

# Lecture 3

## Perfect Secrecy, Information Theory, and Luby-Rackoff

Douglas Wikström  
KTH Stockholm  
dog@csc.kth.se

February 16

- Perfect Secrecy
- Information Theory
- Luby-Rackoff

# Perfect Secrecy (1/3)

When is a cipher perfectly secure?

# Perfect Secrecy (1/3)

When is a cipher perfectly secure?

How should we formalize this?

## Perfect Secrecy (2/3)

**Definition.** A cryptosystem has perfect secrecy if guessing the plaintext is as hard to do given the ciphertext as it is without it.

## Perfect Secrecy (2/3)

**Definition.** A cryptosystem has perfect secrecy if guessing the plaintext is as hard to do given the ciphertext as it is without it.

**Definition.** A cryptosystem has perfect secrecy if

$$\Pr[M = m | C = c] = \Pr[M = m]$$

for every  $m \in \mathcal{M}$  and  $c \in \mathcal{C}$ , where  $M$  and  $C$  are random variables taking values over  $\mathcal{M}$  and  $\mathcal{C}$ .

## Perfect Secrecy (3/3)

**Game Based Definition.**  $\text{Exp}_A^b$ , where  $A$  is a strategy:

1.  $k \leftarrow_R \mathcal{K}$
2.  $(m_0, m_1) \leftarrow A$
3.  $c = E_k(m_b)$
4.  $d \leftarrow A(c)$ , with  $d \in \{0, 1\}$
5. Output  $d$ .

**Definition.** A cryptosystem has perfect secrecy if for every **computationally unbounded** strategy  $A$ ,

$$\Pr [\text{Exp}_A^0 = 1] = \Pr [\text{Exp}_A^1 = 1] \quad .$$

# One-Time Pad

## One-Time Pad (OTP).

- ▶ **Key.** Random tuple  $k = (b_0, \dots, b_{n-1}) \in \mathbb{Z}_2^n$ .
- ▶ **Encrypt.** Plaintext  $m = (m_0, \dots, m_{n-1}) \in \mathbb{Z}_2^n$  gives ciphertext  $c = (c_0, \dots, c_{n-1})$ , where  $c_i = m_i \oplus b_i$ .
- ▶ **Decrypt.** Ciphertext  $c = (c_0, \dots, c_{n-1}) \in \mathbb{Z}_2^n$  gives plaintext  $m = (m_0, \dots, m_{n-1})$ , where  $m_i = c_i \oplus b_i$ .



# Bayes' Theorem

**Theorem.** If  $A$  and  $B$  are events and  $\Pr[B] > 0$ , then

$$\Pr[A|B] = \frac{\Pr[A] \Pr[B|A]}{\Pr[B]}$$

## Terminology:

$\Pr[A]$  – prior probability of  $A$

$\Pr[B]$  – prior probability of  $B$

$\Pr[A|B]$  – posterior probability of  $A$  given  $B$

$\Pr[B|A]$  – posterior probability of  $B$  given  $A$

# One-Time Pad Has Perfect Secrecy

- ▶ **Probabilistic Argument.** Bayes implies that:

$$\begin{aligned}\Pr[M = m | C = c] &= \frac{\Pr[M = m] \Pr[C = c | M = m]}{\Pr[C = c]} \\ &= \Pr[M = m] \frac{2^{-n}}{2^{-n}} \\ &= \Pr[M = m] .\end{aligned}$$

- ▶ **Simulation Argument.** The ciphertext is uniformly and independently distributed from the plaintext. We can **simulate** it on our own!

# Bad News

- ▶ **“The key must be as long as the plaintext”:**
  - ▶  $|\mathcal{K}| \geq |\mathcal{C}|$ , since every ciphertext must have been created using some key.
  - ▶  $|\mathcal{C}| \geq |\mathcal{M}|$ , since every plaintext must be encrypted in an invertible way.

**Theorem.** “For every cipher with perfect secrecy, the key requires at least as much space to represent as the plaintext.”

- ▶ **Dangerous in practice to rely on no reuse.**

# Information Theory

- ▶ Information theory is a mathematical theory of communication.
- ▶ Typical questions studied are how to compress, transmit, and store information.
- ▶ Information theory is also useful to argue about some cryptographic schemes and protocols.

# Classical Information Theory

- ▶ **Memoryless Source Over Finite Alphabet.** A source produces symbols from an alphabet  $\Sigma = \{a_1, \dots, a_n\}$ . Each generated symbol is identically and independently distributed.
- ▶ **Binary Channel.** A binary channel can (only) send bits.
- ▶ **Coder/Decoder.** Our goal is to come up with a scheme to:
  1. convert a symbol  $a$  from the alphabet  $\Sigma$  into a sequence  $(b_1, \dots, b_l)$  of bits,
  2. send the bits over the channel, and
  3. decode the sequence into  $a$  again at the receiving end.

# Classical Information Theory



Alice

Bob

# Optimization Goal

We want to minimize the **expected** number of bits/symbol we send over the binary channel, i.e., if  $X$  is a random variable over  $\Sigma$  and  $l(x)$  is the length of the codeword of  $x$  then we wish to minimize

$$\mathbb{E} [l(X)] = \sum_{x \in \Sigma} P_X(x) l(x) .$$

## Examples:

- ▶  $X$  takes values in  $\Sigma = \{a, b, c, d\}$  with uniform distribution.  
How would you encode this?



## Examples:

- ▶  $X$  takes values in  $\Sigma = \{a, b, c, d\}$  with uniform distribution.  
How would you encode this?

It seems we need  $I(x) = \log |\Sigma|$ . This gives the Hartley measure.

## Examples:

- ▶  $X$  takes values in  $\Sigma = \{a, b, c, d\}$  with uniform distribution. How would you encode this?
- ▶  $X$  takes values in  $\Sigma = \{a, b, c\}$ , with  $P_X(a) = \frac{1}{2}$ ,  $P_X(b) = \frac{1}{4}$ , and  $P_X(c) = \frac{1}{4}$ . How would you encode this?

It seems we need  $I(x) = \log |\Sigma|$ . This gives the Hartley measure.

**hmmm...**

## Examples:

- ▶  $X$  takes values in  $\Sigma = \{a, b, c, d\}$  with uniform distribution. How would you encode this?
- ▶  $X$  takes values in  $\Sigma = \{a, b, c\}$ , with  $P_X(a) = \frac{1}{2}$ ,  $P_X(b) = \frac{1}{4}$ , and  $P_X(c) = \frac{1}{4}$ . How would you encode this?

It seems we need  $I(x) = \log \frac{1}{P_X(x)}$  bits to encode  $x$ .

# Entropy

Let us turn this expression into a definition.

**Definition.** Let  $X$  be a random variable taking values in  $\mathcal{X}$ . Then the **entropy** of  $X$  is

$$H(X) = - \sum_{x \in \mathcal{X}} P_X(x) \log P_X(x) .$$

Examples and intuition are nice, but what we need is a theorem that states that this is **exactly** the right expected length of an optimal code.

# Jensen's Inequality

**Definition.** A function  $f : \mathcal{X} \rightarrow (a, b)$  is **concave** if

$$\lambda \cdot f(x) + (1 - \lambda)f(y) \leq f(\lambda \cdot x + (1 - \lambda)y) \text{ ,}$$

for every  $x, y \in (a, b)$  and  $0 \leq \lambda \leq 1$ .

# Jensen's Inequality

**Definition.** A function  $f : \mathcal{X} \rightarrow (a, b)$  is **concave** if

$$\lambda \cdot f(x) + (1 - \lambda)f(y) \leq f(\lambda \cdot x + (1 - \lambda)y) \ ,$$

for every  $x, y \in (a, b)$  and  $0 \leq \lambda \leq 1$ .

**Theorem.** Suppose  $f$  is continuous and strictly concave on  $(a, b)$ , and  $X$  is a discrete random variable. Then

$$\mathbb{E}[f(X)] \leq f(\mathbb{E}[X]) \ ,$$

with equality iff  $X$  is constant.

# Jensen's Inequality

**Definition.** A function  $f : \mathcal{X} \rightarrow (a, b)$  is **concave** if

$$\lambda \cdot f(x) + (1 - \lambda)f(y) \leq f(\lambda \cdot x + (1 - \lambda)y) \ ,$$

for every  $x, y \in (a, b)$  and  $0 \leq \lambda \leq 1$ .

**Theorem.** Suppose  $f$  is continuous and strictly concave on  $(a, b)$ , and  $X$  is a discrete random variable. Then

$$\mathbb{E}[f(X)] \leq f(\mathbb{E}[X]) \ ,$$

with equality iff  $X$  is constant.

**Proof idea.** Consider two points + induction over number of points.

# Kraft's Inequality

**Theorem.** There exists a prefix-free code  $E$  with codeword lengths  $l_x$ , for  $x \in \Sigma$  if and only if

$$\sum_{x \in \Sigma} 2^{-l_x} \leq 1 .$$

**Proof Sketch.**  $\Rightarrow$  Given a prefix-free code, we consider the corresponding binary tree with codewords at the leaves. We may “fold” it by replacing two neighboring leaves  $E(x)$  and  $E(y)$  by  $(xy)$  with length  $l_x - 1$ . Repeat.

$\Leftarrow$  Given lengths  $l_{x_1} \leq l_{x_2} \leq \dots \leq l_{x_n}$  we start with the complete binary tree of depth  $l_{x_n}$  and prune it.



# Binary Source Coding Theorem (1/2)

**Theorem.** Let  $E$  be an optimal code and let  $l(x)$  be the length of the codeword of  $x$ . Then

$$H(X) \leq E[l(X)] < H(X) + 1 .$$

# Binary Source Coding Theorem (1/2)

**Theorem.** Let  $E$  be an optimal code and let  $l(x)$  be the length of the codeword of  $x$ . Then

$$H(X) \leq E[l(X)] < H(X) + 1 .$$

## Proof of Upper Bound.

Define  $l_x = \lceil -\log P_X(x) \rceil$ . Then we have

$$\sum_{x \in \Sigma} 2^{-l_x} \leq \sum_{x \in \Sigma} 2^{\log P_X(x)} = \sum_{x \in \Sigma} P_X(x) = 1$$

Kraft's inequality implies that there is a code with codeword lengths  $l_x$ . Then note that

$$\sum_{x \in \Sigma} P_X(x) \lceil -\log P_X(x) \rceil < H(X) + 1.$$

## Binary Source Coding Theorem (2/2)

**Proof of Lower Bound.**

$$\begin{aligned} \mathbb{E}[I(X)] &= \sum_x P_X(x) I(x) \\ &= - \sum_x P_X(x) \log 2^{-I(x)} \\ &\geq - \sum_x P_X(x) \log P_X(x) \\ &= H(X) \end{aligned}$$

# Huffman's Code (1/2)

**Input:**  $\{(a_1, p_1), \dots, (a_n, p_n)\}$ .

**Output:** 0/1-labeled rooted tree.

HUFFMAN( $\{(a_1, p_1), \dots, (a_n, p_n)\}$ )

- (1)  $S \leftarrow \{(a_1, p_1, a_1), \dots, (a_n, p_n, a_n)\}$
- (2) **while**  $|S| \geq 2$
- (3)     Find  $(b_i, p_i, t_i), (b_j, p_j, t_j) \in S$  with minimal  $p_i$  and  $p_j$ .
- (4)      $S \leftarrow S \setminus \{(b_i, p_i, t_i), (b_j, p_j, t_j)\}$
- (5)      $S \leftarrow S \cup \{(b_i \| b_j, p_i + p_j, \text{NODE}(t_i, t_j))\}$
- (6) **return**  $S$

## Huffman's Code (2/2)

**Theorem.** Huffman's code is optimal.

**Proof idea.**

There exists an optimal code where the two least likely symbols are neighbors.

# Conditional Entropy

**Definition.** Let  $(X, Y)$  be a random variable taking values in  $\mathcal{X} \times \mathcal{Y}$ . We define **conditional entropy**

$$H(X|y) = - \sum_x P_{X|Y}(x|y) \log P_{X|Y}(x|y) \quad \text{and}$$
$$H(X|Y) = \sum_y P_Y(y) H(X|y)$$

Note that  $H(X|y)$  is simply the ordinary entropy function of a random variable with probability function  $P_{X|Y}(\cdot|y)$ .

# Properties of Entropy

Let  $X$  be a random variable taking values in  $\mathcal{X}$ .

**Upper Bound.**  $H(X) = \mathbb{E}[-\log P_X(X)] \leq \log |\mathcal{X}|$ .

**Chain Rule and Conditioning.**

$$\begin{aligned} H(X, Y) &= - \sum_{x,y} P_{X,Y}(x,y) \log P_{X,Y}(x,y) \\ &= - \sum_{x,y} P_{X,Y}(x,y) (\log P_Y(y) + \log P_{X|Y}(x|y)) \\ &= - \sum_y P_Y(y) \log P_Y(y) - \sum_{x,y} P_{X,Y}(x,y) \log P_{X|Y}(x|y) \\ &= H(Y) + H(X|Y) \leq H(Y) + H(X) \end{aligned}$$

# Negligible Functions

**Definition.** A function  $\epsilon(n)$  is negligible if for every constant  $c > 0$ , there exists a constant  $n_0$ , such that

$$\epsilon(n) < \frac{1}{n^c}$$

for all  $n \geq n_0$ .

**Motivation.** Events happening with negligible probability can not be exploited by polynomial time algorithms! (they “never” happen)



# Pseudo-Random Function

**“Definition”**. A function is pseudo-random if no efficient adversary can distinguish between the function and a random function.

# Pseudo-Random Function

**“Definition”**. A function is pseudo-random if no efficient adversary can distinguish between the function and a random function.

**Definition.** A family of functions  $F : \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$  is pseudo-random if for all polynomial time oracle adversaries  $A$

$$\left| \Pr_K \left[ A^{F_K(\cdot)} = 1 \right] - \Pr_{R: \{0,1\}^n \rightarrow \{0,1\}^n} \left[ A^{R(\cdot)} = 1 \right] \right|$$

is negligible.

# Pseudo-Random Permutation

**“Definition”**. A permutation and its inverse is pseudo-random if no efficient adversary can distinguish between the permutation and its inverse, and a random permutation and its inverse.

# Pseudo-Random Permutation

**“Definition”**. A permutation and its inverse is pseudo-random if no efficient adversary can distinguish between the permutation and its inverse, and a random permutation and its inverse.

**Definition.** A family of permutations  $P : \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$  are pseudo-random if for all polynomial time oracle adversaries  $A$

$$\left| \Pr_K \left[ A^{P_K(\cdot), P_K^{-1}(\cdot)} = 1 \right] - \Pr_{\Pi \in \mathcal{S}_{2^n}} \left[ A^{\Pi(\cdot), \Pi^{-1}(\cdot)} = 1 \right] \right|$$

is negligible, where  $\mathcal{S}_{2^n}$  is the set of permutations of  $\{0, 1\}^n$ .

# Idealized Four-Round Feistel Network

**Definition.** Feistel round (H for “Horst Feistel”).

$$H_{F_K}(L, R) = (R, L \oplus F(R, K))$$

# Idealized Four-Round Feistel Network

**Definition.** Feistel round (H for “Horst Feistel”).

$$H_{F_K}(L, R) = (R, L \oplus F(R, K))$$

**Theorem.** (Luby and Rackoff) If  $F$  is a pseudo-random family of functions, then

$$H_{F_{k_1}, F_{k_2}, F_{k_3}, F_{k_4}}(x) = H_{F_{k_4}}(H_{F_{k_3}}(H_{F_{k_2}}(H_{F_{k_1}}(x))))$$

(and its inverse) is a pseudo-random family of permutations.

# Idealized Four-Round Feistel Network

**Definition.** Feistel round (H for “Horst Feistel”).

$$H_{F_K}(L, R) = (R, L \oplus F(R, K))$$

**Theorem.** (Luby and Rackoff) If  $F$  is a pseudo-random family of functions, then

$$H_{F_{k_1}, F_{k_2}, F_{k_3}, F_{k_4}}(x) = H_{F_{k_4}}(H_{F_{k_3}}(H_{F_{k_2}}(H_{F_{k_1}}(x))))$$

(and its inverse) is a pseudo-random family of permutations.

Why do we need four rounds?