Lecture 5
Ciphers, Information Theory, and Elementary Number Theory

Douglas Wikström
KTH Stockholm
dog@csc.kth.se

February 13, 2015

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 $\{0,1\}^n$.

Perfect Secrecy

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

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

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. Exp_A^b, where A is a strategy:

1. $\mathsf{k} \leftarrow_R \mathcal{K}$
2. $(m_0, m_1) \leftarrow A$
3. $c = \mathsf{E}_\mathsf{k}(m_b)$
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,

$$\Pr\left[\mathrm{Exp}_A^0 = 1\right] = \Pr\left[\mathrm{Exp}_A^1 = 1\right] \quad .$$

One-Time Pad

One-Time Pad (OTP).

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

Bayes' Theorem

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

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

Terminology:

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

One-Time Pad Has Perfect Secrecy

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

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

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

Bad News

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 of, e.g., file containing randomness!

Information Theory

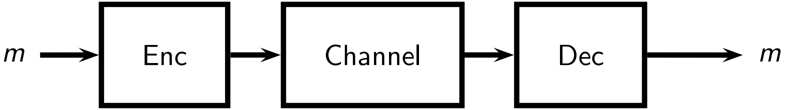
Information Theory

- 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 $\Sigma = \{a_1, \ldots, a_n\}$. Each generated symbol is 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 Σ into a sequence $(b_1, \ldots, b_l)$ 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 Σ and $l(x)$ is the length of the codeword of x then we wish to minimize

$$\mathrm{E}\left[l(X)\right] = \sum_{x \in \Sigma} \mathsf{P}_X\left(x\right) l(x) \enspace .$$

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 $\mathrm{P}_X(a) = \frac{1}{2}$, $\mathrm{P}_X(b) = \frac{1}{4}$, and $\mathrm{P}_X(c) = \frac{1}{4}$. 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 $\mathrm{P}_X(a) = \frac{1}{2}$, $\mathrm{P}_X(b) = \frac{1}{4}$, and $\mathrm{P}_X(c) = \frac{1}{4}$. How would you encode this?

It seems we need $I(x) = \log \frac{1}{\mathrm{P}_X(x)}$ 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

$$H(X) = -\sum_{x \in \mathcal{X}} \mathrm{P}_X(x) \log \mathrm{P}_X(x) \quad .$$

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.

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 sibling leaves $\mathsf{E}(x)$ and $\mathsf{E}(y)$ by (xy) with length $l_x - 1$. Repeat.

$\Leftarrow$ Given lengths $l_{x_1} \leq l_{x_2} \leq \ldots \leq l_{x_n}$ 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

$$H(X) \leq \mathrm{E}\left[l(X)\right] < H(X) + 1 \enspace .$$

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

$$H(X) \leq \mathrm{E}\left[l(X)\right] < H(X) + 1 \enspace .$$

Proof of Upper Bound.
Define $l_x = \lceil -\log \mathrm{P}_X\left(x\right) \rceil$. Then we have

$$\sum_{x\in\Sigma} 2^{-l_x} \leq \sum_{x\in\Sigma} 2^{\log \mathrm{P}_X(x)} = \sum_{x\in\Sigma} \mathrm{P}_X\left(x\right) = 1$$

Kraft's inequality implies that there is a code with codeword lengths l_x. Then note that
$\sum_{x\in\Sigma} \mathrm{P}_X\left(x\right) \lceil -\log \mathrm{P}_X\left(x\right) \rceil < H(X) + 1$.

Binary Source Coding Theorem (2/2)

Proof of Lower Bound.

$$\begin{aligned}
\mathrm{E}\left[l(X)\right] &= \sum_x \mathrm{P}_X\left(x\right) l_x \\
&= -\sum_x \mathrm{P}_X\left(x\right) \log 2^{-l_x} \\
&\geq -\sum_x \mathrm{P}_X\left(x\right) \log \mathrm{P}_X\left(x\right) \\
&= H(X)
\end{aligned}$$

Huffman's Code (1/2)

Input: $\{(a_1, p_1), \ldots, (a_n, p_n)\}$.
Output: 0/1-labeled rooted tree.

$\textsc{Huffman}(\{(a_1, p_1), \ldots, (a_n, p_n)\})$
(1) $S \leftarrow \{(a_1, p_1, a_1), \ldots, (a_n, p_n, a_n)\}$
(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) $S \leftarrow S \cup \{\big(b_i \| b_j, p_i + p_j, \textsc{Node}(t_i, t_j)\big)\}$
(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**

$$H(X|y) = -\sum_{x} \mathrm{P}_{X|Y}(x|y) \log \mathrm{P}_{X|Y}(x|y) \quad \text{and}$$

$$H(X|Y) = \sum_{y} \mathrm{P}_{Y}(y) H(X|y)$$

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

Properties of Entropy

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

Upper Bound. $H(X) = \mathrm{E}\left[-\log \mathsf{P}_X(X)\right] \leq \log|\mathcal{X}|$.

Chain Rule and Conditioning.

$$\begin{aligned}
H(X,Y) &= -\sum_{x,y} \mathsf{P}_{X,Y}(x,y) \log \mathsf{P}_{X,Y}(x,y) \\
&= -\sum_{x,y} \mathsf{P}_{X,Y}(x,y) \left(\log \mathsf{P}_Y(y) + \log \mathsf{P}_{X|Y}(x|y) \right) \\
&= -\sum_{y} \mathsf{P}_Y(y) \log \mathsf{P}_Y(y) - \sum_{x,y} \mathsf{P}_{X,Y}(x,y) \log \mathsf{P}_{X|Y}(x|y) \\
&= H(Y) + H(X|Y) \leq H(Y) + H(X)
\end{aligned}$$

Greatest Common Divisors

Definition. A common divisor of two integers m and n is an integer d such that $d \mid m$ and $d \mid n$.

Definition. A greatest common divisor (GCD) of two integers m and n is a common divisor d such that every common divisor d' divides d.

Greatest Common Divisors

Definition. A common divisor of two integers m and n is an integer d such that $d \mid m$ and $d \mid n$.

Definition. A greatest common divisor (GCD) of two integers m and n is a common divisor d such that every common divisor d' divides d.

- **The** GCD is the **positive** GCD.

Greatest Common Divisors

Definition. A common divisor of two integers m and n is an integer d such that $d \mid m$ and $d \mid n$.

Definition. A greatest common divisor (GCD) of two integers m and n is a common divisor d such that every common divisor d' divides d.

- **The** GCD is the **positive** GCD.
- We denote the GCD of m and n by $\gcd(m, n)$.

Properties

- $\gcd(m, n) = \gcd(n, m)$
- $\gcd(m, n) = \gcd(m \pm n, n)$
- $\gcd(m, n) = \gcd(m \bmod n, n)$
- $\gcd(m, n) = 2\gcd(m/2, n/2)$ if m and n are even.
- $\gcd(m, n) = \gcd(m/2, n)$ if m is even and n is odd.

Euclidean Algorithm

```
EUCLIDEAN(m, n)
    while n ≠ 0
        t ← n
        n ← m mod n
        m ← t
    return m
```

Steins Algorithm (Binary GCD Algorithm)

```
Stein(m, n)
    if m = 0 or n = 0 then return 0
    s ← 0
    while m and n are even
        m ← m/2, n ← n/2, s ← s + 1
    while n is even
        n ← n/2
    while m ≠ 0
        while m is even
            m ← m/2
        if m < n
            Swap(m, n)
        m ← m − n
        m ← m/2
    return 2^s n
```

Bezout's Lemma

Lemma. There exists integers a and b such that

$$\gcd(m,n) = am + bn \quad .$$

Bezout's Lemma

Lemma. There exists integers a and b such that

$$\gcd(m, n) = am + bn \ .$$

Proof. Let $d > \gcd(m, n)$ be the smallest positive integer on the form $d = am + bn$. Write $m = cd + r$ with $0 < r < d$. Then

$$d > r = m - cd = m - c(am + bn) = (1 - ca)m + (-cb)n \ ,$$

a contradiction! Thus, $r = 0$ and $d \mid m$. Similarly, $d \mid n$.

Extended Euclidean Algorithm (Recursive Version)

```
ExtendedEuclidean(m, n)
    if m mod n = 0
        return (0, 1)
    else
        (x, y) ← ExtendedEuclidean(n, m mod n)
        return (y, x − y⌊m/n⌋)
```

If $(x, y) \leftarrow \textsc{ExtendedEuclidean}(m, n)$ then $\gcd(m, n) = xm + yn$.

Coprimality (Relative Primality)

Definition. Two integers m and n are coprime if their greatest common divisor is 1.

Fact. If a and n are coprime, then there exists a b such that $ab = 1 \bmod n$.

Coprimality (Relative Primality)

Definition. Two integers m and n are coprime if their greatest common divisor is 1.

Fact. If a and n are coprime, then there exists a b such that $ab = 1 \bmod n$.

Excercise: Why is this so?