Outline

Information, channel security, noninterference
Encryption and decryption
Cryptanalysis and notions of secrecy
Cyphers and modes of operation
Key establishment
What did we learn?

Recall from Lecture 1

Information security
- **secrecy**: "bad information flows don’t happen"
- **authenticity**: "good information flows do happen"

In network computation
- all information flow constraints are security properties

We could also say

Information security
- **confidentiality**: "bad information flows don’t…"
- **integrity**: "good information flows do…"

Although not synonymous
- secrecy, confidentiality and privacy
- authenticity and integrity are used interchangeably

Security speak

(overheard at a security conference)

**Speaker**: Isn’t it terrifying that on the Internet we have no privacy?

**Charlie**: You mean confidentiality. Get your terms straight.

**Radia**: Why do security types insist on inventing their own language?

**Mike**: It’s a denial-of-service attack.

**Charlie**: You mean chosen cyphertext attack…
Bad information flows

- **secret information**: disclosure prevented
  - e.g., by cryptography
- **private information**: disclosure when authorized
  - information privately owned
- **confidential information**: disclosure restricted
  - penalized when detected

**Examples of bad information flows**

- **secret funds**: it is secret that they exist
  - secret ceremony, secret lover...
- **private funds**: access is restricted
  - private ceremony, private resort...
- **confidential report**: some details confidential
  - content can be disclosed, but not the source

**What is information?**

Before a coin flip, the outcome is unknown.

A coin flip yields exactly 1 bit of information.
What is information?

Before two coin flips, the outcome is even more unknown.

Two coin flips give exactly 2 bits of information.

What is information?

Rolling a fair 4-sided die gives the same amount of information like flipping 2 fair coins.

What is information?

Let's get formal (but don't take it too seriously yet).

Definition

A source is a finite or countable set \( X \) given with a probability distribution.

Let's get formal (but don't take it too seriously yet).

Definition

A source is a finite or countable set \( X \) given with a probability distribution.

A probability distribution over \( X \) is a just function \( \text{Prob}_X : X \rightarrow [0,1] \) such that

\[
\sum_{x \in X} \text{Prob}(x) = 1
\]
What is information?

Definition

Information is the average length of the binary words needed to express the outcome of sampling a source $X$. It is denoted $H(X)$.

Examples

- $H(\text{coin}) = 1$
- $H(\text{2 coins}) = H(\text{4-sided die}) = 2$
- $H(\text{biased coins and dice}) < 2$
- If the outcome of an experiment $X$ is certain, then $H(X) = 0$.

Areas of information security

Just like

- information is a special kind of a resource,
- a message is a special kind of information sample.
Information gathering

Information can be acquired by
- observing accesses to resources
- receiving messages

Accordingly, we subdivide information security into:
- observation security, or *channels* security, and
- message security, or *cryptography*.

Observing confidential information

- Information flows through *channels*.

- Confidential information leaks through covert channels.

Trojan horse

is a covert channel installed through *social engineering*

Figure: A channel is concealed in a resource
State machines

**Definition**
A state machine is a map (pair of maps)
\[ A \times I \xrightarrow{\text{state machine}} A \times O \]
where \( A, I, O \) are finite sets, representing
- \( A \) — states
- \( I \) — input alphabet
- \( O \) — output alphabet

**Inputs and outputs**
The inputs and the outputs of state machines are lists from \( I \) and \( O \).

For any set \( X \), the set of lists
\[ X^* = \{ (x_1, x_2, \ldots, x_n) \in X^n | n \in \mathbb{N} \} \]
is generated from the empty list by prepending
\[ 1 \xrightarrow{e} X^* \]
\[ X \times X^* \xrightarrow{e} X^* \]

Running state machines

**Input-output maps**
At any state \( q \), the state machine \( Q \) induces a map
\[ q \xrightarrow{\text{Ev}^q} O^q \]
where
\[ \text{Ev}^q() = () \]
\[ \text{Ev}^q(x \& y) = \text{Ev}^q(x) \otimes \text{Ev}^q(y) \]
for \( x \in I \) and \( y \in I^* \)

State machines

**Definition**
A state machine is a map (pair of maps)
\[ A \times I \xrightarrow{\text{state machine}} A \times O \]
where \( A, I, O \) are finite sets, representing
- \( A \) — states
- \( I \) — input alphabet
- \( O \) — output alphabet

**Notation**
A state machine is denoted by the name of its state set \( Q \).

Running state machines

**Inputs and outputs**
The inputs and the outputs of state machines are lists from \( I \) and \( O \).

For any set \( X \), the set of lists in it
\[ X^* = \{ (x_1, x_2, \ldots, x_n) \in X^n | n \in \mathbb{N} \} \]
can be generated from the empty list by prepending
\[ 1 \xrightarrow{e} X^* \]
\[ X \times X^* \xrightarrow{e} X^* \]
\[ \langle x, (y_1, y_2, \ldots, y_n) \rangle \mapsto (x, y_1, y_2, \ldots, y_n) \]

Multi level machines

**Definition**
A multi level machine is a map
\[ A \times I \xrightarrow{\text{state machine}} A \times O \]
where \( A, I, O \) are finite sets, representing
- \( A \) — states
- \( I \) — input alphabet
- \( O \) — output alphabet
Hi-Lo machines

Definition
A Hi-Lo machine is a map

\[ Q \times I \xrightarrow{(Q \times \Ev)} Q \times O \]

where \( Q, I, O \) are finite sets, representing

- \( Q \) — states
- \( I = I_H + I_L \) — disjoint union input alphabets
- \( O \) — output alphabet

Hi-Lo machines

Notation
The restriction (or purge) \((-)_L : I' \rightarrow I'_L\) is defined

\[ Q_L = () \]

\[ (x@ys)_L = \begin{cases} x@ys_L & \text{if } x \in I_L \\ ys_L & \text{otherwise} \end{cases} \]

The outputs of Lo’s actions are:

\[ \Ev_{Q_L}'() = () \]

\[ \Ev_{Q_L}'(x@ys) = \begin{cases} \Ev_{Q_L}'(x) \oplus \Ev_{Q_L}'(ys) & \text{if } x \in I_L \\ \Ev_{Q_L}'(ys) & \text{otherwise} \end{cases} \]

Covert channels and Trojans

Definition
We say that the Hi-Lo machine \( Q \) has a covert channel if it has a state \( q \) such that

- \( x_{q_L} = y_{q_L} \), but
- \( \Ev_Q(x) \neq \Ev_Q(y) \)

holds for some input lists \( x, y \in I' \).

Remark
A Hi-Lo-machine is just a multi level machine with just two levels \( L = \{ L < H \} \).

Covert channels and Trojans

Definition
We say that the Hi-Lo machine \( Q \) has a covert channel if it has a state \( q \) such that

- \( x_{q_L} = y_{q_L} \), but
- \( \Ev_Q(x) \neq \Ev_Q(y) \)

holds for some input lists \( x, y \in I' \).

The subject Hi in a Hi-Lo machine with a covert channel is often called a Trojan (horse).
Covert channels and Trojans

Homework
Specify a simple Hi-Lo machine with a covert channel.

Noninterference
(Goguen-Meseguer)

Definition
We say that the Hi-Lo machine $Q$ satisfies the noninterference requirement if it has no covert channels, i.e.

\[ x_L = y_L \implies Ev^L_q(x_L) = Ev^L_q(y_L) \]

holds for all states $q$ and all inputs $x, y \in \Gamma$.

Remark
The no-write-down condition
• prevents Hi from sending to Lo
• any publicly visible signals (messages).

Generalized noninterference
(McCullough, McLean)

Definition
We say that the Hi-Lo machine $Q$ satisfies the generalized noninterference requirement if

\[ \forall x, y, z \in \Gamma: x_L = y_L \land y_H = z_H \land Ev^L_q(x_L) = Ev^L_q(y_L) \]

holds for all states $q$.

Homework
Prove that generalized noninterference and noninterference are equivalent for deterministic machines

Remark
Generalized noninterference is also applicable to nondeterministic machines.
Outline

Information, channel security, noninterference

Encryption and decryption
- Cryptosystems
- Examples of simple crypto systems
- Coding vs encryption

Cryptanalysis and notions of secrecy

Cyphers and modes of operation

Key establishment

What did we learn?

Simple crypto system

Definition

...a simple crypto-system is a triple of algorithms:
- key generation \((K_E, K_D) : K \times K\),
- encryption \(E : K \times M \rightarrow C\), and
- decryption \(D : K \times C \rightarrow M\),

Using a cryptosystem

What where do the plaintexts come from?

Remarks

- The space \(M\) may be
  - monoalphabetic: it consists of symbols
    - \(M = \Sigma\)
  - polyalphabetic: it consists of blocks of symbols
    - \(M = \Sigma^n\)
What where do the plaintexts come from?

Remarks
- The space $M$ may be
  - monalphabetic: it consists of symbols
    - $M = \Sigma$
  - polyalphabetic: it consists of blocks of symbols
    - $M = \Sigma^\omega$
- A plaintext is a string from $M$.
- A well-formed message is a meaningful plaintext:
  a word, a sentence, a paragraph.
- Not every plaintext is a well-formed message.

What shall we study?

- Cryptography: science of crypto systems
  - Cryptology: designing crypto systems
    - to encrypt plaintexts as ciphertexts
    - so that only those with a key can decrypt them
  - Cryptanalysis: breaking crypto systems
    - to extract the plaintexts without a key
    - or even better, to extract the key

Example 1.1: Shift cypher
(monoalphabetic: Caesar $k = 3$, ROT13 $k = 13$...

- $M = \mathbb{Z}_{26} = \{0, 1, 2, 3, \ldots, 25\}$
- $\mathcal{N} = \mathbb{Z}_{26}$
- $K_E = K_D = k$
- $E(k, m) = m + k \mod 26$
- $D(k, c) = c - k \mod 26$
Example 1.1: Shift cypher

E.g., the key $k = 5$ gives

$\begin{array}{cccccccccc}
    m & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\
    k & 3 & 4 & 5 & 6 & 7 & 0 & 1 & 2 & 9 \\
    c & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 \\
\end{array}$

where

$C = N \oplus M$.

Example 1.2: Shift cypher

E.g., the block length $N = 6$ and the keyword $k = \text{monk}y$ give

$\begin{array}{cccccccccc}
    m & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
    k & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\
    c & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 \\
\end{array}$

where

$C = M \oplus k$.

Example 1.2: Shift cypher

Fact

A polyalphabetic shift cypher where

- a key $K \in \mathbb{Z}_N^*$ is used to encrypt
- more than one $m$, $c \in \mathbb{Z}_N$

is insecure.

We shall prove this.

Example 1.2: Shift cypher

Terminology

A polyalphabetic shift cypher where

- each key $K \in \mathbb{Z}_N^*$ is used to encrypt
- a single message $m \in \mathbb{Z}_N$

called a one-time-pad. It is

- perfectly secure, but it reduces
- the task to transfer an $N$-character message to
- the task to transfer an $N$-character key.

Example 1.2: Shift cypher

E.g., the block length $N = 6$ and the keyword $k = \text{monkey}$ give

$\begin{array}{cccccccccc}
    m & 8 & 19 & 8 & 18 & 21 & 4 & 17 & 24 & 2 \\
    k & 13 & 24 & 13 & 24 & 0 & 9 & 22 & 3 & 7 \\
    c & 19 & 55 & 24 & 24 & 30 & 16 & 24 & 15 & 1 \\
\end{array}$

where

$C = N \oplus K \oplus M_0$.

Example 1.2: Shift cypher

$M = C = \mathbb{Z}_N^*$

$K = \mathbb{Z}_N^*$

$K_0 = K_2 = \ldots = K_N$

$E(k, m) = m + k \mod 26$

$D(k, c) = c - k \mod 26$
Example 1.2: Shift cypher (polyalphabetic)

Terminology vs history
Polyalphabetic shift ciphers are often called Vigenère's cyphers. This is a sad confusion. Vigenère had nothing to do with polyalphabetic shift cyphers. He designed the first auto-keying cypher.

Example 1.3: Affine cypher (polyalphabetic)

- $M = C = \mathbb{Z}_N$.
- $\mathcal{K} = \mathbb{S}(\Sigma)$, the permutations of $\Sigma$.
- $K_E = K_D = \sigma$.
- $E(\sigma, m) = \sigma(m)$.
- $D(\sigma, c) = \sigma^{-1}(c)$.

Example 1.4: Substitution cypher (monoalphabetic)

- $M = C = \Sigma = \{a, b, c, \ldots, z\}$.
- $\mathcal{K} = \mathbb{S}(\Sigma)$, the permutations of $\Sigma$.
- $K_E = K_D = \sigma$.
- $E(\sigma, m) = \sigma(m)$.
- $D(\sigma, c) = \sigma^{-1}(c)$.

Example 1.5: Substitution cypher (polyalphabetic)

- $M = C = \Sigma^N$.
- $\mathcal{K} = \mathbb{S}(\Sigma^N)$, the permutations of $\Sigma$.
- $K_E = K_D = \sigma$.
- $E(\sigma, m) = \sigma(m_1, m_2, \ldots, m_N)$.
- $D(\sigma, c) = \left\{ \sigma^{-1}(c_1), \sigma^{-1}(c_2), \ldots, \sigma^{-1}(c_N) \right\}$.

where $N = \{1, 2, \ldots, n\}$.

Example 2: Transposition cypher

- $M = C = \mathbb{Z}_n$.
- $\mathcal{K} = \mathbb{S}(\mathbb{N})$, the permutations of the block positions.
- $K_E = K_D = \sigma$.
- $E(\sigma, m) = \left( m_{\sigma(1)}, m_{\sigma(2)}, \ldots, m_{\sigma(n)} \right)$.
- $D(\sigma, c) = \left( m_{\sigma^{-1}(1)}, m_{\sigma^{-1}(2)}, \ldots, m_{\sigma^{-1}(n)} \right)$.

Example 3: RSA

- $M = C = \mathbb{Z}_n$, where $n = pq$, $p, q$ prime.
- $\mathcal{K} = \mathbb{Z}_n^\times$, where $\phi(n) \neq \phi(k)$, gcd$(n, k) = 1$.
- $K_E = e$.
- $K_D = e^{-1} \mod \phi(n)$.
- $E(e, m) = m^e \mod n$.
- $D(d, c) = c^d \mod n$. 
Example 3: RSA

<table>
<thead>
<tr>
<th>Idea of public key cryptography</th>
</tr>
</thead>
<tbody>
<tr>
<td>- $K_E$ is publicly announced</td>
</tr>
<tr>
<td>- everyone can encrypt</td>
</tr>
<tr>
<td>- $K_D$ is kept secret</td>
</tr>
<tr>
<td>- only those who have it can decrypt</td>
</tr>
</tbody>
</table>

It is important that $K_D$ cannot be derived from $K_E$.

Example 3: RSA

<table>
<thead>
<tr>
<th>History of public key cryptography</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Whit Diffie and Marty Hellman proposed computational hardness as a new foundation for cryptography in 1976.</td>
</tr>
<tr>
<td>- Ron Rivest, Adi Shamir and Len Adleman (RSA) implemented that idea using exponentiation in 1978.</td>
</tr>
</tbody>
</table>

The RSA patent became a base of a very profitable company.

Example 3: RSA

<table>
<thead>
<tr>
<th>History of public key cryptography</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Whit Diffie and Marty Hellman proposed computational hardness as a new foundation for cryptography in 1976.</td>
</tr>
<tr>
<td>- Ron Rivest, Adi Shamir and Len Adleman (RSA) implemented that idea using exponentiation in 1978.</td>
</tr>
<tr>
<td>- The RSA patent became a base of a very profitable company. All involved became rich and famous.</td>
</tr>
</tbody>
</table>
History of public key cryptography

- In December 1997, the British Government Communications Headquarters (GCHQ) released five papers.
- James Ellis' paper "The possibility of non-secret encryption" proposed computational hardness as a foundation for cryptography. 1970
- Clifford Cocks' paper "A note on non-secret encryption" implemented that idea using exponentiation. 1973
- Clifford Cocks became the Chief Mathematician at GCHQ in 2007.
- Public key cryptography was critical in arm treaty control as of 1986, but was not deployed earlier.
- Take \( p = 11 \) and \( q = 17 \).
Example 3: RSA

- Take \( p = 11 \) and \( q = 17 \). Hence
  - \( n = pq = 187 \), and
  - \( \phi(n) = (11 - 1)(17 - 1) = 160 \)
- Take \( K_E = e = 3 \)
- Then \( K_D = d = 3^{-1} = 107 \mod 160 \)
- \( E(3, p) = J \) because
  - \( E(3, 15) = 15^3 \equiv 3375 \equiv 9 \mod 187 \)
Example 3: RSA

Homework

Prove that Euler's totient function

\[ \varphi : \mathbb{N} \rightarrow \mathbb{N} \]

\[ n \mapsto \# \{ k < n \mid \gcd(n, k) = 1 \} \]

has the following properties:

- \( \varphi(p^k) = (p-1)p^{k-1} \) if \( p \) is prime
- \( \varphi(mn) = \varphi(m) \cdot \varphi(n) \) if \( \gcd(m, n) = 1 \)

Derive a general formula to compute \( \varphi(n) \).

Example 3: RSA

\[ e \cdot d = 1 \mod \varphi(n) \implies (m^n)^d = m \mod n \]

Example 3: RSA

\[ \quad \text{is a crypto system because} \]

- **unique decryption** holds by

\[
\left. e \cdot d = 1 \mod \varphi(n) \right\} \implies (m^n)^d = m \mod n
\]

- **trapdoor encryption** holds since for every \( A \)

\[
\forall m \in \mathbb{Z}_n^* \quad m \mod n \implies \forall c \in \mathbb{Z}_n^* \quad c^d \mod n
\]

where \( ed = 1 \mod \varphi(n) \)

Refresher in arithmetic

**Definition**

Let \( \langle G, \cdot \rangle \) be a finite group and \( g \in G \). We define

\[
\text{ord}(G) = |G| \quad (\text{the number of elements})
\]

\[
\text{ord}(g) = |\langle g \rangle| = \min \left\{ |G| \mid g^k \neq 1 \right\}
\]

**Theorem (Lagrange)**

For every \( g \in G \) holds \( \text{ord}(g) \mid \text{ord}(G) \).

**Definition**

The multiplicative group of invertible elements of \( \mathbb{Z}_n \) is

\[
\mathbb{Z}_n^* = \{ x \in \mathbb{Z}_n \mid \exists y \cdot xy = 1 \mod n \}
\]

**Lemma**

\( k \in \mathbb{Z}_n^* \) is invertible if it is mutually prime with \( n \), i.e.

\[
k \in \mathbb{Z}_n^* \iff \gcd(n, k) = 1
\]

Hence \( \text{ord}(\mathbb{Z}_n^*) = \# \{ k < n \mid \gcd(n, k) = 1 \} = \varphi(n) \).
Corollary (Euler)
For every invertible \( k \in \mathbb{Z}_n^* \) holds
\[
k^{\varphi(n)} = 1 \mod n
\]

Proof.
By the Theorem, \( \text{ord}(k) \mid \text{ord}(\mathbb{Z}_n^*) \).
By the Lemma, \( \text{ord}(\mathbb{Z}_n^*) = \varphi(n) \).

RSA unique decryption

Conclusion
Hence the unique decryption property of RSA
\[
ed = 1 \mod \varphi(n) \iff \exists \ell. ed = 1 + l\varphi(n)
\implies m^d = m^{1+l\varphi(n)} = m \mod n
\]

RSA Assumption

RSA Problem
• input:
  • \( n = pq \in \mathbb{N} \) where \( p \) and \( q \) are prime
  • \( e \in \mathbb{Z}_n^* \), i.e. \( \gcd(e, n) = 1 \)
  • \( d \in \mathbb{Z}_n^* \), i.e. \( \gcd(d, n) = 1 \)
• output:
  • \( m = \sqrt[ed]{c} \mod n \), i.e. \( m^e = c \mod n \)

RSA Assumption
There is no feasible algorithm solving the RSA Problem.
Examples of coding

- Morse code
- telegraphic codes
- source characters
- code strings of dots and dashes
- source meaningful phrases
- code strings
- code strings

Coding vs encryption

Definition

A coding scheme is an injective function \( f : X \rightarrow G \)
where
- \( X \) is a source
- \( G \subset \Sigma^* \) is a language (or code).

Terminology

The elements \( \gamma \in G \subset \Sigma^* \) are called codewords.
Codewords are used as well-formed messages.

Remark

RSA problem can be solved by finding \( d = e^{-1} \mod \phi(n) \)
But computing \( \phi(n) \) requires factoring \( n \).

It is believed that factoring is not feasible:
if \( n \) has only large factors, they are hard to find.

Remark

RSA problem can be solved by finding \( d = e^{-1} \mod \phi(n) \)

Definition

A coding scheme is an injective function \( f : X \rightarrow G \)
where
- \( X \) is a source
- \( G \subset \Sigma^* \) is a language (or code).

Terminology

The elements \( \gamma \in G \subset \Sigma^* \) are called codewords.
Codewords are used as well-formed messages.

Remark

RSA problem can be solved by finding \( d = e^{-1} \mod \phi(n) \)

Definition

A coding scheme is an injective function \( f : X \rightarrow G \)
where
- \( X \) is a source
- \( G \subset \Sigma^* \) is a language (or code).

Terminology

The elements \( \gamma \in G \subset \Sigma^* \) are called codewords.
Codewords are used as well-formed messages.
Coding vs encryption

Terminology
The elements \( \gamma \in \mathcal{G} \subseteq \Sigma^* \) are called codewords. Codewords are used as well-formed messages.

Remark
We usually take \( M = \Sigma \). Any string of plaintexts \( m \in \Sigma^* \) can be a message. (E.g., meaningful words and meaningless strings.) Not every message is a codeword. Those that are are said to be well-formed.

Coding vs encryption

Terminology
The elements \( \gamma \in \mathcal{G} \subseteq \Sigma^* \) are called codewords. Codewords are used as well-formed messages.

Remark
We usually take \( M = \Sigma \). Any string of plaintexts \( m \in \Sigma^* \) can be a message. (E.g., meaningful words and meaningless strings.) Not every message is a codeword.

Coding vs encryption

Terminology
The elements \( \gamma \in \mathcal{G} \subseteq \Sigma^* \) are called codewords. Codewords are used as well-formed messages.

Upshot
The difference between

\[ \text{decryption } \mathcal{C} \xrightarrow{D} M \]
\[ \text{decoding } M^* \xleftarrow{} \mathcal{G} \]

will play an important role in cryptanalysis.

Cryptanalytic attacks

Symmetric key attacks
When \( K_E = K_D = K \), the attacks are

- cyphertext only (COA):
  \[ E(K, m_1), \ldots, E(K, m_\ell) \xrightarrow{\mathcal{C}} K \]
- known plaintext (KPA), chosen plaintext (CPA):
  \[ m_1, \ldots, m_\ell, E(K, m_1), \ldots, E(K, m_\ell) \xrightarrow{\mathcal{C}} K \]
- chosen cyphertext (CCA):
  \[ c_1, \ldots, c_\ell, D(K, c_1), \ldots, D(K, c_\ell) \xrightarrow{\mathcal{C}} K \]
Cryptanalytic attacks

Asymmetric key attacks
When $K_E$ is publicly known
- cyphertext only (COA):
  \[ K_E, E(K_E, m_1), \ldots, E(K_E, m_i) \rightarrow K_D \]
- known plaintext (KPA), chosen plaintext (CPA):
  \[ K_E, m_1, \ldots, m_i, E(K_E, m_1), \ldots, E(K_E, m_i) \rightarrow K_D \]
- chosen cyphertext (CCA):
  \[ K_E, c_1, \ldots, c_i, D(K_D, c_1), \ldots, D(K_D, c_i) \rightarrow K_D \]
- adaptive chosen cyphertext (CCA2): \( \ldots \) (later?)

Idea
Since there are just \#K = 26 possible keys, simply try one after the other.

COA on monoalphabetic shift cypher

- $M = C = \mathbb{Z}_{26}$
- $K' = \mathbb{Z}_{26}$
- $K_E = K_D = k$
- $E(k, m) = m + k \mod 26$
- $D(k, c) = c - k \mod 26$

COA on monoalphabetic shift cypher

- $M = C = \mathbb{Z}_{26}$
- $K' = \mathbb{Z}_{26}$
- $K_E = K_D = k$
- $E(k, m) = m + k \mod 26$
- $D(k, c) = c - k \mod 26$

<table>
<thead>
<tr>
<th>CY</th>
<th>N</th>
<th>Y</th>
<th>N</th>
<th>X</th>
<th>A</th>
<th>J</th>
<th>W</th>
<th>D</th>
<th>H</th>
<th>T</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>d</td>
<td>13</td>
<td>24</td>
<td>13</td>
<td>23</td>
<td>0</td>
<td>9</td>
<td>22</td>
<td>3</td>
<td>7</td>
<td>19</td>
<td>16</td>
</tr>
<tr>
<td>k</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>m</td>
<td>11</td>
<td>22</td>
<td>11</td>
<td>21</td>
<td>24</td>
<td>7</td>
<td>20</td>
<td>1</td>
<td>5</td>
<td>17</td>
<td>14</td>
</tr>
<tr>
<td>tx</td>
<td>i</td>
<td>w</td>
<td>i</td>
<td>v</td>
<td>y</td>
<td>h</td>
<td>u</td>
<td>b</td>
<td>f</td>
<td>r</td>
<td>o</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CY</th>
<th>N</th>
<th>Y</th>
<th>N</th>
<th>X</th>
<th>A</th>
<th>J</th>
<th>W</th>
<th>D</th>
<th>H</th>
<th>T</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>13</td>
<td>24</td>
<td>13</td>
<td>23</td>
<td>0</td>
<td>9</td>
<td>22</td>
<td>3</td>
<td>7</td>
<td>19</td>
<td>16</td>
</tr>
<tr>
<td>k</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>m</td>
<td>8</td>
<td>19</td>
<td>8</td>
<td>18</td>
<td>21</td>
<td>4</td>
<td>17</td>
<td>24</td>
<td>12</td>
<td>11</td>
<td>3</td>
</tr>
<tr>
<td>tx</td>
<td>i</td>
<td>t</td>
<td>i</td>
<td>s</td>
<td>v</td>
<td>e</td>
<td>r</td>
<td>y</td>
<td>c</td>
<td>o</td>
<td>l</td>
</tr>
</tbody>
</table>
Fact

Since \( |\mathcal{K}| = 26! \approx 4 \cdot 10^{26} \), enumerating the keys and searching for a well-formed plaintext will not help.

Idea

Align the letter frequencies of plaintext (e.g. English)...

... with the letter frequencies of the ciphertext

Summary

- the messages are drawn from a source \( \mathcal{X} \) and coded along \( f : \mathcal{X} \rightarrow \mathcal{Y} \subseteq \mathcal{M}^* \)
- the frequency distribution \( \Pr_X : \mathcal{X} \rightarrow [0, 1] \) induces the frequency distribution \( \Pr_M : \mathcal{M} \rightarrow [0, 1] \)
- \( \Pr_M(c) = \Pr_X(f^{-1}(c)) \)
- the frequency distribution \( \Pr_M : C \rightarrow [0, 1] \) can be extracted if there is enough ciphertext
**COA on substitution cipher**

The patterns

\[ M \rightarrow C \]

Prob \[ [0, 1] \]

**KPA on the one-time-pad**

- \[ M = C = \mathcal{K} = \mathbb{Z}_{26}^N \]
- \[ E(\mathcal{K}, \mathcal{M}) = \mathcal{M} + \mathcal{K} \]
- \[ D(\mathcal{K}, \mathcal{C}) = \mathcal{C} - \mathcal{K} \]

**Proposition**

If all keys are equally likely, then the one-time-pad is secure, in the sense that the ciphertext provides no information about the plaintext.

**Can we prove that there are no attacks?**

- Attack

Given \[ \mathcal{M} \text{ and } E(\mathcal{K}, \mathcal{M}) = \mathcal{M} + \mathcal{K} \] the cryptanalyst derives

\[ \mathcal{K} = E(\mathcal{K}, \mathcal{M}) - \mathcal{M} \]
Can we prove that there are no attacks?

We need tools for such proofs!

Guessing

Attack scenario: KPA, CPA
The cryptanalyst knows which crypto system is used. He wants to derive the key from the known or chosen plaintext, and its encryptions

\[ m_1, \ldots, m_n, E(K, m_1), \ldots, E(K, m_n) \vdash K \]

In some cases, he
\> may not know the plaintext, but
\> can recognize well-formed messages.

Guessing

Terminology
A random variable is a function \( X : \mathcal{X} \rightarrow \mathcal{V} \) where
\> \( \mathcal{X} \) is a source and
\> \( \mathcal{V} \) is a set, representing values.

Notation
We write

\[ \text{Prob}(X = \nu) = \frac{\text{Prob}(X \in \mathcal{X} \mid X) = \nu}{\sum_{X \mid X = \nu} \text{Prob}(X)} \]

Guessing

Guessing process
Given a probability distribution over the key space \( \mathcal{K} \), a guessing attack is a random variable \( G : \mathcal{K} \rightarrow \mathcal{N} \), where

\[ G(k_1, k_2, \ldots, k_n) = i \]

means that \( k_i = K_0 \).
Guessing

Guessing process
Given a probability distribution over the key space $\mathcal{K}$, a
guessing attack is a random variable $G : \mathcal{K} \rightarrow \mathbb{I}$, where
$G(k_1, k_2, \ldots, k_n) = i$
means that $k_i = K_i$.

Remark
The intuition is that we are given some ciphertext $c$, and
we test whether $D(k, c)$ is a well-formed message for one
$k_i$ after the other.

Solution
Since there are $\ell = \#\mathcal{K}$ equally likely keys,
- the probability that the right key is drawn at once is $\Pr(G = 1) = p_1$;
- the probability that the right key is not drawn at once
  is $q_1 = 1 - p_1$. In this case, we draw again, from $\ell - 1$ untested keys.

Exercise
Suppose that there are $\ell = \#\mathcal{K}$ keys, and that they are all
equally likely. What is the probability that
- $G = 1$, i.e. the key is guessed at once,
- $G = n$, i.e. the key is guessed after exactly $n$ tries.
- $G \leq n$, i.e. the key is guessed in at most $n$ tries.

Solution
Since there are $\ell = \#\mathcal{K}$ equally likely keys,
- the probability that the right key is drawn at once is $\Pr(G = 1) = p_1$;
- the probability that the right key is not drawn at once
  is $q_1 = 1 - p_1$. In this case, we draw again, from $\ell - 1$ untested keys. This time,
  - the probability that the right key is drawn immediately is now $p_2 = \frac{1}{\ell - 1}$, and thus
    $\Pr(G = 2) = \frac{1}{\ell - 1}$. 

Guessing

Solution

- Since there are \( n = \#X \) equally likely keys,
  - the probability that the right key is drawn at once is
    \[ \operatorname{Prob}(G = 1) = p_1 = \frac{1}{n} , \]
  - the probability that the right key is not drawn at once is
    \[ q_1 = \operatorname{Prob}(G \neq 1) = 1 - p_1 = 1 - \frac{1}{n} . \]
  - In this case, we draw again, from \( n-1 \) untested keys. This time,
    - the probability that the right key is drawn immediately is now \( p_2 = \frac{1}{n-1} \), and thus
      \[ \operatorname{Prob}(G = 2) = q_1 \cdot p_2 = \frac{1}{n(n-1)} ; \]
    - whereas the probability that the right key is still not drawn is \( q_2 = \cdots \).

Guessing

In general, with \( p_i = \frac{1}{n-i} \) and \( q_i = \frac{n-i}{n} \), the probability that a particular key is drawn in the \( n \)-th draw is

\[
\operatorname{Prob}(G = n) = q_1 \cdot q_2 \cdot \cdots \cdot q_{n-1} \cdot p_n = \frac{1}{n(n-1) \cdots (n-n+1)} \cdot \frac{1}{n} = \frac{1}{n}.
\]

The probability that a particular key is drawn in at most \( n \) tries is

\[
\operatorname{Prob}(G \leq n) = \sum_{i=1}^{n} \operatorname{Prob}(G = i) = \frac{n}{n} = 1.
\]

Elements of probability

Notation

Given a source \( X \) and events \( \alpha, \beta, \gamma \ldots \subseteq X \), we write

\[
[\alpha] = \sum_{x \in X} \operatorname{Prob}(x) \quad [\alpha \cap \beta] = \frac{[\alpha \cap \beta]}{[\alpha]}
\]

Remark

Traditionally, our \([\alpha \implies \beta] \) is written \( \operatorname{Prob}(\beta \mid \alpha) \), and called conditional probability.

While the traditional notations need to be respected, cryptography puts conditional probability to heavy use, and abuse.
Elements of probability

Remark
Traditionally, our \( \alpha \vdash \beta \) is written \( \text{Prob} (\beta \mid \alpha) \), and called conditional probability. While the traditional notations need to be respected, cryptography puts conditional probability to heavy use, and abuse. \( \alpha \vdash \beta \) tells how likely it is to guess \( \beta \) from \( \alpha \).

Elements of probability

Homework
\[
\alpha \vdash \neg \beta = 1 - [\alpha \vdash \beta]
\]
\[
[\beta] = [\alpha] \cdot [\alpha \vdash \beta] + [\neg \alpha] \cdot [\neg \alpha \vdash \beta]
\]
\[
[\alpha \vdash \beta \cup \gamma] = [\alpha \vdash \beta] + [\alpha \vdash \gamma] - [\alpha \vdash \beta \cap \gamma]
\]
Moreover
\[
[\alpha \cap \beta] = [\alpha] \cdot [\beta] \iff [\alpha \vdash \beta] = [\beta]
\]
\[
[\beta \vdash \alpha] \iff [\alpha \vdash \beta] = [\alpha]
\]

Elements of probability

Bayes theorem
\[
[\beta \vdash \alpha] = \frac{[\alpha][\alpha \vdash \beta]}{[\alpha][\alpha \vdash \beta] + [\neg \alpha][\neg \alpha \vdash \beta]}
\]

Elements of probability

Proposition
\[
[\beta \vdash \alpha] = [\gamma \vdash \alpha]
\]
\[
[\alpha \vdash \beta] : [\beta \vdash \gamma] = [\alpha \vdash \gamma] : [\gamma \vdash \beta]
\]

Elements of probability

Since
\[
[\alpha \vdash \beta \cap \gamma] = [\alpha \vdash \beta] \cdot [\alpha \cap \beta \vdash \gamma]
\]
it follows that
\[
[\alpha \vdash \beta] : [\alpha \cap \beta \vdash \gamma] \leq [\alpha \vdash \gamma]
\]
with the equality when \( [\alpha \cap \gamma \vdash \beta] = 1 \), so that \( [\alpha \vdash \gamma] = [\alpha \vdash \beta \cap \gamma] \).
### Problem with simple crypto systems

#### Leaking partial information

The trapdoor encryption condition

\[ \forall m \in \mathbb{Z} \forall c \in \mathbb{Z} : A(E(K, m)) = m \implies \forall c : D(K, c) = c \]

only talks about total decryptions. A simple crypto system can leak partial information.

### Example: Reusing one-time-pad

#### Proposition

If the same one-time-pad key is used to encrypt more than one block, then a CPA attacker can extract partial information.

#### Proof

The CPA attacker forms two messages in the form:

\[ m_0 = \vec{m} \circ \vec{x} \quad m_1 = \vec{m} \circ \vec{y} \]

where \( \vec{x} \circ \vec{y} \) is concatenation and \( \vec{x} \neq \vec{y} \) are of length \( N \).
Example: Reusing one-time-pad

Proof
The CPA attacker forms two messages in the form:
\[\vec{n}_0 = \vec{m} \oplus \vec{x} \quad \vec{n}_1 = \vec{m} \oplus \vec{y}\]
where \(\vec{x}\oplus\vec{y}\) is concatenation and \(\vec{t} = \vec{m}\) are of length \(N\). Encrypting with the key \(\vec{k}\) of length \(N\) gives
\[E(\vec{k}, \vec{n}_0) = \vec{c} \quad E(\vec{k}, \vec{n}_1) = \vec{d} \oplus \vec{d}\]
where \(\vec{d} = \vec{m} \oplus \vec{k}\) and \(\vec{d} = \vec{m} \oplus \vec{t}\).

Probabilistic crypto system

Definition
...a probabilistic crypto-system is a triple of algorithms:
- key generation \((K_E, K_D): \mathcal{R} \rightarrow \mathcal{K} \times \mathcal{K}\),
- encryption \(E: \mathcal{K} \times \mathcal{K} \times M \rightarrow C\), and
- decryption \(D: \mathcal{K} \times C \rightarrow M\).
When confusion seems unlikely, we abbreviate
- \(K(r)\) to \(k\) and
- \(E(r, k, m)\) to \(E(k, m)\) and even \(E(m)\).

Probabilistic crypto system

Definition
...that together provide
- unique decryption:
  \[D(K_D, E(K_E, m)) = m\]
- secrecy (Shannon: unconditional, "perfect security"):
  \[c \in E(\mathcal{K}, m) \vdash m \in M = \left[m \in M\right] \quad (I\text{-SEC})\]
for every feasible probabilistic algorithm \(A: C \rightarrow M\),
(i.e. \(A: \mathcal{K} \times \mathcal{K} \times C \rightarrow M\))

Probabilistic crypto system

Definition
...that together provide
- unique decryption:
  \[D(K_D, E(K_E, m)) = m\]
- secrecy:
  \[\left[m_0, m_1 \in M, c \in E(\mathcal{K}, m_0) \vdash b \in [0, 1]\right] = \left[m_0, m_1 \in M \vdash b \in [0, 1]\right] = \frac{1}{2} \quad (I\text{-IND})\]
Probabilistic crypto system

Definition
... that together provide
- unique decryption:
   \[ D(\mathbb{K}_D, \mathbb{B}(\mathbb{K}_E, m)) = m \]
- secrecy:
   \[ | m_1, m_2, c \in \mathbb{B}(m_0) \rightarrow b \in A(m_1, m_2, c) | \leq \frac{1}{2} \] (COM-IND)
   for any feasible probabilistic algorithm \( A : M \times M \times C \rightarrow \{0,1\} \) (with \( K_c \) and the seed implicit)

- secrecy (under chosen cyphertext attack):
   \[ | m_1, m_2, c \in \mathbb{B}(m_0) \rightarrow b \in A_2(q_0, m, m_1, c) | \leq \frac{1}{2} \] (IND-CCA)
   for any probabilistic algorithm \( A = (A_0, A_1, A_2) \)...

Example: El Gamal

Fix a finite field \( \mathbb{F} \) and \( g \in \mathbb{F}^* \).
\[
M = \mathbb{R} = \mathbb{F} \\
K_C = \mathbb{F}^* \times \mathbb{F} \\
K_D = \mathbb{F}^* \\
C = \mathbb{F}^* \times \mathbb{F} \\
K = \mathbb{F}^* \\
E(r, k, m) = (g^r, k^* \cdot m) \\
D(K, (c_1, c_2)) = g^c_1 a^c_2 
\]
Example: El Gamal

Fix a finite field $\mathbb{F}$ and $g \in \mathbb{F}$.

$M = \mathbb{R} = \mathbb{F}$
$C = \mathbb{F} \times \mathbb{F}$
$\mathcal{K} = \mathbb{F} \times \mathbb{F}$

$K_E(a) = g^a$
$E(r, k, m) = (g^r, k^r \cdot m)$
$D((c_1, c_2)) = \frac{c_1}{c_2}$

Unique decryption

$D(K_D(a), E(r, K_E(a), m)) = D(a, E(r, g^a, m)) = D(a, (g^r \cdot (g^a)^r \cdot m)) = g^{ar} \cdot m / (g^a)^r = m$

Unconditional security of one-time-pad

Proposition

If all keys are equally likely, then the one-time-pad is unconditionally secure, i.e. it satisfies (IT-SEC).

Proof (continued)

On one hand, for all messages $m$ and ciphertexts $c$ holds

$\Pr[m \in M \mid c \in C] = \Pr[k = c - m \in \mathcal{K}] = \frac{1}{26^n}$

Security of El Gamal

Computational Diffie-Hellman Assumption (CDH)

There is no feasible probabilistic algorithm $CDH : \mathbb{F}^n \rightarrow \mathbb{F}$ such that for all $a, b \in \mathbb{F}$ holds with a high probability

$CDH(g^a, g^b) = g^{ab}$
Security of El Gamal

Computational Diffie-Hellman Assumption (CDH)
There is no feasible probabilistic algorithm \( CDH : \mathbb{Z}^3 \rightarrow \mathbb{F} \) such that for all \( a, b \in \mathbb{F} \) holds with a high probability
\[
CDH(g^a, g^b) = g^{ab}
\]

Decision Diffie-Hellman Assumption (DDH)
There is no feasible probabilistic algorithm \( DDH : \mathbb{Z}^3 \rightarrow \{0, 1\} \) such that for all \( a, b \in \mathbb{F} \) holds with a probability \( \frac{1}{2} \)
\[
DDH(x, y, z) = \begin{cases} 
1 & \text{if } 3uvx = g^y \land y = g^v \land z = g^{uv} \\
0 & \text{otherwise}
\end{cases}
\]

Proposition
El Gamal satisfies (IND-CPA) if and only if (DDH) holds.

El Gamal does not satisfy (IND-CCA).

Security of El Gamal
Recall the definitions:
- unique decryption:
  \[
  D(\mathbb{K}_D, \mathbb{B}(\mathbb{K}_E, m)) = m
  \]
- secrecy (Goldwasser-Micali: “semantic security”)
  \[
  \left[m_0, m_1 \in \mathbb{A}_0, c \in \mathbb{E}(m_0) \right] \Rightarrow
  b \in \mathbb{A}_1(m_0, m_1, c) \leq \frac{1}{2} \quad \text{IND-CPA}
  \]
  for any probabilistic algorithm \( \mathbb{A} = \langle \mathbb{A}_0, \mathbb{A}_1 \rangle \)

Security of El Gamal
Recall the definitions:
- unique decryption:
  \[
  D(\mathbb{K}_D, \mathbb{B}(\mathbb{K}_E, m)) = m
  \]
- secrecy (under chosen cyphertext attack):
  \[
  \left[ c_0 \in \mathbb{A}_0, m \in D(c_0), m_0, m_1 \in \mathbb{A}_0(c_0), c \in \mathbb{E}(m_0) \right] \Rightarrow
  b \in \mathbb{A}_1(c_0, m, m_0, m_1, c) \leq \frac{1}{2} \quad \text{IND-CCA}
  \]
  for any probabilistic algorithm \( \mathbb{A} = \langle \mathbb{A}_0, \mathbb{A}_1, \mathbb{A}_2 \rangle \)

Proof of (DDH) \( \Rightarrow \) (IND-CPA)
Suppose \( \neg \) (IND-CPA).
This means that there is a feasible probabilistic algorithm \( \mathbb{A} = \langle \mathbb{A}_0, \mathbb{A}_1 \rangle \) which
- generates \( m_0, m_1 \in \mathbb{A}_0(k) \), and then
- guesses \( b \in \mathbb{A}_1(k, m_0, m_1, c_0) \) with a probability \( \geq \frac{1}{2} \)
- where \( c_0 = E(a, k, m_0) \) for \( b \in \{0, 1\} \).
Security of El Gamal

Proof of (DDH) ⇒ (IND-CPA)

Suppose ¬(IND-CPA).
This means that there is a feasible probabilistic algorithm \( A = (A_0, A_1) \) which
> generates \( m_0, m_1 \in \mathcal{A}_0(k) \), and then
> guesses \( b \in \mathcal{A}_1(k, m_0, m_1, c_0) \) with a probability \( > \frac{1}{2} \)
> where \( c_0 = E(s, k, m_i) \) for \( b \in \{0, 1\} \).

We construct the algorithm \( \text{DDH} : \mathbb{F}^2 \rightarrow \{0, 1\} \) to decide whether a triple \((x, y, z)\) is in the form \((g^u, g^v, g^w)\) for some \(u, v \in \mathbb{F}\).

Security of El Gamal

Proof (continued)

If the private key \( K^0 = u \), then El Gamal encrypts
\[
E(v, g^u, m) = (g^u, g^{uv} \cdot m)
\]
This means that
\[
\text{DDH}(x, y, z) = 1 \iff \forall m \exists (x, m) = (y, z, m)
\]

Security of El Gamal

Homework

Complete the proof of the Proposition, showing that
> (IND-CPA) ⇒ (DDH)
> (IND-CCA) does not hold.

Outline

Information, channel security, noninterference
Encryption and decryption
Cryptanalysis and notions of secrecy
Cyphers and modes of operation
Modes of operation
Composite cryptosystems
Key establishment
What did we learn?
Modes of operation

ECB
CCB
(Ramzan)

Composite cryptosystems

Shannon's group algebra.
We mix and compose
- substitution cyphers and
- transposition cyphers

In diagrams, substitutions are boxes; but transpositions are knots of threads.

Feistel cyphers are a standardized form to perform a simple transposition: they split the output in two sets of strings, and send them to different places.

Algebra of dataflow

There is a whole algebra of transpositions. Transpositions are the terms of an algebra where each variable must be used exactly once. (Pitts-Gabbay: names, variables, nonces.)
The Feistel cypher and the modes of operation are very special terms in this algebra.
DES and AES.

Outline

Information, channel security, noninterference
Encryption and decryption
Cryptanalysis and notions of secrecy
Cyphers and modes of operation
Key establishment
- "Programming Satan's computer"
  Diffie-Hellman Key Agreement
  Needham-Schroeder Public Key Protocol

What did we learn?

Key establishment

Traditionally, keys sent through a secure channel
- messenger, direct handover, physical protection
Key establishment

- Traditionally, keys sent through a secure channel
  - messenger, direct handover, physical protection
- In cyberspace, there are no secure channels
  - only you and me and cryptography

Key establishment in cyberspace

What is cyberspace?

- space of costless communication
  - instantaneous message delivery
  - any two nodes are neighbors: no notion of distance
- end-to-end architecture (TCP, UDP)
  - simple network links
  - smart network nodes ("ends")
- "Satan’s computer" (Ross Anderson)
  - network controlled by the adversaries: Eve, Satan
  - security only through crypto at the "ends"

Generate your own public key

- El Gamal: Alice generates $K = (g^a, a)$
  - she picks $K_0 = a$
  - computes $K_E = g^a$ and
  - sends $K_E$ to Bob
Key establishment in cyberspace

Generate your own public key
  - **El Gamal**: Alice generates $K = \langle g^a, a \rangle$
    - she picks $K_0 = a$
    - computes $K_e = g^a$ and
    - sends $K_e$ to Bob
  - **RSA**: Alice generates $K = \langle (n, e) \rangle$
    - she picks large primes $p$ and $q$ and sets $n = pq$
    - picks $e \in \mathbb{Z}_n^*$
      - computes $K_0 = d = e^{-1} \mod (p - 1)(q - 1)$
      - sends $K_e = (n, e)$ to Bob

Problem
Eve can impersonate Alice
  - Eve can generate $K_e$ and $K_D$
  - send $K_D$ to Bob
  - and say “Hi, Alice here, this is my key”.
  - Bob encrypts his messages to Alice by $K_E$
  - Eve decrypts them by $K_D$.

Two party key agreement

Diffie-Hellman Key Agreement Protocol (DHKA)

```
\begin{array}{c}
A \\
\uparrow x \\
\downarrow y \\
B \\
\end{array}
```

A to $B g^x$
B to $A g^y$

$g^{AB} = (g^y)^x$

```
\begin{array}{c}
A \\
\uparrow x \\
\downarrow y \\
B \\
\end{array}
```

A to $B g^x$
B to $A g^y$

$g^{AB} = (g^y)^x$

```
\begin{array}{c}
A \\
\uparrow x \\
\downarrow y \\
B \\
\end{array}
```

A to $B g^x$
B to $A g^y$

$g^{AB} = (g^y)^x$

Two party key agreement

Attack on DHKA

```
\begin{array}{c}
A \\
\uparrow x \\
\downarrow y \\
B \\
\end{array}
```

A to $B g^y$
B to $A g^x$

$g^{AB} = (g^y)^x$

```
\begin{array}{c}
A \\
\uparrow x \\
\downarrow y \\
B \\
\end{array}
```

A to $B g^x$
B to $A g^y$

$g^{AB} = (g^y)^x$

```
\begin{array}{c}
A \\
\uparrow x \\
\downarrow y \\
B \\
\end{array}
```

A to $B g^x$
B to $A g^y$

$g^{AB} = (g^y)^x$

```
\begin{array}{c}
A \\
\uparrow x \\
\downarrow y \\
B \\
\end{array}
```

A to $B E^y(x, A)$
B to $A E^y(x, y)$

Bootstrapping key agreement

Needham-Schroeder Public Key Protocol (NSPK)

```
\begin{array}{c}
A \\
\uparrow x \\
\downarrow y \\
B \\
\end{array}
```

A to $B E^y(x, A)$
B to $A E^y(x, y)$

$g^{AB} = (g^y)^x$
### History of NSPK

- NSPK was proposed by in a seminal paper in 1978.
- It was often used and studied.
- In 1996, Gavin Lowe found the attack using the FDR (Failure Divergence Refinement) checker as a part of his project work at Comlab.
Lessons about the bad information flows

- information leaks through interference of resources
  - covert channels are hard to eliminate
  - formal models help prevent Trojan intrusions

- secrecy is achieved in complicated ways
  - some of the “purest” maths became the most applied
  - public key crypto needed a public science of crypto

- but cryptanalysis is also hard
  - encryptions are not broken every day
  - most security failures arise from protocol failures
Lessons about computation

- The simple insights that
  - some computations are hard to invert
    - e.g., getting $p$ or $q$ from $pq$, or $a$ from $g^a$ and $g$
  - some informations are hard to guess
    - if the source is large and unbiased

  point to the important lesson that
  - complexity and
  - randomness

are powerful computational resources.

...are used to push good information flows

- The absence of bad information flows
  - is a fulcrum to move the good information flows.

Guiding principles for the next part

- The absence of bad information flows
  - "If noone can forge Alice's signature...
  - is a fulcrum to move the good information flows.
  - ...then this message must be from Alice :)"

The negative can be used as the positive.
Every secret must be authenticated
- to prevent impersonation.
- Most protocol failures are authentication failures.

Every authentication must be based on a secret
- (in cyberspace).
- The chicken and the egg.

Security is always bootstrapped
- secrecy and authenticity are based on each other
- new secrets are derived from old secrets