Lesson 14. Numerical Algorithms (3): Cryptography

I111E – Algorithms and Data Structures

Ryuhei Uehara & Giovanni Viglietta
uehara@jaist.ac.jp & johnny@jaist.ac.jp

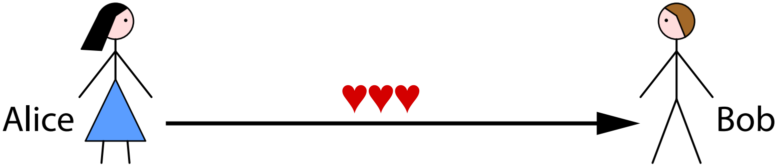
JAIST – December 2, 2019

All material is available at
www.jaist.ac.jp/~uehara/course/2019/i111e

Topics of today's lecture

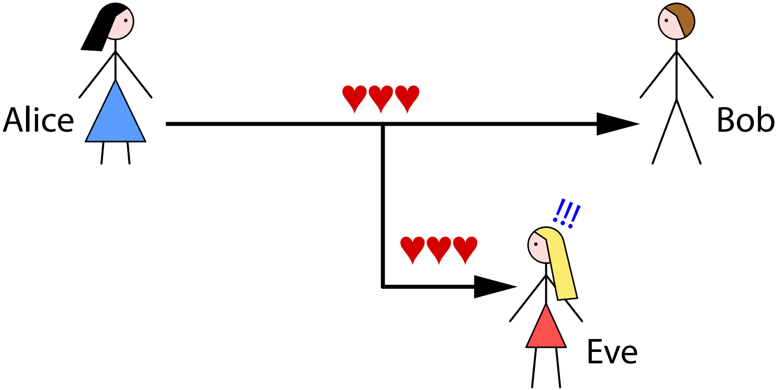
- Private-key cryptography
  - Caesar cipher
  - One-time pad
- Public-key cryptography
  - RSA cryptosystem

Cryptography: setting



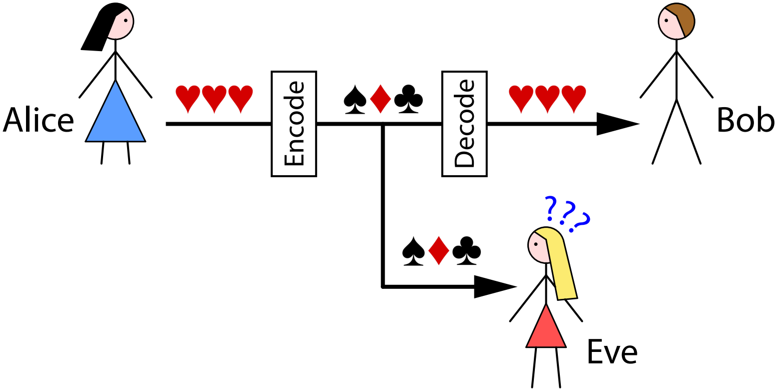
Alice wants to send a secret message to her friend Bob.

Cryptography: setting



But an *eavesdropper*, Eve, can intercept and read the message.

Cryptography: setting

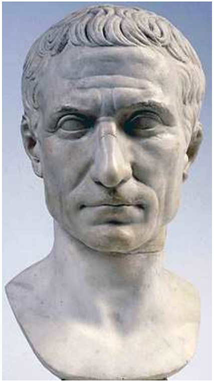


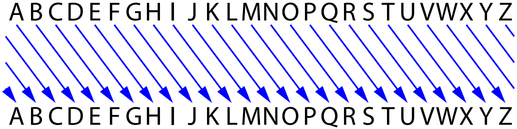
Alice must find a way to *encode* the message, so that:

- Bob can *decode* it and read it,
- Eve cannot decode it even if she intercepts it.

Caesar cipher

The Caesar cipher is one of the first ciphers in history, used by the ancient Roman general Julius Caesar in his private correspondence.

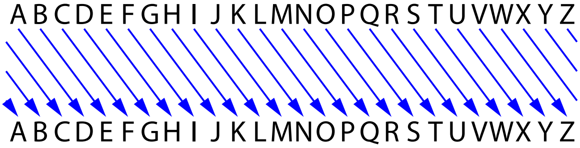




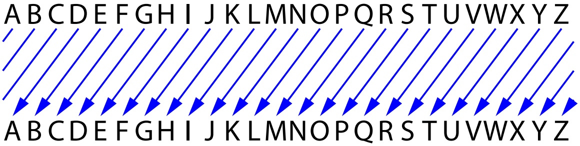
Each letter is shifted by 3 positions down the alphabet.

Caesar cipher

So, the message "DEAR BOB, HOW ARE YOU?"
becomes "GHDU ERE, KRZ DUH BRX?"



To decode the message, Bob applies the reverse transformation:



Caesar-like ciphers

Generalizing, we can shift each letter by any fixed number k of positions down the alphabet:

Each character x of Alice's original message is encoded as $E(x) = (x + k) \bmod 26$.

Each character y in the encrypted message received by Bob is decoded as $D(y) = (y - k) \bmod 26$.

Clearly, $D(E(x)) = ((x + k) - k) \bmod 26 = x$.

Caesar-like ciphers: weaknesses

At the time of Julius Caesar (1st century BC),
this cipher must have been effective enough:

- Most of Caesar's enemies were illiterate,
- The literate ones must have thought the encoded message was probably written in some unknown foreign language.

Caesar-like ciphers: weaknesses

At the time of Julius Caesar (1st century BC),
this cipher must have been effective enough:

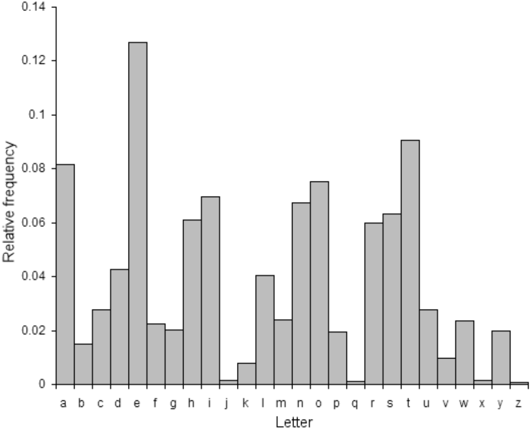
- Most of Caesar's enemies were illiterate,
- The literate ones must have thought the encoded message was probably written in some unknown foreign language.

But by today's standards, Caesar-like ciphers offer no security:

- **Brute-force attack**: if Eve knows that Alice and Bob are using a Caesar-like cipher, she can guess the shift value k by trying to decode the message in the 26 possible ways.
- **Frequency attack**: even if Eve does not know about Caesar-like ciphers, she can do some frequency analysis. E.g., if she knows that the original message is in English, she infers that the most frequent letter must correspond to E, etc.

Frequency analysis of English texts

By counting the appearance rate of every letter in the encoded message, Eve can guess the most frequent letters: E, T, A, etc.



Once she knows the most frequent letters, she can guess entire words, which give her more letters, until the message is decoded.

One-time pad

Here is a better scheme, the <u>one-time pad</u>:

- Alice and Bob privately agree on a secret binary sequence r.
- When Alice wants to send a message to Bob, she converts it to a binary string x, and encodes it as $y = x \oplus r$ (cf. report 1).
- Bob receives y and decodes it in the same way: $y \oplus r$.

The one-time pad works because

$$(x \oplus r) \oplus r = x \oplus (r \oplus r) = x \oplus 0 = x.$$

One-time pad

Here is a better scheme, the one-time pad:

- Alice and Bob privately agree on a secret binary sequence r.
- When Alice wants to send a message to Bob, she converts it to a binary string x, and encodes it as $y = x \oplus r$ (cf. report 1).
- Bob receives y and decodes it in the same way: $y \oplus r$.

The one-time pad works because

$$(x \oplus r) \oplus r = x \oplus (r \oplus r) = x \oplus 0 = x.$$

Example: if $r = 01110010$ and $x = 11110000$,

then Alice sends $y = 11110000 \oplus 01110010 = 10000010$.

Bob decodes it as $10000010 \oplus 01110010 = 11110000 = x$.

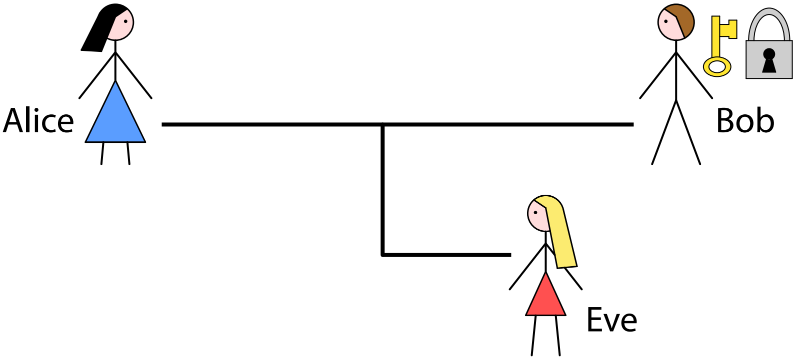
One-time pad: disadvantages

The one-time pad does not have the security flaws of a Caesar-like cipher, because a letter is not always encoded in the same way. However, the one-time pad has other disadvantages:

- The "key" r should be as long as the message x. If Alice and Bob want to send more messages, they have to agree on a longer key. (What if Alice used the same key r to encode two messages x and x' as $x \oplus r$ and $x' \oplus r$? Then Eve could intercept them and compute $(x \oplus r) \oplus (x' \oplus r) = x \oplus x'$, obtaining information on x and x'.)
- Alice and Bob have to agree on a key privately. This means that they should be able to communicate safely at least once. What if this is impossible? (E.g., internet money transactions)

Public-key cryptography

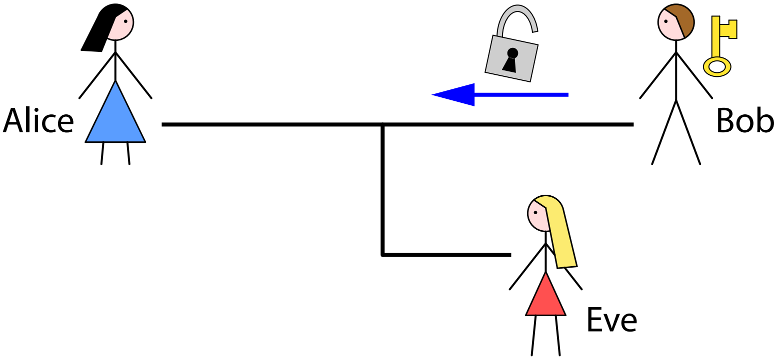
Public-key cryptography is a completely different system:



- Bob has a lock and a key. He sends the open lock to Alice.
- Alice puts her message in a box and locks it with Bob's lock. Then she sends the box to Bob.
- Bob receives the box and unlocks it with his own key.
- Eve cannot open the box because she does not have Bob's key.

Public-key cryptography

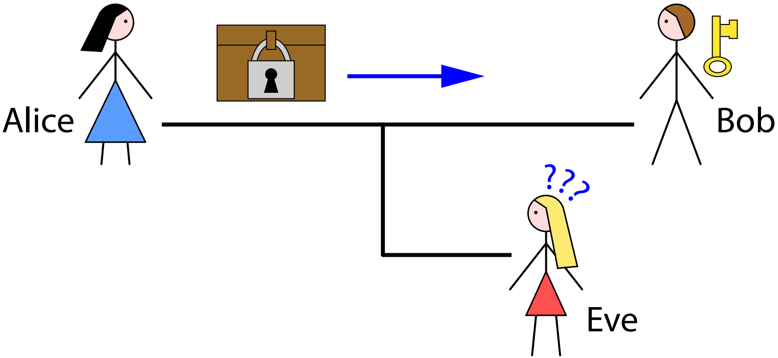
Public-key cryptography is a completely different system:



- Bob has a lock and a key. He sends the open lock to Alice.
- Alice puts her message in a box and locks it with Bob's lock. Then she sends the box to Bob.
- Bob receives the box and unlocks it with his own key.
- Eve cannot open the box because she does not have Bob's key.

Public-key cryptography

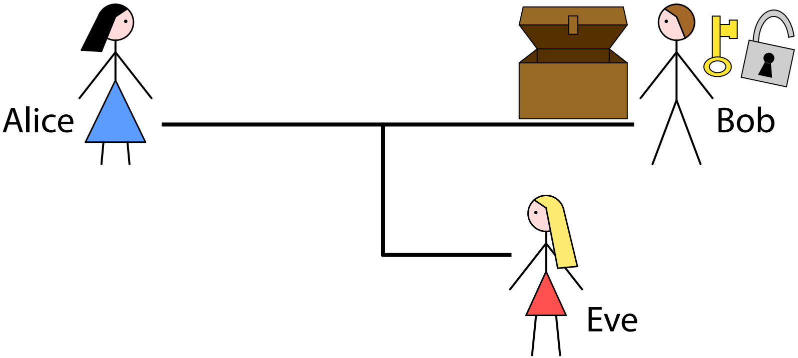
Public-key cryptography is a completely different system:



- Bob has a lock and a key. He sends the open lock to Alice.
- Alice puts her message in a box and locks it with Bob's lock. Then she sends the box to Bob.
- Bob receives the box and unlocks it with his own key.
- Eve cannot open the box because she does not have Bob's key.

Public-key cryptography

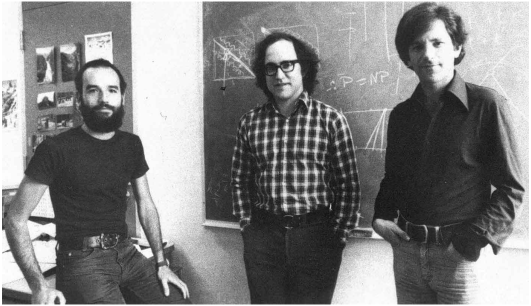
Public-key cryptography is a completely different system:



- Bob has a lock and a key. He sends the open lock to Alice.
- Alice puts her message in a box and locks it with Bob's lock. Then she sends the box to Bob.
- Bob receives the box and unlocks it with his own key.
- Eve cannot open the box because she does not have Bob's key.

RSA cryptosystem

A popular and practical public-key cryptosystems is RSA, which was published in 1977 by Rivest, Shamir, and Adleman:



RSA cryptosystem

First, Bob chooses a public key (i.e., the "lock")
and a private key (i.e., Bob's own "key"):

- Bob randomly picks two large prime numbers p and q.
- Bob computes $N = pq$ and $\varphi = (p-1)(q-1)$.
- Bob chooses an integer e relatively prime to φ.
- Bob computes d, the inverse of e modulo φ. (e is invertible, why?)
- Bob publishes (e, N): his public key, everyone can see it.
- The pair (d, N) is Bob's private key: no one else knows it.

RSA cryptosystem

First, Bob chooses a public key (i.e., the "lock") and a private key (i.e., Bob's own "key"):

- Bob randomly picks two large prime numbers p and q.
- Bob computes $N = pq$ and $\varphi = (p-1)(q-1)$.
- Bob chooses an integer e relatively prime to φ.
- Bob computes d, the inverse of e modulo φ. (e is invertible, why?)
- Bob publishes (e, N): his public key, everyone can see it.
- The pair (d, N) is Bob's private key: no one else knows it.

Then, whenever Alice wants to send a message x to Bob:

- Alice looks up Bob's public key (e, N).
- Alice computes $y = x^e \mod N$ and sends it to Bob.

When Bob receives a message y:

- Bob remembers his private key (d, N).
- Bob decodes y by computing $y^d \mod N$.

Why RSA works

Why does RSA work? Why is $y^d \bmod N$ the same as x?

Theorem: for every x, we have $(x^e)^d \equiv x \pmod{N}$.

Proof: e and d are inverses modulo φ, so $ed \equiv 1 \pmod{\varphi}$, or equivalently $ed = k\varphi + 1 = k(p-1)(q-1)+1$, for some k.

Our claim: $x^{ed} - x = x^{k(p-1)(q-1)+1} - x$ is a multiple of $N = pq$.

By Fermat's little theorem, $x^{p-1} \equiv 1 \pmod{p}$. It follows that

$x^{k(p-1)(q-1)+1} - x = (x^{p-1})^{k(q-1)} \cdot x - x \equiv x - x \equiv 0 \pmod{p}$.

So $x^{ed} - x$ is a multiple of p. By a symmetric argument, it is also a multiple of q. But p and q are primes, so it is also a multiple of pq.

Why RSA is secure

All the operations Alice and Bob have to do are easy:

- Bob finds two random primes p and q in $O(n^4)$ average time,
- Bob computes N and φ in $O(n^2)$ time,
- Bob picks e and inverts it modulo φ in $O(n^3)$ time,
- Alice encodes x by modular exponentiation in $O(n^3)$ time,
- Bob decodes y by modular exponentiation in $O(n^3)$ time.

Why RSA is secure

All the operations Alice and Bob have to do are easy:

- Bob finds two random primes p and q in $O(n^4)$ average time,
- Bob computes N and φ in $O(n^2)$ time,
- Bob picks e and inverts it modulo φ in $O(n^3)$ time,
- Alice encodes x by modular exponentiation in $O(n^3)$ time,
- Bob decodes y by modular exponentiation in $O(n^3)$ time.

If Eve intercepts a message from Alice to Bob and wants to decode it, she has to perform complex operations:

- She knows e, N, and $x^e \bmod N$. But she cannot easily compute x from these numbers: no efficient algorithm is known for the "modular root" problem.
- She could try to find p and q by factoring N, and so compute φ, and invert e modulo φ to find Bob's private exponent d. But no efficient factorization algorithm is known.

RSA and digital signatures

Now Eve is sending messages to Bob pretending to be Alice. So, Alice and Bob need an *authentication method*: a way Bob can tell which messages come from Alice and which are forged.

Digital signature: Alice chooses her own public key (e', N') and private key (d', N') according to the RSA scheme.

- Alice first encodes her message x using her own private key, obtaining $y = x^{d'} \bmod N'$.
- Alice then encodes y a second time using Bob's public key, obtaining $z = y^e \bmod N$.
- When Bob receives z, he decodes it with his own private key, obtaining y.
- Then, Bob uses Alice's public key on y to obtain x.
- If Eve tried to forge messages without knowing Alice's private key, Bob would end up obtaining a meaningless message.