History of crypto in a nutshell

- until the second half of the 20th century:
  - cryptography = encryption, ciphers
  - almost exclusively used in military and diplomacy

- from the second half of the 20th century:
  - cryptography is increasingly used in business applications (banking, electronic funds transfer)
  - besides confidentiality, integrity protection, authentication, and non-repudiation becomes important too

- from the end of the 20th century:
  - cryptography is used in everyday life of people (although they may be unaware of that)
    » SSL/TLS – security web transactions
    » GSM/3G security – subscriber authentication, encryption on the air interface
    » WiFi, Bluetooth, smart cards, ...
Basic model

sender
\[\text{key}\]
data
ENCODING
receiver
\[\text{key}\]
data
DECODING
eavesdropping

Histrorical ciphers

- Skytale from Sparta
- Caesar cipher
- Vigenère cipher (le chifre indéchiffrable)
- German Enigma from WWII
Skytale

- used by the Spartans in the 3rd century BC
- *transposition cipher* (mixes letters of the plaintext)
- encoding and decoding:

- the key is the (diameter of the) rod
- key space is small \(\rightarrow\) easy to break

Caesar cipher

- used by Julius Caesar
- *substitution cipher* (replaces letters of the plaintext)
- each letter is replaced by the letter at some fixed number of positions (e.g., 3) down the alphabet

plain: \[A B C D E F G H I J K L M N O P Q R S T U V W X Y Z\]
cipher: \[D E F G H I J K L M N O P Q R S T U V W X Y Z A B C\]

example: \[CRYPTOGRAPHY \rightarrow FUBSWRJUDSKB\]

- the key is the value of the shift (of the alphabet)
- size of the key space is \(26-1 = 25\) \(\rightarrow\) easy to break
Monoalphabetic substitution

- generalization of the Caesar cipher
- replacement of letters is determined by a permutation

cipher: H T K C U O I S J Y A R G M Z N B V F P X D L W Q E

example: CIPHER → KJNSUV

- the key is the permutation
- the key space is huge: \(26! \approx 1.56 \times 2^{88}\)

  » time left until the next ice age ………………………………………… 239 sec
  » time left until the Sun becomes a supernova …………………… 255 sec
  » age of the Earth …………………………………………………………… 255 sec
  » age of the Universe ………………………………………………………… 259 sec

Breaking monoalphabetic substitutions

- every language has its own letter statistics
  - there are letters that are more frequently encountered than others
    e.g., in English:
    e → 12.7%, t → 9.1%
  - and letters that are less frequent
    e.g., in English:
    z → 0.1%, j → 0.2%

- in case of monoalphabetic substitution, the ciphertext preserves the letter statistics of the original plaintext!
  - after decoding the most frequent and least frequent letters, the rest of the text can be figured out much like solving a crossword puzzle
### The Vigenère Code

#### Coding:

<table>
<thead>
<tr>
<th>Key: RELAT IONSR ELA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plaintext: TOBEO RNOTT OBE</td>
</tr>
<tr>
<td>Ciphertext: KSMEH ZBBLK SME</td>
</tr>
</tbody>
</table>

#### Decoding:

<table>
<thead>
<tr>
<th>Key: RELAT IONSR ELA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ciphertext: KSMEH ZBBLK SME</td>
</tr>
<tr>
<td>Plaintext: TOBEO RNOTT OBE</td>
</tr>
</tbody>
</table>

---

**History of Cryptography | 9/42**

---

**History of Cryptography | 10/42**
The Enigma

- first electro-mechanical cipher
- patented by Arthur Scherbius in 1918
- adopted by the German Army in 1926

Main components of the Enigma

- four main components:
  - keyboard
    - for input of the plaintext / ciphertext
  - lampboard
    - for display of the ciphertext / plaintext
  - plugboard
    - for swapping some input letter pairs
  - scrambler unit (including the rotors)
    - producing the ciphertext from the plaintext (and vice versa)
The rotors

The scrambler unit and the plugboard
Enigma in action

- rotor advances automatically
- set rotors

Enigma key space

- the key consists of the following basic settings:
  - letter pairs swapped (e.g., A/L – P/R – T/D – B/W – K/F – O/Y)
  - order of rotors in the slots (e.g., II – III – I)
  - initial position of the rotors (e.g., R – D – D)

- key space size:
  100391791500 x 6 x 26^3 \sim 2^{53}

- yet, Enigma was broken by the Allies in WWII
  - exploiting protocol weaknesses and weak keys
  - code breaking was partly automated \rightarrow birth of first computers
  - credit goes to Marian Rejewski and Alan Turing
Breaking the Enigma

- every morning, the Germans distribute a daily key to their units to be used with Enigma.
- however, they do not directly use the daily key to encrypt messages.
- instead:
  - they generate a fresh message key for every message.
  - they encrypt the message key with the daily key, and send this at the beginning of the communication.
  - they encrypt the message with the message key, and send it to the receiver.
  - the receiver first decrypts the message key with the daily key and then decrypts the message with the message key.
- in order to cope with errors during transmission, the message key is repeated twice at the beginning of the message!

Example:

```
PGHPGH

ATTACKATMIDNIGHT

KIVBJEGHIOPERLWRMLSAUK
```

From 1933, Poland was able to routinely break encrypted German communications.

Rejewski thought that the repetition of the message key at the beginning of the message is a weakness that may be exploited.
- a guess for the daily key can be confirmed by checking if decoding with the guessed key produces a repeating letter triplet at the beginning of the decoded message.

The Polish codebreakers built a machine that tried different guesses for the daily key in an automated way.
- the machine consisted of 6 Enigma copies (each corresponding to one of the 6 possible rotor orders).
- the machine continuously modified the position setting of the rotors, and attempted decrypting some intercepted message, until it found the daily key.

From 1933, Poland was able to routinely break encrypted German communications.
Breaking the Enigma

- in December 1938, the Germans increase the security of the Enigma
  - they introduce 2 new rotors (operators have to choose 3 rotors out of 5, and the order in which they are put in the machine → this increases possible rotor placements from 6 to 60)
  - they increase the number of letter pairs swapped on the plugboard from 6 to 10
  - key space grows to \(~2^{66}\)
- in April 1939, Hitler breaks the non-aggression treaty with Poland
- in July 1939, Poland reveals their Enigma breaking capability to England
- on August 16, 1939, the design documents of the Enigma breaking machine are transferred to London
- on September 1, 1939, Germany invades Poland

Breaking the Enigma

- some weaknesses exploited by the British
  - cillies
    - German Enigma operators sometimes used very weak (far from random) message keys (e.g., QWE, BNM)
    - an operator always used the same message key (C.I.L.) → perhaps the initials of his wife or girlfriend?
    - these weak keys were called cillies (~silly)
  - Germans had usage constraints that actually weakened their system
    - rotors had to be changed every day, and the same rotor must not be placed in the same slot on two consecutive days
    - e.g., after I-II-V, they could not use III-II-IV
    - this actually reduced the size of the key space that the British had to search over
Breaking the Enigma

- In September 1939, Alan Turing joins the code breakers in Bletchley Park.
- His task is to find a new method for breaking the cipher that does not rely on the repetition of the message key at the beginning of the coded message.
- Turing invents a new method that is essentially an attack known today as the known-plaintext attack:
  - German messages are well structured.
  - Some messages contain guessable words at guessable locations.
  - E.g., every evening at 6pm, they send a weather forecast, which includes the word “wetter” always at the same position within the message.
- The British build new Enigma breaking machines (Victory, Agnus Dei) based on the plans of Turing in 1940.
- Indeed, Germans change their message key sending protocol in May 1940, but this does not affect the cryptanalytic capabilities of the British anymore.
Modern cryptography

- Shannon’s work on information theoretical characterization of encryption [1948]
- substitution-permutation ciphers and the Data Encryption Standard (DES) [1970’s]
- the birth of public key cryptography [1976-78]
- quantum cryptography [1980’s]

The birth of modern cryptography

- first theoretically sound formulation of the notion of security of an encryption algorithm
  - used information theory to define the concept of perfect secrecy
  - gave necessary conditions for a cipher to be perfectly secure
  - proved that the one-time pad provides perfect secrecy

- practical ideas to build strong block ciphers
  - create a complex cipher by repeated use of otherwise simple transformations
  - none of the simple transformations alone would be sufficiently strong, but their repeated use and the large number of iterations would ultimately result in a strong cipher (aka. product ciphers)
Data Encryption Standard (DES)

- based on Lucifer, a cipher developed by IBM in the 70's
- symmetric key block cipher
- features:
  - Feistel structure (same structure can be used for encoding and decoding)
  - number of rounds: 16
  - input block size: 64 bits
  - output block size: 64 bits
  - key size: 56 bits

HW implementation:
DES chip

DES round function F

- $S_i$ – substitution box (S-box)
  - non-linear look-up tables
- $P$ – permutation box (P-box)
  - linear bit permutation
Security of DES

- average complexity of a brute force attack is $2^{55}$
  - was suspected breakable by NSA back in the 70's
  - definitely became breakable by the late 90's by distributed computing
  - new standard AES was accepted in 2001

- algebraic attacks
  - DES has never been broken in a practical sense
  - best known attacks:
    - linear cryptanalysis (LC)
      - requires $2^{43}$ known plaintext–ciphertext pairs
    - differential cryptanalysis (DC)
      - requires $2^{47}$ chosen plaintexts (and corresponding ciphertexts)
  - DC and LC were discovered in the late 80's and early 90's
  - it was revealed in the late 90's that the designers of DES had known about DC, and optimized the DES S-boxes such that DES provides maximum resistance against DC

A breakthrough in modern cryptography


Ralph Merkle, Martin Hellman, and Whitfield Diffie
The key exchange problem

- by the 70’s digital computers and telecommunication networks were increasingly used in the financial sector
- banks could use symmetric key ciphers, such as Lucifer and later DES, to encrypt sensitive data
- but they faced a practical question: how to setup a shared DES key between two end points (e.g., two remote branches of the same bank) ???
  - in case of earlier military and diplomatic applications, keys were transferred by agents in a physically secure way
  - this was expensive and inflexible for banks

The Diffie-Hellman key exchange protocol

public parameters:
A large prime $p$ and a generator element $g$ of $\mathbb{Z}_p^* = \{1, 2, ..., p-1\}$

Alice
- select random $x$
- compute $g^x \mod p$
- compute $k = (g^y)^x \mod p$

Bob
- select random $y$
- compute $g^y \mod p$
- compute $k = (g^x)^y \mod p$
The Diffie-Hellman key exchange protocol

- if an attacker can only eavesdrop the communications between Alice and Bob, then he has only $g^a \mod p$ and $g^b \mod p$
- to compute $g^{xy} \mod p$, he would need $x$ or $y$
- it is hard to compute $x$ from $g^x \mod p$
  - this is the so-called "discrete logarithm" problem
  - no polynomial time algorithm is known to solve it
  - if $p$ is large, then computing discrete logarithm (mod $p$) is practically infeasible

- there seem to exist one way functions:
  - given $x$, it is easy to compute $f(x)$
  - given $y$, it is hard to find an $x$ for which $y = f(x)$
- can we use such functions to realize a sort of asymmetric key cryptography???

The idea of asymmetric key cryptography

- encoding and decoding keys are not the same (unlike in symmetric key cryptography)
- computing the decoding key from the encoding key is hard (infeasible in practice)
- encoding key can be made public, decoding key should be kept secret
  - anybody can obtain the public encoding key of Alice, and send an encrypted message to her
  - only Alice can decrypt the message with the private decoding key
  - an attacker cannot compute the private key from the public key
  - aka. public key cryptography
  - solves the key exchange problem (but has other issues to solve)
The RSA cryptosystem


key-pair generation algorithm:
- choose two large primes $p$ and $q$ (easy)
- $n = pq$, $\phi(n) = (p-1)(q-1)$ (easy)
- choose $e$, such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$ (easy)
- compute the inverse $d$ of $e$ mod $\phi(n)$, i.e., $d$ such that $ed \mod \phi(n) = 1$ (easy if $p$ and $q$ are known)
- output public key: $(e, n)$ (public exponent and modulus)
- output private key: $d$ (private exponent)

encryption algorithm:
- represent the plaintext message as an integer $m \in [0, n-1]$
- compute the ciphertext $c = m^e \mod n$

decryption algorithm:
- compute the plaintext from the ciphertext $c$ as $m = c^d \mod n$
Security of asymmetric key algorithms

- security is typically related to the difficulty of solving some hard mathematical problem
  - e.g., factoring or discrete logarithm

- provable security by reduction proofs:
  - we show that any efficient algorithm that breaks our crypto scheme could be used to efficiently solve a believed to be hard mathematical problem
  - this means that breaking our crypto scheme is at least as hard as solving the hard mathematical problem

- there exist provably secure crypto systems, but most of them are not efficient (fast) enough for practical applications

- most of the public key crypto schemes that we use in practice are not provably secure (or only partial proofs exist)

Example: Security of the RSA crypto system

- factoring integers is believed to be a hard problem
  - given a composite integer \( n \), find its prime factors
  - true complexity is unknown
  - it is believed that no polynomial time algorithm exists to solve it

- computing \( d \) from \( (e, n) \) is equivalent to factoring \( n \)

- computing \( m \) from \( c \) and \( (e, n) \) may not be equivalent to factoring \( n \) (this is known as the RSA problem)
  - if the factors \( p \) and \( q \) of \( n \) are known, then one can easily compute \( d \), and using \( d \), one can also compute \( m \) from \( c \)
  - we don’t know if one could factor \( n \), given that he can efficiently compute \( m \) from \( c \) and \( (e, n) \)
The secret story of public key cryptography

- Ellis, Cocks, and Williamson worked for GCHQ (British security agency)
- in 1969, Ellis defined the general model of asymmetric key cryptography (called it non-secret key coding)
  - public and private keys
  - (trap-door) one way functions
- in 1973, Cocks invented a cryptosystem same as RSA
  - he was introduced to the idea of non-secret key crypto
  - he worked in the field of number theory, and immediately thought of using factoring as a hard problem
- in 1974, Williamson (a friend of Cocks) invented a key exchange protocol same as the Diffie-Hellman protocol
- by 1975, Ellis, Cocks, and Williamson worked out all the major results of public key cryptography, which were (re)invented some years later
- the story was made public only in 1997
Quantum cryptography

- using quantum effects to solve traditional problems in new ways
  - e.g., quantum key exchange

- using quantum computers to break modern ciphers efficiently
  - e.g., the Schor factorization algorithm to break RSA

General model of cryptographic coding

- **Sender**
  - encoding key
  - DATA
  - ENCODING
  - e.g.: e-mail, file, IP packet
  - spatial or temporal distance

- **Attacker**
  - decoded DATA
  - decoding key = encoding key
  - conventional cryptography (symmetric key crypto)
  - eavesdropping, replay, modification, forgery

- **Receiver**
  - decoding key
  - DATA
  - DECODING
  - decoded DATA
  - checksum → integrity protection, message origin authentication
  - decoding key ≠ encoding key
  - public key cryptography (asymmetric key crypto)
Practical applications of cryptography

- secure communication over public channels / networks
  - WWW (https / TLS)
  - WiFi (WPA, WPA2)
  - GSM/3G
  - Bluetooth
- secure data storage
  - disk encryption (TrueCrypt, BitLocker, ...)
  - encrypted cloud storage (Tresorit, CipherCloud, ...)
- authentication
  - smart cards (e.g., bank cards)
  - ignition keys of cars
  - electronic tickets in public transport (automated fare collection systems)
- software authentication and integrity protection
  - digitally signed code (e.g., drivers, applets, Android packages)
- ...

Further readings