Information Theory and Networks

Lecture 30: Cryptography and Information Theor

Matthew Roughan
<matthew.roughan@adelaide.edu.au>
http://www.maths.adelaide.edu.au/matthew.roughan/Lecture_notes/InformationTheory/

School of Mathematical Sciences,
University of Adelaide

October 29, 2013

Part I

Cryptography and Information Theory

Section 1

Public and Private Key Cryptography

Private-Key Cryptography

Symmetric Cryptography, is so named as the keys used to encrypt and decrypt a message are the same.

Anyone who knows the encryption key can decrypt a message, so it must be kept private hence its also called Private-Key Cryptography

So there are some problems:

1. Key distribution: How can two parties agree on a key?
   - rely on pre-existing secure communication...
   - meet in person.
   - use a trusted courier.
2. Key management: a group of t parties thus requires $t(t-1)/2$ keys when each pair wishes to communicate securely.

These keys must be kept secure and regularly changed to avoid potential security breaches.

Private-Key Cryptography

Symmetric Cryptography, is so named as the keys used to encrypt and decrypt a message are the same.

Anyone who knows the encryption key can decrypt a message, so it must be kept private hence its also called Private-Key Cryptography

So there are some problems:

1. Key distribution: How can two parties agree on a key?
   - rely on pre-existing secure communication...
   - meet in person.
   - use a trusted courier.
2. Key management: a group of t parties thus requires $t(t-1)/2$ keys when each pair wishes to communicate securely.

These keys must be kept secure and regularly changed to avoid potential security breaches.

Public-Key Cryptography

Idea: use keys k_E and k_D such that it is infeasible to calculate the one from the other.

The first practical public key protocol was introduced by Whitfield Diffie and Martin Hellman in 1976 in the form of a key exchange protocol.

The public key, k_E, can be published and anyone wishing to communicate with Alice just needs to find Alice's public key from a list and encrypt the message; only Alice will be able to decrypt the message using her corresponding private key, k_D.

This concept solves the problem of securely distributing keys. What's more, in a network of t people, only t keys are needed, a huge improvement on the situation using symmetric cryptography.

Public-Key Cryptography

Idea: use keys k_E and k_D such that it is infeasible to calculate the one from the other.

The first practical public key protocol was introduced by Whitfield Diffie and Martin Hellman in 1976 in the form of a key exchange protocol.

The public key, k_E, can be published and anyone wishing to communicate with Alice just needs to find Alice's public key from a list and encrypt the message; only Alice will be able to decrypt the message using her corresponding private key, k_D.

This concept solves the problem of securely distributing keys. What's more, in a network of t people, only t keys are needed, a huge improvement on the situation using symmetric cryptography.

Public-key cryptosystems can also be used to sign a message. Alice uses the private key to perform encryption (of the message) and then this signature accompanies the message. Then anyone can use the public key to decrypt the signature, and check that the message has not been tampered with, and that Alice is the only person who could have constructed the signature because she is the only one to know the private key.

Public-Key Cryptography Requirements and Assumptios

- The encryption function $e_{k_E} : \mathcal{P} \mapsto \mathcal{C}$ should
  - be easy to compute $\mathbf{y} = e_{k_E}(\mathbf{x})$
  - should be 1-1 and must have an inverse (to decrypt) $d_{k_D} : \mathcal{C} \mapsto \mathcal{P}$
- we assume Eve knows the function e_{k_E} and the encryption key k_E, and can evesdrop to learn $\mathbf{y}$.
  - It must be computationally infeasible to calculate k_D from k_E and $\mathbf{y}$
  - It must be computationally infeasible to calculate d_{k_D} without k_D

A function e_k is known as a "trap-door one-way function".

- "One-way" means it is difficult to invert.
- "Trap-door" means that the inverse can be found if one knows some additional information (the key k).

The RSA Cryptosystem

- Developed by Rivest, Shamir and Adelman, published in 1977.
  - Clifford Cocks, an English mathematician, had developed an equivalent system in 1973 at GCHQ, but it wasn't declassified until 1997
- **Key Generation**: Alice chooses two large primes (usually of approximately the same size) p and q and then

$$n_A = p \cdot q$$

  - Alice then chooses an encryption key e_A such that $\gcd(e_A, \phi(n_A)) = \gcd(e_A, (p-1)\cdot(q-1)) = 1$.
  - the public information is the pair $k_E = (e_A, n_A)$.
  - the coprime condition ensures that e_A has an inverse d_A modulo $\phi(n_A)$, and $k_D = d_A$.
  - to find d_A we need to factor n_A

Euler's totient function $\phi(n)$ counts the number of positive integers less than n that are relatively prime to n. So for instance, for a prime p

$$\phi(p) = p - 1$$

The totient function has useful properties, e.g., it is multiplicative, i.e., for coprime numbers m and n

$$\phi(mn) = \phi(m)\phi(n)$$

One way to write the function is in terms of the prime factors p_i of n, e.g.,

$$\begin{aligned} \phi(n) &= n \prod_{p_i} \left(1 - \frac{1}{p_i}\right) \\ e.g., \phi(36) = \phi(2^2 3^2) &= 36 \left(1 - \frac{1}{2}\right)\left(1 - \frac{1}{3}\right) \\ &= 12 \end{aligned}$$

RSA

The message and ciphertext spaces, $\mathcal{M}$ and $\mathcal{C}$, are the integers modulo n_A

Encryption

To encrypt a message x to send to Alice, Bob uses the public key $k_E = (e_A, n_A)$ to compute

$$y = e_{k_E}(x) = x^{e_A} \mod n_A$$

Decryption

When Alice receives y she computes

$$x = d_{k_D}(y) = y^{d_A} = x^{e_A \cdot d_A} = x^{l \cdot \phi(n)+1} = x \mod n_A$$

by the Euler-Fermat theorem.

RSA
The message and ciphertext spaces, $\mathcal{M}$ and $\mathcal{C}$, are the integers modulo n_A
Encryption
To encrypt a message x to send to Alice, Bob uses the public key $k_E = (e_A, n_A)$ to compute
$y = e_{k_E}(x) = x^{e_A} \mod n_A$
Decryption
When Alice receives y she computes
$x = d_{k_D}(y) = y^{d_A} = x^{e_A \cdot d_A} = x^{l \cdot \phi(n)+1} = x \mod n_A$
by the Euler-Fermat theorem.

The RSA decryption method uses the Euler-Fermat theorem:

Theorem (Euler-Fermat)

For positive, coprime integers x and m,

$$x^{\phi(m)} \equiv 1 \mod m.$$

where $\phi(m)$ is Euler's totient function.

Anyone can encrypt a message to send to Alice as (n_A, e_A) is public knowledge.

Only Alice knows d_A, so only Alice can decrypt the message.

The security of RSA is based on the belief that $e_k(x) = x^b \pmod n$ is a trapdoor one way function.

An opponent can find Alice's public key (e_A, n_A), but as they do not know p and q, they cannot easily find d_A such that

$$e_A d_A \equiv 1 \mod (p-1)(q-1)$$

this requires factorising n_A.

There most efficient (known) algorithm for factorising large numbers, the number field sieve, runs in subexponential time.

Anyone can encrypt a message to send to Alice as (n_A, e_A) is public knowledge.
Only Alice knows d_A, so only Alice can decrypt the message.
The security of RSA is based on the belief that $e_k(x) = x^b \pmod n$ is a trapdoor one way function.
An opponent can find Alice's public key (e_A, n_A), but as they do not know p and q, they cannot easily find d_A such that
$e_A d_A \equiv 1 \mod (p-1)(q-1)$
this requires factorising n_A.
There most efficient (known) algorithm for factorising large numbers, the number field sieve, runs in subexponential time.

There is still a key distribution problem, but it is slightly different. If Alice and Bob never meet, then how does Bob know that the public key listed for Alice, really belongs to Alice?

RSA Encryption

Suppose that Alice wishes to send a message to Bob. She first converts the message to numerical form.

A	B	C	D	E	F	G	H	I	J	K	L	M
00	01	02	03	04	05	06	07	08	09	10	11	12

N	O	P	Q	R	S	T	U	V	W	X	Y	Z	space
13	14	15	16	17	18	19	20	21	22	23	24	25	26

Note: we encipher spaces as well as letters, and we represent each letter by a two digit string.

RSA Encryption

Suppose that Alice wishes to send a message to Bob. She first converts the message to numerical form.

A B C D E F G H I J K L M
00 01 02 03 04 05 06 07 08 09 10 11 12

N O P Q R S T U V W X Y Z space
13 14 15 16 17 18 19 20 21 22 23 24 25 26

Note: we encipher spaces as well as letters, and we represent each letter by a two digit string.

We need each plaintext string to be in $\mathbb{Z}_n$ (and each ciphertext string will be in $\mathbb{Z}_n$). We use the following systematic method to divide the message into blocks that we can encipher:

- Suppose that n has d digits. Then the digits of the plaintext message are divided into blocks $x_1, x_2, \ldots, x_k$ such that each block has size $d-1$ digits (with 0 s added to the last block, if necessary, to ensure that it has $d-1$ digits).
- The ciphertext consists of k integers $y_1, y_2, \ldots, y_k$, each computed by:
$$y_j \equiv x_j^e \pmod{n}.$$
As we are working modulo n, each ciphertext y_j is in $\mathbb{Z}_n$, that is, it satisfies $0 \leq y_j < n$.
- The ciphertext is sent to Bob through *any* communication channel.

We need each plaintext string to be in $\mathbb{Z}_n$ (and each ciphertext string will be in $\mathbb{Z}_n$). We use the following systematic method to divide the message into blocks that we can encipher:

- Suppose that n has d digits. Then the digits of the plaintext message are divided into blocks $x_1, x_2, \ldots, x_k$ such that each block has size $d-1$ digits (with 0 s added to the last block, if necessary, to ensure that it has $d-1$ digits).
- The ciphertext consists of k integers $y_1, y_2, \ldots, y_k$, each computed by:
$$y_j \equiv x_j^e \pmod{n}.$$
As we are working modulo n, each ciphertext y_j is in $\mathbb{Z}_n$, that is, it satisfies $0 \leq y_j < n$.
- The ciphertext is sent to Bob through *any* communication channel.

The RSA cryptosystem is considered **computationally secure**: the best known algorithm for breaking it involves solving the integer factorisation problem.

To find $x_1, x_2, \ldots, x_k$ given $y_1, y_2, \ldots, y_k$, with $y_j \equiv x_j^e \pmod{n}$: compute d. Since e is public knowledge this requires computing $e^{-1} \bmod \phi(n)$. To find $\phi(n)$ we need to know the factorisation of n.

This is computationally infeasible to factor (within a given time frame), given current known techniques.

- There are no proofs that integer factorization is computationally difficult.
- There are no proofs that the RSA problem is equally difficult.

The best known method for breaking RSA is to factor a large number $\Rightarrow$ RSA problem is *at most* as hard as factoring (it may be easier using a yet unpublished method).

The RSA cryptosystem is considered **computationally secure**: the best known algorithm for breaking it involves solving the integer factorisation problem.

To find $x_1, x_2, \ldots, x_k$ given $y_1, y_2, \ldots, y_k$, with $y_j \equiv x_j^e \pmod{n}$: compute d. Since e is public knowledge this requires computing $e^{-1} \bmod \phi(n)$. To find $\phi(n)$ we need to know the factorisation of n.

This is computationally infeasible to factor (within a given time frame), given current known techniques.

- There are no proofs that integer factorization is computationally difficult.
- There are no proofs that the RSA problem is equally difficult.

The best known method for breaking RSA is to factor a large number $\Rightarrow$ RSA problem is *at most* as hard as factoring (it may be easier using a yet unpublished method).

Basic Framework

- Obviously, encrypting is a lot like coding
- Ciphers
  - have a key $\mathbf{k} \in \mathcal{K}$ from keyspace $\mathcal{K}$
  - convert plaintext message $\mathbf{x} \in \mathcal{M}$ into ciphertext $\mathbf{y} \in \mathcal{C}$
  - encryption: $\mathbf{y} = e_{\mathbf{k}}(\mathbf{x})$
  - decryption: $\mathbf{x} = d_{\mathbf{k}}(\mathbf{y})$
  - in some cases, a different (but related) key is used for encryption and decryption (e.g., public key encryption)
- Different attack models:
  - Assume attacker has the ciphertext and plaintext, and just needs key
  - Assume attacker only has ciphertext

Basic Framework

- Obviously, encrypting is a lot like coding
- Ciphers
  - have a key $\mathbf{k} \in \mathcal{K}$ from keyspace $\mathcal{K}$
  - convert plaintext message $\mathbf{x} \in \mathcal{M}$ into ciphertext $\mathbf{y} \in \mathcal{C}$
  - encryption: $\mathbf{y} = e_{\mathbf{k}}(\mathbf{x})$
  - decryption: $\mathbf{x} = d_{\mathbf{k}}(\mathbf{y})$
  - in some cases, a different (but related) key is used for encryption and decryption (e.g., public key encryption)
- Different attack models:
  - Assume attacker has the ciphertext and plaintext, and just needs key
  - Assume attacker only has ciphertext

Section 2

Modern Notion of Security

Section 2
Modern Notion of Security

Computational Security

A cryptosystem is computationally secure if the best algorithm for breaking it involves at least N operations (for some specified very large number N).

No known practical cryptosystem can be proved to be secure under this definition.

In practice, we say a cryptosystem is computationally secure if the best *known* algorithm to break it requires an unreasonably large amount of computer time.

Often breaking a cryptosystem requires solving a one-way mathematical problem: easy to compute in one direction, computationally infeasible to reverse.

Computational Security

A cryptosystem is computationally secure if the best algorithm for breaking it involves at least N operations (for some specified very large number N).

No known practical cryptosystem can be proved to be secure under this definition.

In practice, we say a cryptosystem is computationally secure if the best *known* algorithm to break it requires an unreasonably large amount of computer time.

Often breaking a cryptosystem requires solving a one-way mathematical problem: easy to compute in one direction, computationally infeasible to reverse.

Example: RSA

The RSA cryptosystem is considered computationally secure because the best known algorithm for breaking it involves solving the integer factorisation problem: It is easy to multiply numbers together but given a large composite number it is computationally infeasible to factor it (within a given time frame), given current known techniques.

- There are no proofs that integer factorization is computationally difficult.
- There are no proofs that the RSA problem is difficult.

The best known method for breaking RSA is to factor a large number $\Rightarrow$ RSA problem is *at least* as easy as factoring (it may be easier using a yet unpublished method).

Example: RSA

The RSA cryptosystem is considered computationally secure because the best known algorithm for breaking it involves solving the integer factorisation problem: It is easy to multiply numbers together but given a large composite number it is computationally infeasible to factor it (within a given time frame), given current known techniques.

- There are no proofs that integer factorization is computationally difficult.
- There are no proofs that the RSA problem is difficult.

The best known method for breaking RSA is to factor a large number $\Rightarrow$ RSA problem is *at least* as easy as factoring (it may be easier using a yet unpublished method)

Other Forms of Security

- computational security says that no known algorithm can break the security (with practical resources)
  - this is typical today
- unconditional security says that the encryption security doesn't depend on unproven assumptions
  - e.g., our belief that integer factorisations is hard
- perfect or information theoretic security says it cannot be broken, even with infinite computational resources.

We develop now the theory of cryptosystems that are information theoretically secure against ciphertext only attack (that is, we assume that an opponent knows the cryptosystem used and has access to some ciphertext).

Other Forms of Security

- computational security says that no known algorithm can break the security (with practical resources)
  - this is typical today
- unconditional security says that the encryption security doesn't depend on unproven assumptions
  - e.g., our belief that integer factorisations is hard
- perfect or information theoretic security says it cannot be broken, even with infinite computational resources.

We develop now the theory of cryptosystems that are information theoretically secure against ciphertext only attack (that is, we assume that an opponent knows the cryptosystem used and has access to some ciphertext).

Sometimes a subtle distinction is made between information theoretic and perfect security in that the former may allow some leakage of information, but still maintains its security, but this subtelty depends on the type of secret sharing we are doing, and the attack.

RSA is only computationally secure. We might one-day prove it is unconditionally secure, but it will never be informational theoretically secure, because with infinite computing power we can always solve the factorisation problem.

Assumptions on the Cryptosystem

To study unconditional security of a cryptosystem, we make the following assumptions about our cryptosystem and its operation.

- A1 Each key is used for at most one encryption.
- A2 The probability distribution on the message space $\mathcal{M}$ is $p_{\mathcal{M}}$
- A3 The probability distribution on the keyspace $\mathcal{K}$ is $p_{\mathcal{K}}$.
- A4 The key and the plaintext are chosen independently.

Assumptions on the Cryptosystem
To study unconditional security of a cryptosystem, we make the following assumptions about our cryptosystem and its operation.
A1 Each key is used for at most one encryption.
A2 The probability distribution on the message space $\mathcal{M}$ is $p_{\mathcal{M}}$
A3 The probability distribution on the keyspace $\mathcal{K}$ is $p_{\mathcal{K}}$.
A4 The key and the plaintext are chosen independently.

So $p_{\mathcal{M}}(\mathbf{x})$ is the probability that $\mathbf{x}$ is the message.

Alice and Bob choose the key according to this distribution: the probability that the key k is chosen is $p_{\mathcal{K}}(k)$.

The probability distributions on $\mathcal{M}$ and $\mathcal{K}$ induce a probability distribution on the ciphertext $\mathcal{C}$:

For $k \in \mathcal{K}$, define

$$\mathcal{C}(k) = \{\, e_k(\mathbf{x}) \mid \mathbf{x} \in \mathcal{M} \,\}.$$

So $\mathcal{C}(k) \subseteq \mathcal{C}$ is the set of all possible ciphertexts that can be obtained by using the key k.

Then for all $\mathbf{y} \in \mathcal{C}$,

$$p_{\mathcal{C}}(\mathbf{y}) = \sum_{\{k \mid \mathbf{y} \in \mathcal{C}(k)\}} p_{\mathcal{K}}(k) p_{\mathcal{M}}(d_k(\mathbf{y})).$$

The probability distributions on $\mathcal{M}$ and $\mathcal{K}$ induce a probability distribution on the ciphertext $\mathcal{C}$:
For $k \in \mathcal{K}$, define
$\mathcal{C}(k) = \{\, e_k(\mathbf{x}) \mid \mathbf{x} \in \mathcal{M} \,\}$.
So $\mathcal{C}(k) \subseteq \mathcal{C}$ is the set of all possible ciphertexts that can be obtained by using the key k.
Then for all $\mathbf{y} \in \mathcal{C}$,
$p_{\mathcal{C}}(\mathbf{y}) = \sum_{\{k \mid \mathbf{y} \in \mathcal{C}(k)\}} p_{\mathcal{K}}(k) p_{\mathcal{M}}(d_k(\mathbf{y}))$.

Now, for each $\mathbf{y} \in \mathcal{C}$, $\mathbf{x} \in \mathcal{M}$, we can calculate $p_{\mathcal{C}}(\mathbf{y}|\mathbf{x})$, the probability that $\mathbf{y}$ is the ciphertext given that $\mathbf{x}$ was the plaintext:

$$p_{\mathcal{C}}(\mathbf{y}|\mathbf{x}) = \sum_{\{k|\mathbf{x}=d_k(\mathbf{y})\}} p_{\mathcal{K}}(k).$$

So by Bayes' Theorem, we can also find $p_{\mathcal{M}}(\mathbf{x}|\mathbf{y})$:

$$p_{\mathcal{M}}(\mathbf{x}|\mathbf{y}) = \frac{p_{\mathcal{C}}(\mathbf{y}|\mathbf{x})p_{\mathcal{M}}(\mathbf{x})}{p_{\mathcal{C}}(\mathbf{y})}.$$

Now, for each $\mathbf{y} \in \mathcal{C}$, $\mathbf{x} \in \mathcal{M}$, we can calculate $p_{\mathcal{C}}(\mathbf{y}|\mathbf{x})$, the probability that $\mathbf{y}$ is the ciphertext given that $\mathbf{x}$ was the plaintext:
$p_{\mathcal{C}}(\mathbf{y}|\mathbf{x}) = \sum_{\{k|\mathbf{x}=d_k(\mathbf{y})\}} p_{\mathcal{K}}(k)$.
So by Bayes' Theorem, we can also find $p_{\mathcal{M}}(\mathbf{x}|\mathbf{y})$:
$p_{\mathcal{M}}(\mathbf{x}|\mathbf{y}) = \frac{p_{\mathcal{C}}(\mathbf{y}|\mathbf{x})p_{\mathcal{M}}(\mathbf{x})}{p_{\mathcal{C}}(\mathbf{y})}$.

Definition (Perfect Secrecy)

A cryptosystem has perfect secrecy if for all $\mathbf{x} \in \mathcal{M}$, $\mathbf{y} \in \mathcal{C}$,

$$p_{\mathcal{M}}(\mathbf{x}|\mathbf{y}) = p_{\mathcal{M}}(\mathbf{x}).$$

That is, knowledge of the ciphertext $\mathbf{y}$ does not give any information above the plaintext $\mathbf{x}$ it came from.
Note: from Bayes' Theorem, a cryptosystem has perfect secrecy if $p_{\mathcal{C}}(\mathbf{y}|\mathbf{x}) = p_{\mathcal{C}}(\mathbf{y})$ for all $\mathbf{x} \in \mathcal{M}$, $\mathbf{y} \in \mathcal{C}$.

We make a fifth assumption about our cryptosystem, namely that each ciphertext occurs with non-zero probability (if this is not true, we can remove any ciphertext $\mathbf{y}$ with $p_{\mathcal{C}}(\mathbf{y}) = 0$ from $\mathcal{C}$).

A5 For each $\mathbf{y} \in \mathcal{C}$, $p_{\mathcal{C}}(\mathbf{y}) \neq 0$.

Definition (Perfect Secrecy)
A cryptosystem has perfect secrecy if for all $\mathbf{x} \in \mathcal{M}$, $\mathbf{y} \in \mathcal{C}$,
$p_{\mathcal{M}}(\mathbf{x}|\mathbf{y}) = p_{\mathcal{M}}(\mathbf{x})$.
That is, knowledge of the ciphertext $\mathbf{y}$ does not give any information above the plaintext $\mathbf{x}$ it came from.
Note: from Bayes' Theorem, a cryptosystem has perfect secrecy if $p_{\mathcal{C}}(\mathbf{y}|\mathbf{x}) = p_{\mathcal{C}}(\mathbf{y})$ for all $\mathbf{x} \in \mathcal{M}$, $\mathbf{y} \in \mathcal{C}$.
We make a fifth assumption about our cryptosystem, namely that each ciphertext occurs with non-zero probability (if this is not true, we can remove any ciphertext $\mathbf{y}$ with $p_{\mathcal{C}}(\mathbf{y}) = 0$ from $\mathcal{C}$).
A5 For each $\mathbf{y} \in \mathcal{C}$, $p_{\mathcal{C}}(\mathbf{y}) \neq 0$.

Example

Suppose that the 26 keys in the Shift Cipher are used with equal probability 1/26. Then for any plaintext probability distribution, the Shift Cipher has perfect secrecy.

Note: Recall that by the assumption A1, each key is used for the encryption of only one letter, then another key is chosen.

Recall that for the shift cipher, $\mathcal{M} = \mathcal{C} = \mathcal{K} = \mathbb{Z}_{26}$, $e_k(x) = x + k \pmod{26}$ and $d_k(y) = y - k \pmod{26}$. Now

$$p_{\mathcal{C}}(y) = \sum_{k \in \mathbb{Z}_{26}} p_{\mathcal{K}}(k) p_{\mathcal{M}}(d_k(y)) = \frac{1}{26} \sum_{k \in \mathbb{Z}_{26}} p_{\mathcal{M}}(y - k) = \frac{1}{26}$$

Now for $x \in \mathcal{M}$, $y \in \mathcal{C}$, there exists a unique key $k \in \mathcal{K}$ such that $e_k(x) = y$, namely $k = y - x \pmod{26}$. Hence

$$p_{\mathcal{C}}(y|x) = p_{\mathcal{K}}(y - x) = p_{\mathcal{K}}(k) = \frac{1}{26}$$

Thus

$$p_{\mathcal{M}}(x|y) = \frac{p_{\mathcal{C}}(y|x) p_{\mathcal{M}}(x)}{p_{\mathcal{C}}(y)} = \frac{\frac{1}{26} p_{\mathcal{M}}(x)}{\frac{1}{26}} = p_{\mathcal{M}}(x)$$

Hence we have perfect secrecy. Thus the shift cipher is unbreakable provided a new (random) key k is chosen to encrypt each plaintext letter.

Example 2

Suppose that $\mathcal{M} = \{A, B\}$, $\mathcal{C} = \{a, b, c, d\}$, $\mathcal{K} = \{k_1, k_2, k_3\}$ and that

$$p_{\mathcal{M}}(A) = \frac{1}{4}, \quad p_{\mathcal{M}}(B) = \frac{3}{4}$$

$$p_{\mathcal{K}}(k_1) = \frac{1}{2}, \quad p_{\mathcal{K}}(k_2) = \frac{1}{4}, \quad p_{\mathcal{K}}(k_3) = \frac{1}{4}.$$

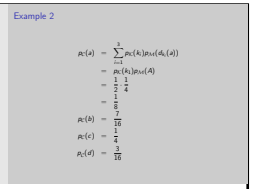
Further, suppose that the encryption functions e_{k_i} are given by the table

	A	B
e_{k_1}	a	b
e_{k_2}	b	c
e_{k_3}	c	d

We need to calculate $p_{\mathcal{C}}(y)$ and $p_{\mathcal{M}}(x|y)$ for each $x \in \mathcal{M}$ and $y \in \mathcal{C}$.

Example 2

$$
\begin{aligned}
p_{\mathcal{C}}(a) &= \sum_{i=1}^{3} p_{\mathcal{K}}(k_i) p_{\mathcal{M}}(d_{k_i}(a)) \\
&= p_{\mathcal{K}}(k_1) p_{\mathcal{M}}(A) \\
&= \frac{1}{2} \cdot \frac{1}{4} \\
&= \frac{1}{8} \\
p_{\mathcal{C}}(b) &= \frac{7}{16} \\
p_{\mathcal{C}}(c) &= \frac{1}{4} \\
p_{\mathcal{C}}(d) &= \frac{3}{16}
\end{aligned}
$$



$$
\begin{aligned}
p_{\mathcal{C}}(b) &= \sum_{\{k | k \in \{k_1, k_2\}\}} p_{\mathcal{K}}(k) p_{\mathcal{M}}(d_k(b)) \\
&= p_{\mathcal{K}}(k_1) p_{\mathcal{M}}(B) + p_{\mathcal{K}}(k_2) p_{\mathcal{M}}(A) \\
&= \frac{1}{2} \cdot \frac{3}{4} + \frac{1}{4} \cdot \frac{1}{4} = \frac{7}{16} \\
p_{\mathcal{C}}(c) &= p_{\mathcal{K}}(k_2) p_{\mathcal{M}}(B) + p_{\mathcal{K}}(k_3) p_{\mathcal{M}}(A) \\
&= \frac{1}{4} \cdot \frac{3}{4} + \frac{1}{4} \cdot \frac{1}{4} = \frac{1}{4} \\
p_{\mathcal{C}}(d) &= p_{\mathcal{K}}(k_3) p_{\mathcal{M}}(B) \\
&= \frac{1}{4} \frac{3}{4} = \frac{3}{16}
\end{aligned}
$$

Example 2

Now

$$p_{\mathcal{C}}(a|A) = P_{\mathcal{K}}(k|d_k(a) = A) = p(k_1) = \frac{1}{2},$$

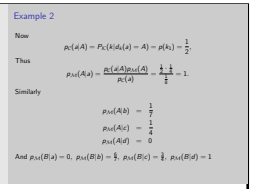
Thus

$$p_{\mathcal{M}}(A|a) = \frac{p_{\mathcal{C}}(a|A) p_{\mathcal{M}}(A)}{p_{\mathcal{C}}(a)} = \frac{\frac{1}{2} \cdot \frac{1}{4}}{\frac{1}{8}} = 1.$$

Similarly

$$
\begin{aligned}
p_{\mathcal{M}}(A|b) &= \frac{1}{7} \\
p_{\mathcal{M}}(A|c) &= \frac{1}{4} \\
p_{\mathcal{M}}(A|d) &= 0
\end{aligned}
$$

And $p_{\mathcal{M}}(B|a) = 0,\ p_{\mathcal{M}}(B|b) = \frac{6}{7},\ p_{\mathcal{M}}(B|c) = \frac{3}{4},\ p_{\mathcal{M}}(B|d) = 1$



Example 2

Note now that

$$\begin{aligned} p_{\mathcal{M}}(A|c) &= p_{\mathcal{M}}(A) \\ p_{\mathcal{M}}(B|c) &= p_{\mathcal{M}}(B) \end{aligned}$$

so knowing that the ciphertext is c doesn't give any information about which plaintext was used.

However, this is not true for the other ciphertext values. So the cryptosystem does not have perfect secrecy.

Example 2

Note now that

$$\begin{aligned} p_{\mathcal{M}}(A|c) &= p_{\mathcal{M}}(A) \\ p_{\mathcal{M}}(B|c) &= p_{\mathcal{M}}(B) \end{aligned}$$

so knowing that the ciphertext is c doesn't give any information about which plaintext was used.

However, this is not true for the other ciphertext values. So the cryptosystem does not have perfect secrecy.

Perfect Security Lemma

Lemma (Perfect Security)

For perfect secrecy we must have $|\mathcal{K}| \geq |\mathcal{C}| \geq |\mathcal{M}|$.

That is, the size of the keyspace must be at least as large as the size of the set of all possible ciphertexts (which must in turn be at least as large as the space of possible messages).

Perfect Security Lemma

Lemma (Perfect Security)

For perfect secrecy we must have $|\mathcal{K}| \geq |\mathcal{C}| \geq |\mathcal{M}|$.

That is, the size of the keyspace must be at least as large as the size of the set of all possible ciphertexts (which must in turn be at least as large as the space of possible messages).

Perfect Security Lemma

Proof.

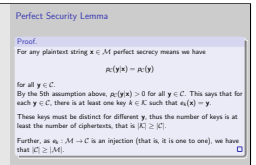
For any plaintext string $\mathbf{x} \in \mathcal{M}$ perfect secrecy means we have

$$p_{\mathcal{C}}(\mathbf{y}|\mathbf{x}) = p_{\mathcal{C}}(\mathbf{y})$$

for all $\mathbf{y} \in \mathcal{C}$.
By the 5th assumption above, $p_{\mathcal{C}}(\mathbf{y}|\mathbf{x}) > 0$ for all $\mathbf{y} \in \mathcal{C}$. This says that for each $\mathbf{y} \in \mathcal{C}$, there is at least one key $k \in \mathcal{K}$ such that $e_k(\mathbf{x}) = \mathbf{y}$.

These keys must be distinct for different $\mathbf{y}$, thus the number of keys is at least the number of ciphertexts, that is $|\mathcal{K}| \geq |\mathcal{C}|$.

Further, as $e_k : \mathcal{M} \to \mathcal{C}$ is an injection (that is, it is one to one), we have that $|\mathcal{C}| \geq |\mathcal{M}|$. □



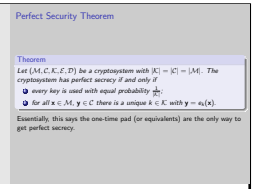
Perfect Security Theorem

Theorem

Let $(\mathcal{M}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ be a cryptosystem with $|\mathcal{K}| = |\mathcal{C}| = |\mathcal{M}|$. The cryptosystem has perfect secrecy if and only if

1. *every key is used with equal probability $\frac{1}{|\mathcal{K}|}$;*
2. *for all $\mathbf{x} \in \mathcal{M}$, $\mathbf{y} \in \mathcal{C}$ there is a unique $k \in \mathcal{K}$ with $\mathbf{y} = e_k(\mathbf{x})$.*

Essentially, this says the one-time pad (or equivalents) are the only way to get perfect secrecy.



Perfect Security Theorem

Proof.

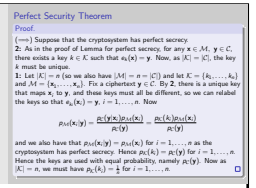
($\Longrightarrow$) Suppose that the cryptosystem has perfect secrecy.

2: As in the proof of Lemma for perfect secrecy, for any $\mathbf{x} \in \mathcal{M}$, $\mathbf{y} \in \mathcal{C}$, there exists a key $k \in \mathcal{K}$ such that $e_k(\mathbf{x}) = \mathbf{y}$. Now, as $|\mathcal{K}| = |\mathcal{C}|$, the key k must be unique.

1: Let $|\mathcal{K}| = n$ (so we also have $|\mathcal{M}| = n = |\mathcal{C}|$) and let $\mathcal{K} = \{k_1, \ldots, k_n\}$ and $\mathcal{M} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. Fix a ciphertext $\mathbf{y} \in \mathcal{C}$. By **2**, there is a unique key that maps $\mathbf{x}_i$ to $\mathbf{y}$, and these keys must all be different, so we can relabel the keys so that $e_{k_i}(\mathbf{x}_i) = \mathbf{y}$, $i = 1, \ldots, n$. Now

$$p_{\mathcal{M}}(\mathbf{x}_i|\mathbf{y}) = \frac{p_{\mathcal{C}}(\mathbf{y}|\mathbf{x}_i)p_{\mathcal{M}}(\mathbf{x}_i)}{p_{\mathcal{C}}(\mathbf{y})} = \frac{p_{\mathcal{K}}(k_i)p_{\mathcal{M}}(\mathbf{x}_i)}{p_{\mathcal{C}}(\mathbf{y})}$$

and we also have that $p_{\mathcal{M}}(\mathbf{x}_i|\mathbf{y}) = p_{\mathcal{M}}(\mathbf{x}_i)$ for $i = 1, \ldots, n$ as the cryptosystem has perfect secrecy. Hence $p_{\mathcal{K}}(k_i) = p_{\mathcal{C}}(\mathbf{y})$ for $i = 1, \ldots, n$. Hence the keys are used with equal probability, namely $p_{\mathcal{C}}(\mathbf{y})$. Now as $|\mathcal{K}| = n$, we must have $p_{\mathcal{K}}(k_i) = \frac{1}{n}$ for $i = 1, \ldots, n$. □

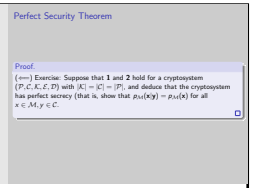


Perfect Security Theorem

Proof.

($\Longleftarrow$) Exercise: Suppose that **1** and **2** hold for a cryptosystem $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ with $|\mathcal{K}| = |\mathcal{C}| = |\mathcal{P}|$, and deduce that the cryptosystem has perfect secrecy (that is, show that $p_{\mathcal{M}}(\mathbf{x}|\mathbf{y}) = p_{\mathcal{M}}(\mathbf{x})$ for all $x \in \mathcal{M}, y \in \mathcal{C}$.

□



The One-Time Pad

One well known realization of perfect secrecy is the one-time pad. This was first described by Gilbert Vernam in 1917 for use in the encryption of telegraph messages. It was proved unbreakable by Shannon over 30 years later.

Let $n \geq 1$. We put

$$\mathcal{M} = \mathcal{C} = \mathcal{K} = (\mathbb{Z}_2)^n = \{ (a_1, \ldots, a_n) \mid a_i \in \mathbb{Z}_2 \}.$$

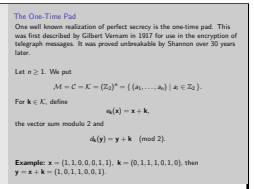
For $\mathbf{k} \in \mathcal{K}$, define

$$e_{\mathbf{k}}(\mathbf{x}) = \mathbf{x} + \mathbf{k},$$

the vector sum modulo 2 and

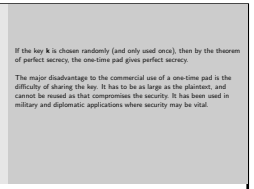
$$d_{\mathbf{k}}(\mathbf{y}) = \mathbf{y} + \mathbf{k} \pmod 2.$$

Example: $\mathbf{x} = (1,1,0,0,0,1,1)$, $\mathbf{k} = (0,1,1,1,0,1,0)$, then $\mathbf{y} = \mathbf{x} + \mathbf{k} = (1,0,1,1,0,0,1)$.



If the key **k** is chosen randomly (and only used once), then by the theorem of perfect secrecy, the one-time pad gives perfect secrecy.

The major disadvantage to the commercial use of a one-time pad is the difficulty of sharing the key. It has to be as large as the plaintext, and cannot be reused as that compromises the security. It has been used in military and diplomatic applications where security may be vital.



We need to keep one-time pads distinct from book ciphers: that is, ciphers that use a large-key in the form of some part of a (standard) book as the cipher.

- e.g., replace words/letters in the plaintext with locations words/letters from in the book
- e.g., (2nd Beale cipher)

These may (effectively) use each key only once, but the distribution of keys isn't uniform.

Information-Theoretic interpretation of Information Theoretic Security

- perfect secrecy if for all $\mathbf{x} \in \mathcal{M}$, $\mathbf{y} \in \mathcal{C}$, we have $p_{\mathcal{M}}(\mathbf{x}|\mathbf{y}) = p_{\mathcal{M}}(\mathbf{x})$.
- theorem: if $|\mathcal{K}| = |\mathcal{C}| = |\mathcal{M}|$ the cryptosystem has perfect secrecy if and only if
  1. every key is used with equal probability $\frac{1}{|\mathcal{K}|}$;
  2. for all $\mathbf{x} \in \mathcal{M}$, $\mathbf{y} \in \mathcal{C}$ there is a unique $k \in \mathcal{K}$ with $\mathbf{y} = e_k(\mathbf{x})$.
- **A1** Each key is used for at most one encryption.
- **A4** The key and the plaintext are chosen independently

Together these various properties and assumptions mean that

- the keys are IID uniform
- unique key to map $\mathbf{y} = e_k(\mathbf{x})$ means that the cipher text must be IID uniform as well

So the output of perfect encryption will be IID uniform.

Makes sense, as otherwise there would be some pattern or correlation in the data, and hence some lever to start decoding.

In general, even when you don't have perfect security, you would like to get as close to this property as possible.

Obviously, this means that compressing an encrypted signal is pointless – so compression must preceed encryption. However, that isn't guaranteed by the OSI layered structure. More importantly, you would really like to encypt a signal as early as possible, so there is as little trace of it on any part of a system (many "hacks" work by pulling bits of data out of the signal parth before encryption).

Other Examples

- Perfect, or information-theoretic security can't be broken even if the hypothetical adversay has infinite computing power
  - not vulnerable to future developments in computing or mathematics
- But perfect security is impractical for many problems
  - one-time pads are awkward at best
- There are other algorithms, in other "secret sharing" problems which have perfect security, that are more practical
  - e.g., Shamir's secret sharing
  - private information retrieval
  - quantum cryptography (?)

Further reading I

Claude Shannon, *Communication theory of secrecy systems*, Bell System Technical Journal **28** (1949), no. 4, 656–715, netlab.cs.ucla.edu/wiki/files/shannon1949.pdf.