\Pr\left[\mathrm{PrivK}^{\text{LR-cpa}}_{A,\,\Pi}(n)=1\right] \leq \frac{1}{2}+\mathrm{negl}(n)$$

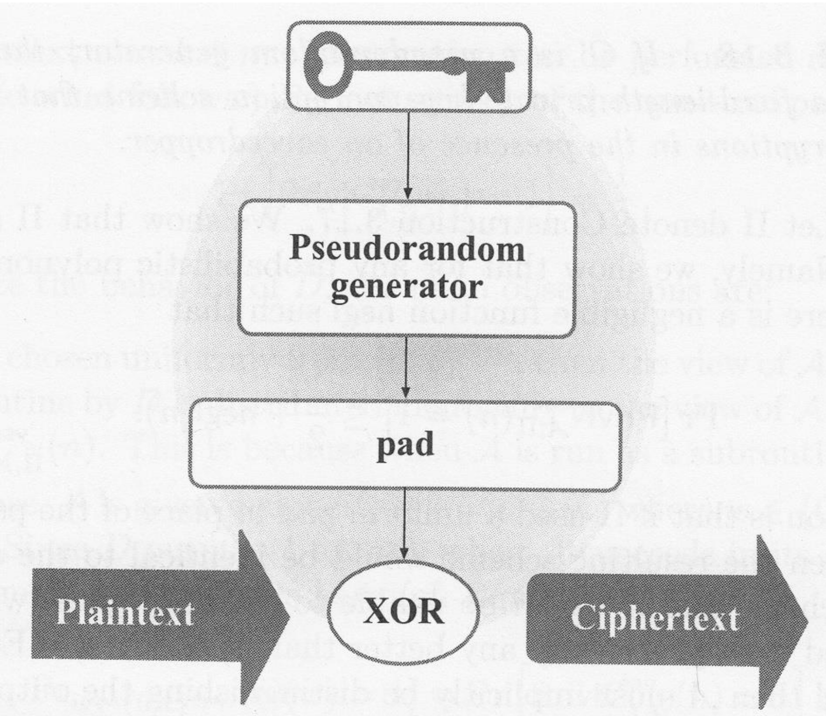
where the probability is taken over the randomness used by A as well as the randomness used in the experiment.

- Theorem: For any private-key encryption scheme,

  CPA-security $\Rightarrow$ CPA-security for multiple encryptions.

Constructing CPA-Secure Encryption Schemes

K&L: Section 3.5


Pseudorandom generator
pad
Plaintext
XOR
Ciphertext

A CPA-secure encryption scheme (inefficient)

- Let Func_n be the set of all functions $f:\{0,1\}^n \to \{0,1\}^n$.
- Construct an encryption scheme Π as follows.
- Key generation: uniformly choose a function $f \leftarrow_u \text{Func}_n$.
- To encrypt a message $m \in \{0,1\}^n$, uniformly choose a string $r \leftarrow_u \{0,1\}^n$, and encrypt m as $c := \langle r, m \oplus f(r) \rangle$.
- To decrypt a ciphertext $c = \langle r, s \rangle$, compute $m := s \oplus f(r)$.

- Theorem: The encryption scheme Π is CPA-secure.
- Proof (sketch). Consider any arbitrary adversary A. In the experiment $\mathrm{PrivK}^{\mathrm{cpa}}_{A,\Pi}(n)$, let $c := \langle \tilde{r}, m_b \oplus f(\tilde{r}) \rangle$ be the challenge ciphertext. Since $f(\tilde{r})$ is uniformly random, c *is* indistinguishable unless, on A's query m, the oracle happens to return $c_m := \langle \tilde{r}, m \oplus f(\tilde{r}) \rangle$, in which case A will learn $f(\tilde{r})$. This may occur with probability at most $\mathrm{poly}(n)/2^n$, where $\mathrm{poly}(n)$ is an upper bound on the number of queries A may make to the oracle. Thus,

$$\Pr\left[\mathrm{PrivK}^{\mathrm{cpa}}_{A,\Pi}(n) = 1\right] \leq \frac{1}{2} + \mathrm{poly}(n)/2^n = \frac{1}{2} + \mathrm{negl}(n).$$

- The secret key here is f. Q: What's its length?
- Suppose we label the elements/functions in Func_n with strings $k \in \{0,1\}^{\ell_{\text{key}}}$. What's the key length ℓ_{key}?
- How many elements/functions are there in Func_n?
  - View each function as a table of 2^n strings of length *n*.
  - There are 2 choices (0 or 1) for each of the $n \cdot 2^n$ bits.
  - So, there are $2^{n \cdot 2^n}$ different functions. I.e., $|\text{Func}_n| = 2^{n \cdot 2^n}$.
- Thus, $\ell_{\text{key}} \geq \log_2 2^{n \cdot 2^n} = n \cdot 2^n$, which is infeasible.

- Solution:
  - Choose a "small" subset of $Func_n$, say $Func'_n$, such that $Func_n$ and $Func'_n$ are indistinguishable.
  - Then, randomly picking a function from $Func'_n$ (as the key) will be almost as good as randomly picking a function from $Func_n$.
  - If we choose $Func'_n$ to contain no more than 2^n elements, the key length will be at most n.
  - We will describe $Func'_n$ (which is a set of functions) as a single function with two parameters, called a keyed function.

Keyed functions

- A keyed function $F: \{0,1\}^{a(n)} \times \{0,1\}^{b(n)} \to \{0,1\}^{c(n)}$ for all $n \geq 1$, has two inputs. The first one is called the key and denoted k.
- Each key $k \in \{0,1\}^{a(n)}$ induces a single-input function:

$$F_k : \{0,1\}^{b(n)} \to \{0,1\}^{c(n)}$$

$$F_k(x) = F(k, x)$$

- F is associated with three functions, $a(n)$, $b(n)$, $c(n)$ (often written as $l_{\text{key}}(n)$, $l_{\text{in}}(n)$, $l_{\text{out}}(n)$) which indicate the lengths of k, x, and $F_k(x)$.
- F is length-preserving if $l_{\text{key}}(n) = l_{\text{in}}(n) = l_{\text{out}}(n) = n$.
- If F is length-preserving, F induces a set of functions for each n:
  $\left\{ F_k : \{0,1\}^n \to \{0,1\}^n \mid k \in \{0,1\}^n \right\}$
- Q: In general, what set of functions does F induce?

Keyed Length-Preserving functions

- A keyed length-preserving function $F: \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ has two inputs. The first one is called the key and denoted k.
- Each key $k \in \{0,1\}^n$ induces a single-input function:

$$F_k : \{0,1\}^n \rightarrow \{0,1\}^n$$
$$F_k(x) = F(k,x)$$

- That is, F induces a set of functions for each n:

$$\left\{ F_k : \{0,1\}^n \rightarrow \{0,1\}^n \mid k \in \{0,1\}^n \right\}$$

Pseudorandom functions

- Let F be a keyed length-preserving function.
- Recall $\mathrm{Func}_n =$ the set of all functions $f:\{0,1\}^n \to \{0,1\}^n$.
- F is a pseudorandom function if the two ensembles of sets

$$\left(\left\{F_k \mid k \in \{0,1\}^n\right\}\right)_{n\in\mathbb{N}} \quad \text{and} \quad \left(\mathrm{Func}_n\right)_{n\in\mathbb{N}}$$

are polynomially indistinguishable, i.e., if for every PPT distinguisher D, it holds:

$$\left| \Pr\left[D^{F_k(\cdot)}(1^n) = 1 :\ k \leftarrow_u \{0,1\}^n\right] - \Pr\left[D^{f(\cdot)}(1^n) = 1 :\ f \leftarrow_u \mathrm{Func}_n\right] \right| \le \mathrm{negl}(n)$$

General pseudorandom functions

- Let $F: \{0,1\}^{l_{\text{key}}(n)} \times \{0,1\}^{l_{\text{in}}(n)} \to \{0,1\}^{l_{\text{out}}(n)}$ be a keyed function.
- Define $\overline{\text{Func}}_n =$ the set of all functions $f: \{0,1\}^{l_{\text{in}}(n)} \to \{0,1\}^{l_{\text{out}}(n)}$.
- F is a pseudorandom function if the two ensembles of sets

$$\left(\left\{F_k \mid k \in \{0,1\}^{l_{\text{key}}(n)}\right\}\right)_{n\in\mathbb{N}} \quad \text{and} \quad \left(\overline{\text{Func}}_n\right)_{n\in\mathbb{N}}$$

are polynomially indistinguishable, i.e., if for every PPT distinguisher D, it holds:

$$\left| \Pr\left[D^{F_k(\cdot)}(1^n)=1:\ k \leftarrow_u \{0,1\}^{l_{\text{key}}(n)}\right] - \Pr\left[D^{f(\cdot)}(1^n)=1:\ f \leftarrow_u \overline{\text{Func}}_n\right] \right| \leq \text{negl}(n)$$

Example keyed length-preserving function

- Suppose $F(k,x) = k \oplus x.$
- Then, $F_k(x) = k \oplus x.$
- Is F a pseudorandom function?
- For any k and x, $F_k(x) \oplus F_k(\overline{x}) = \left(k \oplus x\right) \oplus \left(k \oplus \overline{x}\right) = 1^n.$
- Based on this, we design a distinguisher D as follows.

  Given a function h (as an oracle), D asks the oracle to compute $h(x)$ and $h(\overline{x})$ for some $x \in \{0,1\}^n$, say $x = 0^n$. If $h(x) \oplus h(\overline{x}) = 1^n$, D returns 1, else returns 0. We have

$$\Pr\left[D^{F_k(\cdot)}(1^n) = 1 : \ k \leftarrow_u \{0,1\}^n \right] = 1$$

$$\Pr\left[D^{f(\cdot)}(1^n) = 1 : \ f \leftarrow_u \mathrm{Func}_n \right] = 2^{-n}$$

$$\Pr\left[D^{f(\cdot)}(1^n) = 1 : \ f \leftarrow_u \mathrm{Func}_n \right]$$

$$= \sum_f \Pr\left[f \text{ is picked} \right] \cdot \Pr\left[D^{f(\cdot)}(1^n) = 1 \right]$$

$$= \frac{1}{2^{n2^n}} \cdot \sum_f \Pr\left[f(x) \oplus f(\overline{x}) = 1^n \right]$$

$$= \frac{1}{2^{n2^n}} \cdot \frac{2^{n2^n}}{2^n}$$

$$= \frac{1}{2^n}$$

Permutations

- A function $f : X \to X$ is called a permutation if it is bijective (one-to-one and onto).
- We are interested in permutations $f : \{0,1\}^{l(n)} \to \{0,1\}^{l(n)}$, especially with $l(n) = n$.

Pseudorandom permutations

- A keyed permutation is a keyed function F for which each F_k is a permutation.
- Perm_n, the set of all permutations $f:\{0,1\}^n \rightarrow \{0,1\}^n$.
- A length-preserving keyed permutation F is a pseudorandom permutation if for every PPT distinguisher D, it holds:

$$\left| \Pr\left[D^{F_k(\cdot)}(1^n) = 1 : k \leftarrow_u \{0,1\}^n \right] - \Pr\left[D^{f(\cdot)}(1^n) = 1 : f \leftarrow_u \text{Perm}_n \right] \right| \leq \text{negl}(n)$$

- Theorem: A pseudorandom permutation is also a pseudorandom function (assuming $l(n) \geq n$).

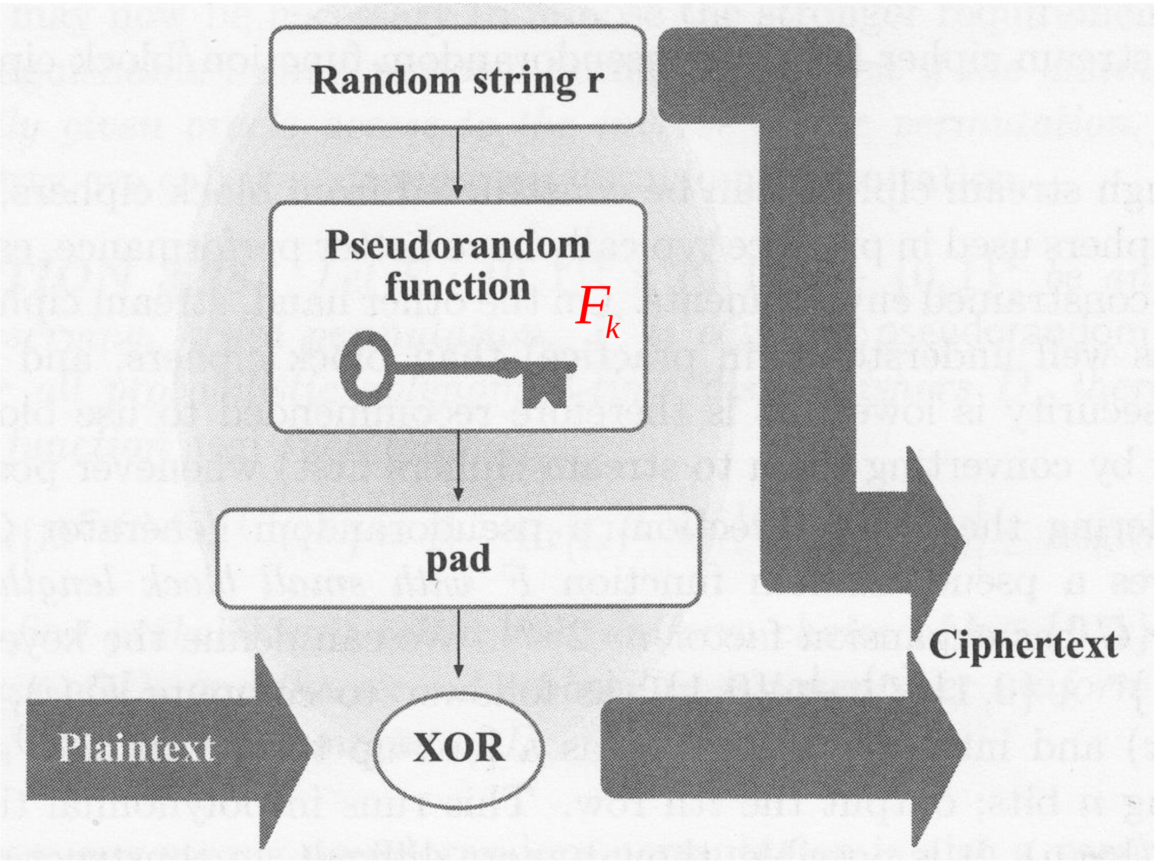
CPA-secure encryption using pseudorandom functions

- Let F be a pseudorandom function. Construct an encryption scheme Π for messages of length n as follows.

- Gen: on input 1^n, output a key $k \leftarrow_u \{0,1\}^n$.

- Enc: on input a key $k \in \{0,1\}^n$ and a message $m \in \{0,1\}^n$, choose uniformly a string $r \leftarrow_u \{0,1\}^n$ and output the ciphertext $c := \langle r,\ F_k(r) \oplus m \rangle$.

- Dec: on input a key $k \in \{0,1\}^n$ and a ciphertext $c = \langle r, s \rangle$, output the plaintext message $m := F_k(r) \oplus s$.


Random string r
Pseudorandom function
F_k
pad
Plaintext
XOR
Ciphertext

- Theorem: The encryption scheme Π is CPA-secure.
- Proof (basic idea).
  - In scheme Π, a function $f \in Func_n$ is used as a key.
  - In scheme Π, a function $F_k \in \left\{F_k : k \in \{0,1\}^n\right\}$ is used as a key.
  - Since $Func_n$ and $\left\{F_k : k \in \{0,1\}^n\right\}$ are indistinguishable, it can be shown by reduction that

$$\left|\Pr\left[\mathrm{PrivK}^{\mathrm{cpa}}_{A,\Pi}(n)=1\right] - \Pr\left[\mathrm{PrivK}^{\mathrm{cpa}}_{A,\Pi}(n)=1\right]\right| \leq \mathrm{negl}(n)$$

  - We already know

$$\Pr\left[\mathrm{PrivK}^{\mathrm{cpa}}_{A,\Pi}(n)=1\right] \leq \frac{1}{2}+\mathrm{negl}(n).$$

  - Thus, $\Pr\left[\mathrm{PrivK}^{\mathrm{cpa}}_{A,\Pi}(n)=1\right] \leq \frac{1}{2}+\mathrm{negl}(n).$

If F is a pseudorandom permutation

- Since F is also a pseudorandom function, we may encrypt a message $m \in \{0,1\}^n$ as before:

  1) $c := \langle r,\ F_k(r) \oplus m \rangle$, where $r \leftarrow_u \{0,1\}^n$. //CPA-secure//

- If $F_k^{-1}(m)$ is efficiently computable, we may also encrypt m as

  2) $c := F_k(m)$ //deterministic, so not CPA-secure//

  3) $c := \langle r,\ F_k(r \oplus m) \rangle$, where $r \leftarrow_u \{0,1\}^n$. //CPA-secure//

  Q: How to decrypt a ciphertext $c = \langle r,\ s \rangle$?

  (Assume that F_k is efficiently computable.)

Modes of Operations

K&L: Section 3.6.2

Encrypting long messages

- Now let's see how to encrypt a message of arbitrary length using a pseudorandom function or permutation.
- Encryption algorithm: On input $m \in \{0,1\}^*$ and key k,
  - Pad the message so that its length is a multiple of n (block size).
  - Divide the padded message m into blocks, say

$$m = (m_1, m_2, m_3, \ldots, m_t)$$

  - Individually encrypt each block m_i:

$$r_i \leftarrow_u \{0,1\}^n \quad \text{and} \quad c_i := F_k(r_i) \oplus m_i$$

  - The final ciphertext is

$$c := \langle (r_1, c_1), (r_2, c_2), \ldots, (r_t, c_t) \rangle$$

- The ciphertext is twice as long as the message. Inefficient!

Modes of operation

- More efficient ways to do it are traditionaly called modes of operation (of block ciphers).
- Main idea: generate a single random string $IV \leftarrow_u \{0,1\}^n$ and derive $r_1, r_2, \ldots, r_t$ from IV. (IV: Initialization Vector)
- The ciphertext will be of the form

$$c = \langle IV, c_1, c_2, \ldots, c_t \rangle$$

- Important modes of operation:
  - Counter mode (CTR): $r_i = IV + i$
  - Output feedback mode (OFB): $r_1 = IV, \; r_i = F_k(r_{i-1})$
  - Cipher feedback mode (CFB): $c_0 = IV, \; r_i := c_{i-1}$
  - Cipher block chaining mode (CBC): $c_0 = IV, \; r_i := c_{i-1}$

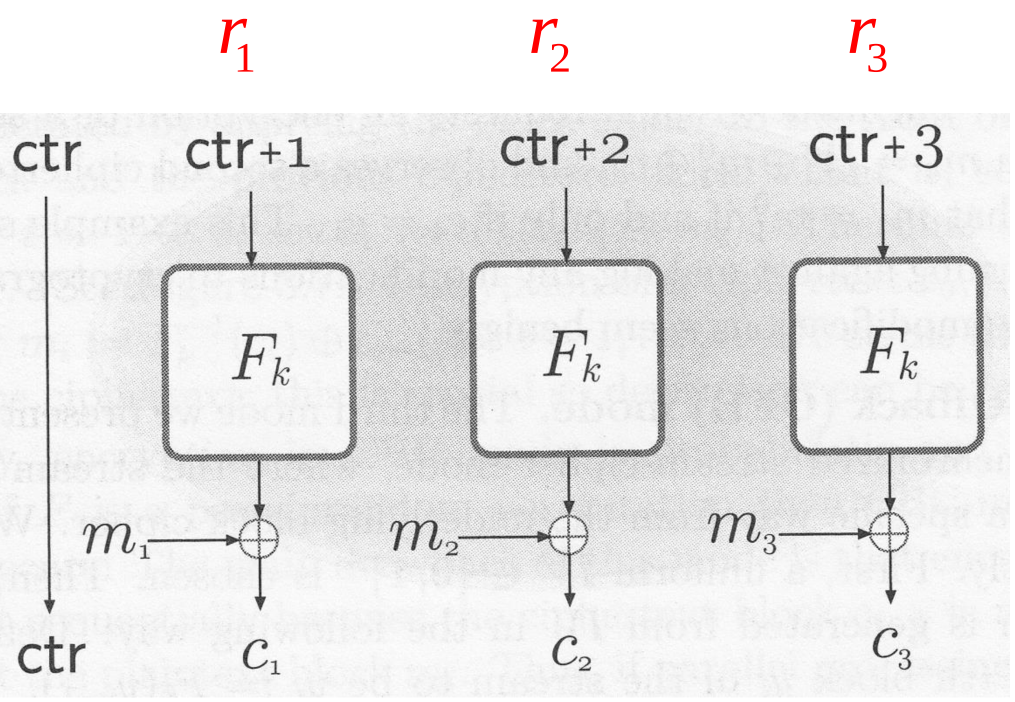
Counter mode (CTR)

- Idea: The strings $r_1, r_2, \ldots, r_t$ are $r_i = IV + i$ for $1 \le i \le t$.
- Thus, to encrypt a message $m = (m_1, m_2, m_3, \ldots, m_t)$ with key k
  - Choose a random string $IV \leftarrow_u \{0,1\}^n$.
  - Encrypt m as

$$c := \langle IV,\ c_1,\ c_2,\ \ldots,\ c_t \rangle, \text{ where } c_i := F_k(r_i) \oplus m_i$$
$$r_i := IV + i$$

- Strength: Blocks can be encrypted (or decrypted) in parallel or in a "random access" fashion.

Counter Mode (CTR)



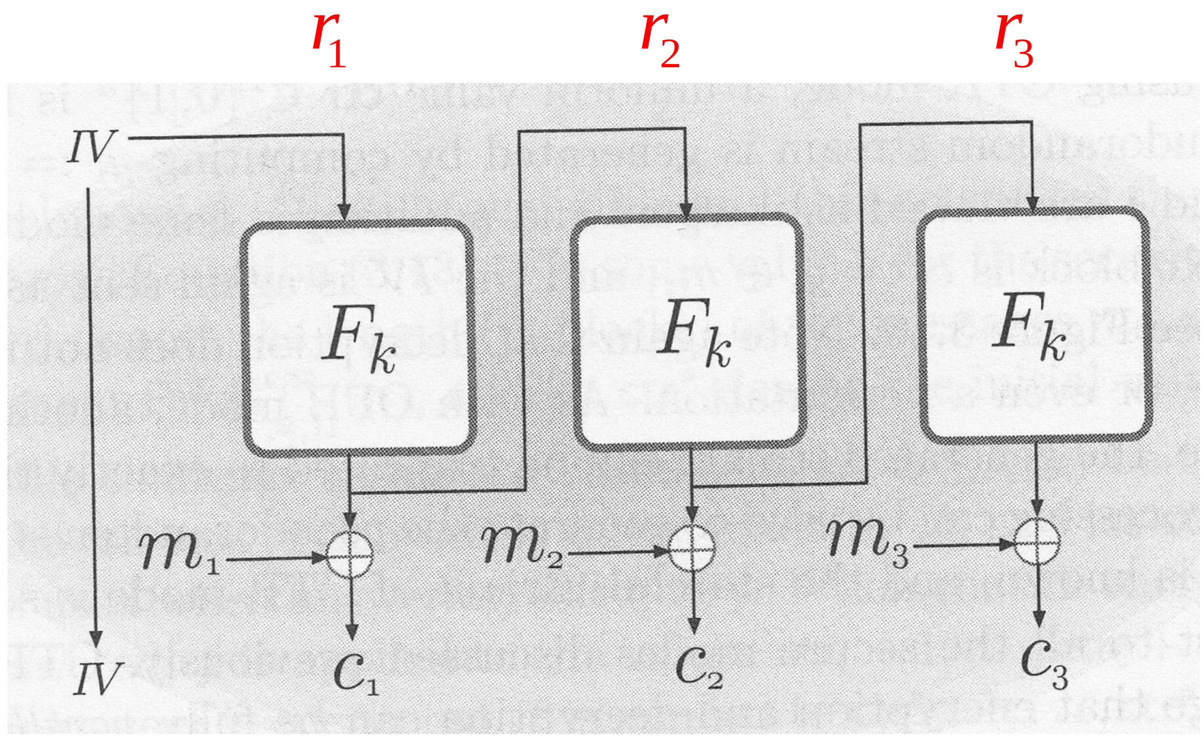
Output feedback mode (OFB)

- Idea: The strings $r_1, r_2, \ldots, r_t$ are $r_1 = IV$ and $r_i = F_k(\, r_{i-1})$
- Thus, to encrypt a message $m = \left(m_1, m_2, m_3, \ldots, m_t\right)$ with key k
  - Choose a random string $IV \leftarrow_u \{0,1\}^n$.
  - Encrypt m as $c := \left\langle IV, c_1, c_2, \ldots, c_t \right\rangle$

    where $c_i := F_k(r_i) \oplus m_i$

    $r_1 := IV$, and $r_i := F_k(r_{i-1})$ for $2 \le i \le t$

Output feedback



Cipher feedback mode (CFB)

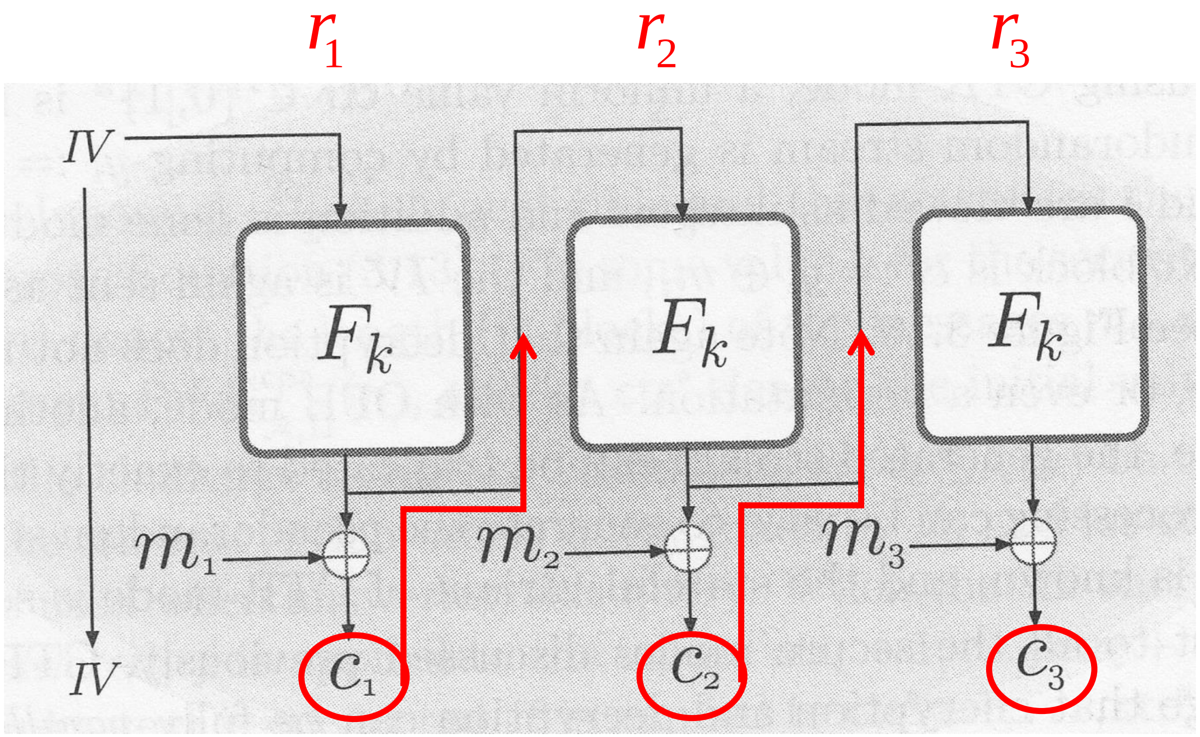
- Idea: The strings $r_1, r_2, \ldots, r_t$ are chosen to be $r_i := c_{i-1}$, where $c_0 = IV$ and c_{i-1} is the previous cipher block.
- Thus, the ciphertext of $m = (m_1, m_2, m_3, \ldots, m_t)$ is

  $c := (c_0, c_1, c_2, \ldots, c_t)$

  where $c_0 := IV$

  $$c_i := F_k(c_{i-1}) \oplus m_i \text{ for } 1 \le i \le t.$$

How is Cipher Feedback (CFB) different from OFB?



Cipher block chaining mode (CBC)

- Assume F is a pseudorandom permutation and F_k^{-1} is efficiently computable.

- Each block m_i is encrypted as $c_i = F_k\left(r_i \oplus m_i\right)$.

- The strings $r_1, r_2, \ldots, r_t$ are chosen to be $r_i = c_{i-1}$ for $1 \le i \le t$, with $c_0 = IV$, and c_{i-1} being the previous cipher block.

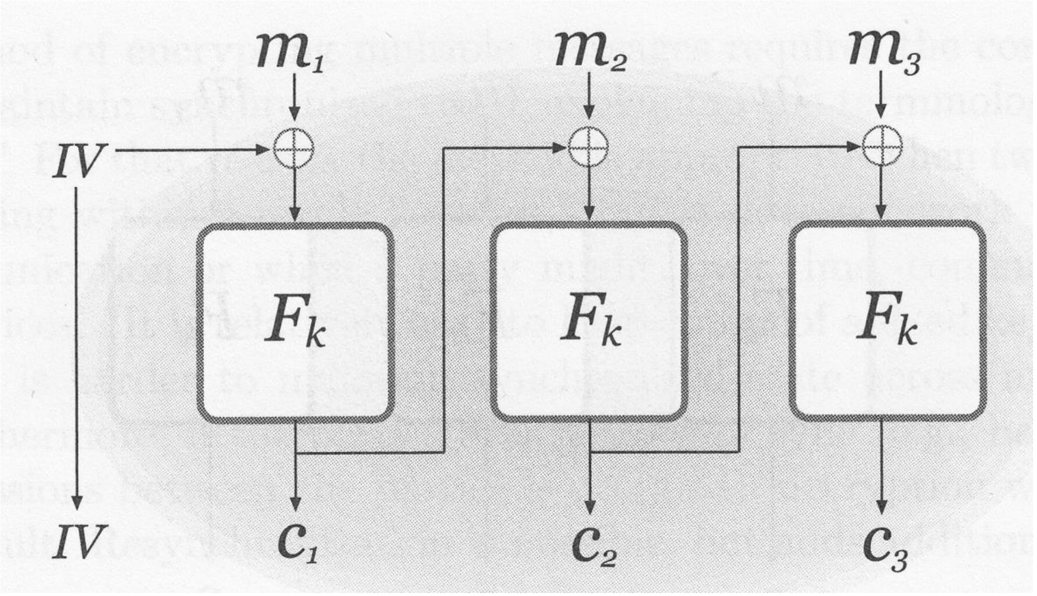
- Thus, the ciphertext of $m = \left(m_1, m_2, m_3, \ldots, m_t\right)$ is

$$c := \left(c_0, c_1, c_2, \ldots, c_t\right)$$

where $c_0 := IV$

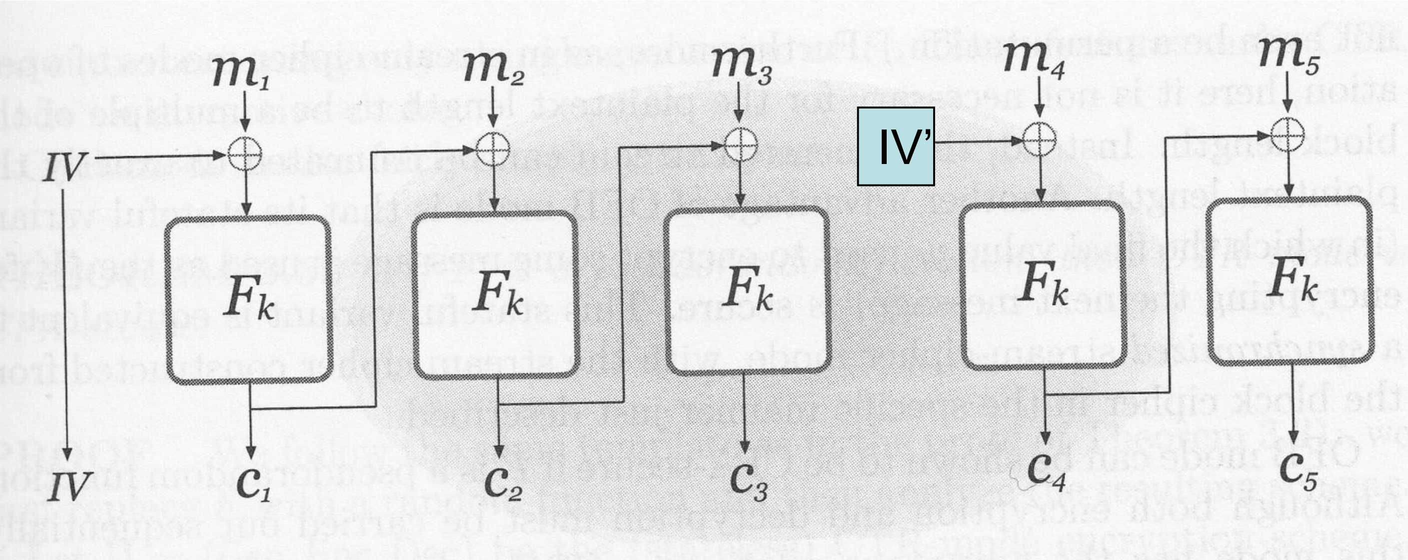
$$c_i := F_k(c_{i-1} \oplus m_i) \text{ for } 1 \le i \le t.$$

Cipher block chaining (CBC)



CBC

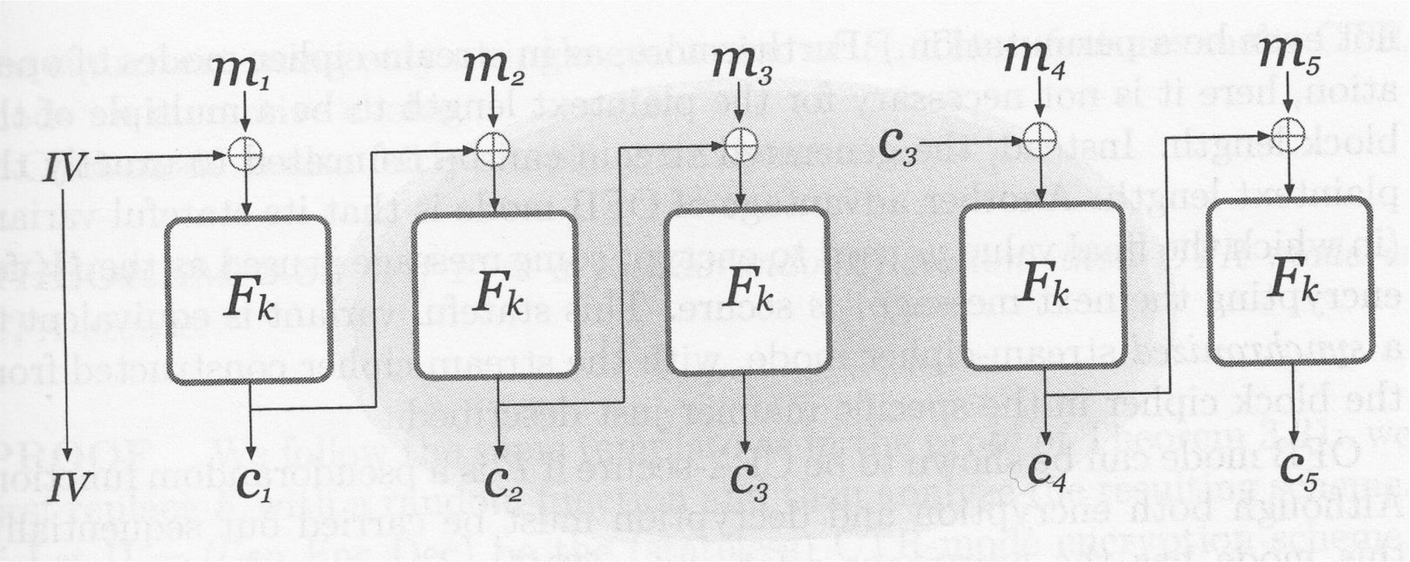
Message 1: (m_1, m_2, m_3) Message 2: (m_4, m_5)



Chained CBC

- Used in SSL 3.0 and TLS 1.0, but is not CPA-secure.

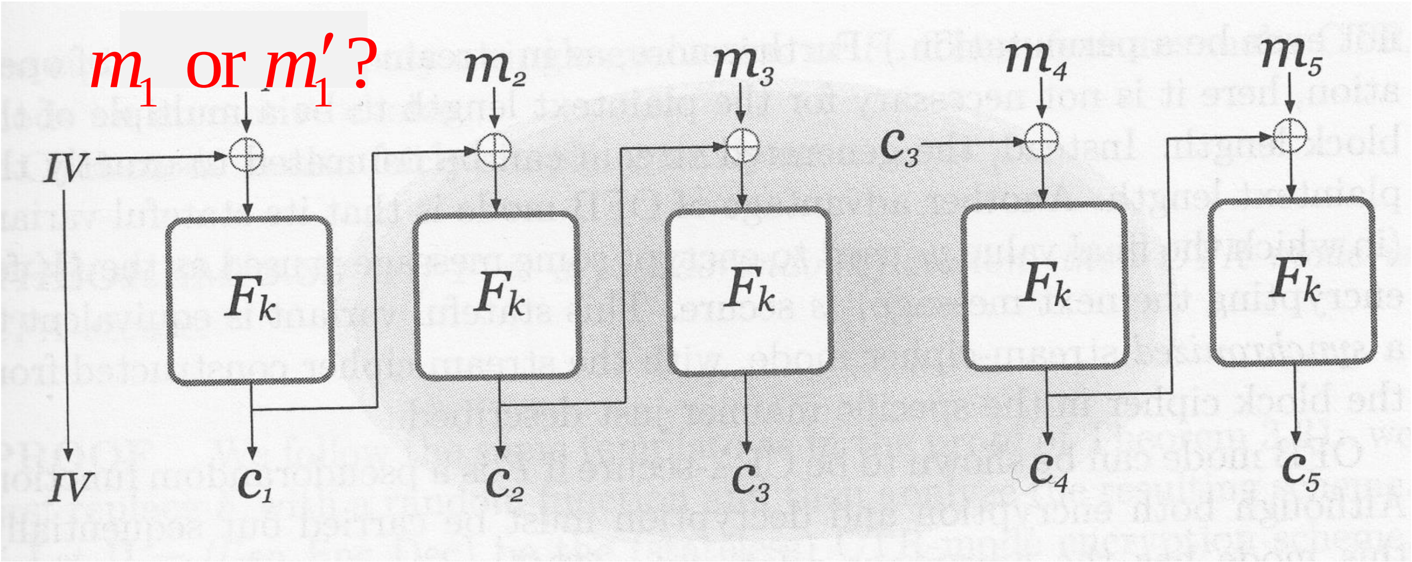
Message 1: (m_1, m_2, m_3) Message 2: (m_4, m_5)



Insecurity of Chained CBC

- Let adversary A chooses two messages $M = (m_1, m_2, m_3)$, $M' = (m_1', m_2, m_3)$ such that $m_1 \neq m_1'$.
- Let $C = (IV, c_1, c_2, c_3)$ be the challenge ciphertext.
- A knows the oracle is going to use c_3 in the next encryption. So, A prepares m_4 such that $IV \oplus m_1 = c_3 \oplus m_4$, and asks the oracle to encrypt it. Suppose A receives c_4 from the oracle.
- Depending on whether $c_1 = c_4$, A knows whether C is the encryption of M or M'.

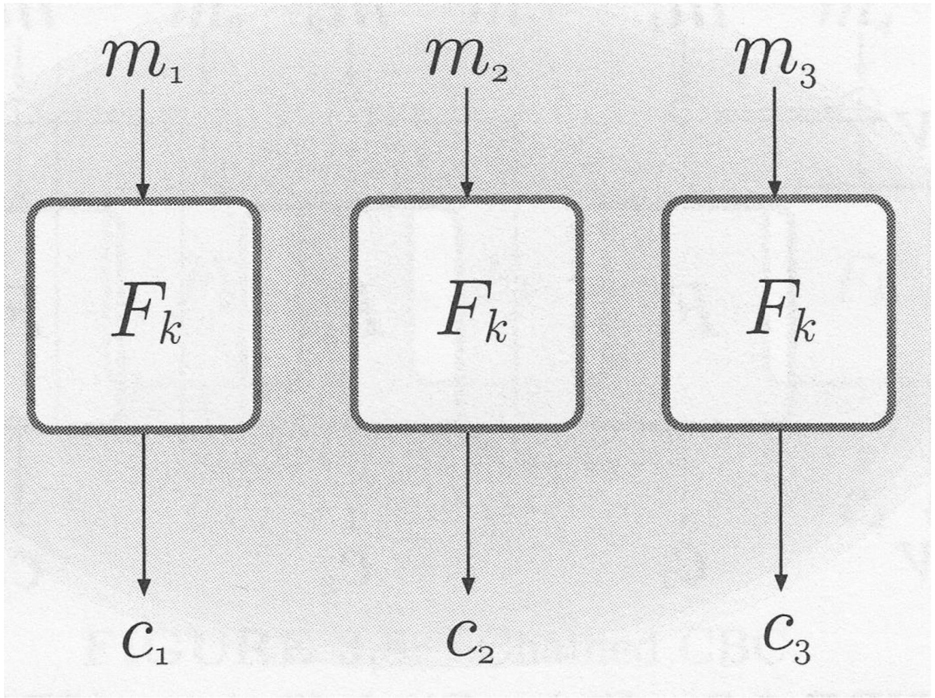
Is $C = (IV,\ c_1,\ c_2,\ c_3)$ the encryption of $M = (m_1,\ m_2,\ m_3)$ or $M' = (m_1',\ m_2,\ m_3)$?



Electronic codebook mode (ECB)

- Use a pseudorandom permutation F.
- $m = (m_1,\ m_2,\ m_3,\ \ldots,\ m_t)$
- Each block m_i is encrypted as $c_i = F_k(m_i)$.
- The resulting scheme is deterministic and not CPA secure.
- Used only for sending a short message (in a single block).

Electronic Code Book (ECB)



Security of CBC, OFB, CFB, CTR

- If F is a pseudorandom function or permutation, then OFB, CFB, CTR are CPA-secure.

- If F is a pseudorandom permutation, then CBC is CPA-secure.

Chosen-Ciphertext Attacks

K&L Section 3.7

CCA indistinguishability experiment $\mathrm{PrivK}^{\mathrm{cca}}_{A,\Pi}(n)$

1. A key $k \leftarrow Gen(1^n)$ is generated.
2. The adversary is given input 1^n and oracle access to $Enc_k(\cdot)$ and $Dec_k(\cdot)$.
3. The adversary chooses two message $m_0,\ m_1$ with $|m_0| = |m_1|$; and is given a challenge ciphertext $c \leftarrow Enc_k(m_b)$, where $b \leftarrow_u \{0,1\}$.
4. The adversary continues to have oracle access to $Enc_k(\cdot)$ and $Dec_k(\cdot)$, but is not allowed to request the decryption of c itself.
5. The adversary finally outputs a bit b'.
6. The output of the experiment is 1 iff $b = b'$.

Note: The CCA defined here has the capabilities of both CPA and "pure CCA".

CCA-security

- Definition: A private-key encryption scheme Π has indistinguishable encryptions under a chosen-ciphertext attack, or is CCA-secure, if for all PPT adversaries A, there is a negligible function $negl(n)$ such that (for all n)

$$\Pr\left[\mathrm{PrivK}_{A,\Pi}^{\mathrm{cca}}(n)=1\right] \leq \frac{1}{2}+\mathrm{negl}(n)$$

where the probability is taken over the randomness used by A as well as the randomness used in the experiment.

- $\Pr\left[\mathrm{PrivK}_{A,\Pi}^{\mathrm{cca}}(n)=1\right]=\Pr\left[\begin{array}{l} A^{Enc_k(\cdot),Dec_k(\cdot)}(1^n,m_0,m_1,Enc_k(m_b))=b: \\ b\leftarrow_u \{0,1\},\ k\leftarrow Gen(1^n),\ m_0,m_1\leftarrow A(1^n) \end{array}\right]$

CCA-security for multiple encryptions

- Experiment $\text{PrivK}^{\text{LR-cca}}_{A,\Pi}(n)$: same as $\text{PrivK}^{\text{LR-cpa}}_{A,\Pi}(n)$ except ... (what?)
- Definition: A private-key encryption scheme Π has indistinguishable multiple encryptions under a chosen-ciphertext attack, or is CCA-secure for multiple encryptions, if for all PPT adversaries *A*, there is a negligible function *negl*(n) such that (for all n)

$$\Pr\left[\text{PrivK}^{\text{LR-cca}}_{A,\Pi}(n)=1\right] \le \frac{1}{2}+\text{negl}(n)$$

where the probability is taken over the randomness used by *A* as well as the randomness used in the experiment.

- Theorem: For any private-key encryption scheme,

CCA-security $\Rightarrow$ CCA-security for multiple encryptions.

CCA insecurity

- The encryption schemes we have seen so far are not CCA-secure.
- If a ciphertext $c \leftarrow Enc_k(m)$ can be manipulated in a controlled way, then the encryption scheme is not CCA-secure.
- Example: consider the scheme $Enc_k(m) \leftarrow \left(r,\ F_k(r) \oplus m\right)$.
  - The adversary chooses any two messages m_0, m_1 of equal length.
  - Let the challenge ciphertext be $\langle r,\ c \rangle$ where $c := F_k(r) \oplus m_b$, with $b \in \{0,1\}$.
  - The adversary modifies $\langle r,\ c \rangle$ to $\langle r,\ \bar{c} \rangle = \langle r,\ f_k(r) \oplus \bar{m}_b \rangle$, which is a legitimate ciphertext of $\bar{m}_b$.
  - Requesting the oracle to decrypt $\langle r,\ \bar{c} \rangle$, the adversary will get $\bar{m}_b$ and hence know the value of b.

Constructing a CCA-secure encryption scheme

- We will see that:

CPA-secure encryption + secure MAC

$\Rightarrow$ CCA-secure encryption

Padding-Oracle Attack:
a concrete example of (partial) chosen-ciphertext attacks

K&L Section 3.7.2

The Setting

- We will attack the CBC-mode encryption scheme that uses PKCS#5 padding.
- L: block length (in bytes).
- b: pad length (in bytes). $1 \le b \le L \le 255$
- PKCS#5 padding:
  - The value of b (as an 8-bit binary) is repeated b times.
  - Examples: 0x01, 0x0202, 0x030303, 0x04040404.
- Message refers to the original message (w/o padding).
- Encoded data refers to the padded message.
- The encoded data is encrypted using CBC-mode encryption.

A Padding Oracle

- On receiving a ciphertext, the receiver decrypts it to recover the encoded data and checks if the padding is correct.
- If not correct, the receiver typically sends back a "bad padding" error message (e.g., in Java, javax.crypto.BadPaddingException).
- Such receivers provide the adversary with a padding oracle which may be viewed as a partial decryption oracle.

ciphertext $\rightarrow$ Padding Oracle

error (if padding incorrect) $\leftarrow$

- Using such a padding oracle, the adversary can recover the original message.

Modify the encoded data in a controlled fashion

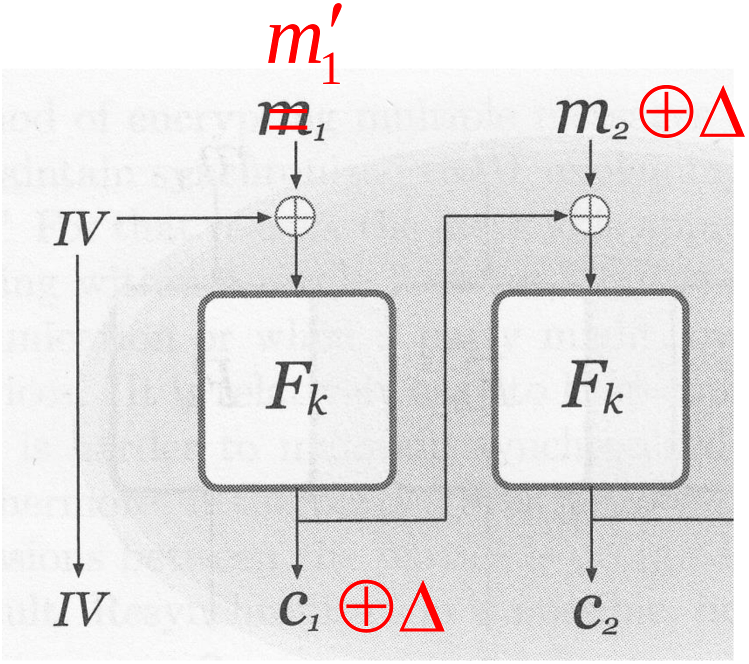
- Suppose the encoded data is $\langle m_1, m_2 \rangle$, unknown to the adversary; and the ciphertext is $\langle IV, c_1, c_2 \rangle$, known to the adversary.
- Recall: $c_2 = F_k(m_2 \oplus c_1)$ and so $m_2 = F_k^{-1}(c_2) \oplus c_1$.
- Thus, $m_2 \oplus \Delta = F_k^{-1}(c_2) \oplus c_1 \oplus \Delta$. That is,

$$\langle IV, c_1, c_2 \rangle \xrightarrow{Dec} \langle m_1, m_2 \rangle$$

$$\langle IV, c_1 \oplus \Delta, c_2 \rangle \xrightarrow{Dec} \langle m_1', m_2 \oplus \Delta \rangle$$

- By modifying the ciphertext, the adversary can modify the encoded data in a controlled fashion and then ask the oracle if the padding (of the modified encoded data) is correct.

Cipher block chaining (CBC)



Find out the pad length b

- Example: modifying the 5th byte will result in a padding error.

m_2 =

0x33	0x22	0x11	0x44	0x03	0x03	0x03

- In general, to find the pad length, the adversary runs:

for $i \leftarrow 1$ **to** L **do**

modify the ith byte of c_1

send the resulting ciphertext to the receiver/oracle

if receiving a padding error then return $b := L - (i - 1)$

Recover the message byte by byte

- Having known $b = 3$, how to recover the byte w?

m_2 =	x	y	z	w	0x03	0x03	0x03
m_2' =	x	y	z	$w \oplus i$	0x04	0x04	0x04

  - Try (how?) every string $i \in \{0,1\}^8$ until there is no padding error,

    for which i, $\quad w \oplus i = 0\text{x}04 \;\Rightarrow\; w = 0\text{x}04 \oplus i$

  - How: modify c_1 to $c_1 \oplus \Delta_i$, with $\Delta_i = 0^8 0^8 0^8 i \left(0\text{x}03 \oplus 0\text{x}04\right)^3$

    and present the resulting ciphertext $\langle IV,\ c_1 \oplus \Delta_i,\ c_2 \rangle$ to the oracle, which after decryption will see $\langle m_1',\ m_2' \rangle$.

Recover the message byte by byte

- Having recovered *w*, how to recover *z*?

$m_2 =$

x	y	z	w	0x03	0x03	0x03

$m_2' =$

x	y	$z \oplus i$	0x05	0x05	0x05	0x05

  - Try every string $i \in \{0,1\}^8$ until no padding error, then

$$z \oplus i = \text{0x05} \;\Rightarrow\; z = \text{0x05} \oplus i$$

  - How: modify c_1 to $c_1 \oplus \Delta_i$, with $\Delta_i = 0^8 0^8 i \left(w \oplus \text{0x05}\right)\left(\text{0x03} \oplus \text{0x05}\right)^3$ and present the resulting ciphertext $\langle IV,\ c_1 \oplus \Delta_i,\ c_2 \rangle$ to the oracle, which after decryption will see $\langle m_1',\ m_2' \rangle$.