Cryptography and Security

Lecture 1: a bit of history

About the course

Requirements

- Listen to the lectures
- Consultation the week after
- Exam in canvas (probably)
- Option to take a pre-exam
- Brief oral „defense” of the exam

Topics

- Mathematical foundations (algebra, number theory, probability)
- Cryptographic protocols (encryption, hash, signatures)
- Attacks (algorithmic, physical, . . .)

About the course

References

- Katz–Lindell: Introduction to Modern Cryptography
- Schneier: Applied cryptography

Already the ancient greeks . . .

Motivational questions

- Who sends what? (model of communication)
- What is an attack? (threat model)

Simple classical examples

- A message to be sent between two people
- Attack: the enemy intercepts the message

Already the ancient greeks . . .

Atbash

- alphabet: replace kth letter from the front with kth from the back
- first with hebrew alphabet
- Simple „substitution”: $\aleph \leftrightarrow tav, \beth \leftrightarrow shin, \ldots$
- works just as well with any alphabet, e.g. latin

```
Original:  abcdefghijklmnopqrstuvwxyz
Cipher:    zyxwvutsrqponmlkjihgfedcba
```

Properties

- Used in several places in the Bible
- Quite easy to „break”
- fixed permutation: considered weak today

Already the ancient greeks . . .

Caesar cipher

- Replace every letter with the letter that comes 3 positions later

Properties

- linked to Julius Ceasar (1st century B.C.E.)
- easy to break (Al-Kindi, 9th century)
- Still used as recently as 1915 by the Russian army
- A variant today: ROT-13 (e.g. for anti-spoiler forum posts)

Classical cryptography

19th century onwards

- military applications: protect communication
- ad-hoc constructions $\Rightarrow$ breaking is a matter of time only

Classical cryptography in modern terms

Symmetric key cryptography (informally)

- goal: protect communication between two parties
- setup: generate and share *common* secret key
- communication: on an open channel (except for key sharing)
- m : plain text („message”), c : encrypted text („ciphertext”), k : key
- Consists of 3 separate algorithms
  1. Gen – generation of key k. Choose randomly from a predefined set according to a predefined distribution
  2. Enc – encryption. Given the key k and message m, compute $c = Enc_k(m)$
  3. Dec – decryption. Given the key k and ciphertext c, compute $m = Dec_k(c)$

Kerchkoffs's principles

Design principles of encryption methods (Kerckhoffs, 1883)

1. A system must be practically, if not mathematically indecipherable.
2. It should not require secrecy. Should be no problem if enemy intercepts.
3. Key: easy to remember and change
4. Should be communicated via telegraph (low bandwidth:)
5. Should be portable and not require several people to handle, operate.
6. Should be easy to use.

„The" Kerchkoffs's principle

Design principles of encryption methods (Kerckhoffs, 1883)

2. It should not require secrecy. Should be no problem if enemy intercepts.
3. (VS: security through obscurity)

Protect key vs. protect method

- secure distribution and storage
- replace compromised components
- communication with several peers

Conclusion

- goal: without the key, attacker cannot decrypt (or only with enormous effort)
- good encryption algorithms should be public
- public domain vs. security by obscurity

The attacker's toolkit

Traditional methods

- brute-force (try every possible key)
- frequency analysis (entropy of language)
- collision detection
- anything we (the good guys) have not even thought of

Other methods

- reverse-engineering (sec-by-obsc)
- software/hardware attacks (DOS/fault attacks)
- attacks using human weaknesses (social engineering, phishing)
- anything we have not even thought of

The attacker's toolkit

Passive attacks

- ciphertext-only attack: obtain m knowing one or more ciphertexts obtained by using the same k
- known plaintext attack: obtain further pairs (m, c) knowing one or more message-ciphertext pairs obtained by using the same k.

The attacker's toolkit

Active attacks

- chosen plaintext attack: the attacker has access to some message-ciphertext pairs where the messages are chosen by them (k fixed).
- chosen ciphertext attack: the attacker has access to some message-ciphertext pairs where the ciphertexts are chosen by them (k fixed).

Classical cryptographic protocols

Shift cipher

- principle: shift every letter by some (secret) number
- map alphabet to $0, 1, 2, \ldots, 25$, „wrap around": $25 + 1 = 0$.

Shift cipher

1. $Gen : k \in \{0, 1, \ldots, 25\}$ random.
2. Enc : for each character of message:
   $c_i = Enc_k(m_i) \equiv m_i + k \pmod{26}$
3. Dec : for each character of ciphertext:
   $m_i = Dec_k(c_i) \equiv c_i - k \pmod{26}$

Example. $m = \texttt{EXAMPLE} \rightarrow (4, 23, 0, 12, 15, 11, 4)$

1. $Gen : k = 11$
2. $Enc : c = (15, 8, 11, 23, 0, 22, 15) \rightarrow \texttt{PILXAWP}$
3. $Dec : m = (4, 23, 0, 12, 15, 11, 4) \rightarrow \texttt{EXAMPLE}$

Classical cryptographic protocols

Shift cipher

1. $Gen : k \in \{0, 1, \ldots, 25\}$ random.
2. Enc : for each character of message:
   $c_i = Enc_k(m_i) \equiv m_i + k \pmod{26}$
3. Dec : for each character of ciphertext:
   $m_i = Dec_k(c_i) \equiv c_i - k \pmod{26}$

Properties

- trivial to break: try every possible key (brute-force)
- consequence: key space needs to be large
- a *necessary* condition for security (not sufficient though ...)
- E.g. space of 128-bit long 0-1 sequences($= 2^{128}$ possibilities)

Classical cryptographic protocols

Mono-alphabetic replacement

- principle: replace each letter according to a randomly chosen permutation
- key space: all permutations (bijective mappings) on the alphabet

Mono-alphabetic replacement

1. $Gen : k$ a random permutation of the 26 elements
2. Enc, Dec : apply permutation/inverse caharecter by character

Example. $m = \texttt{PERMUTATION}$

1. $Gen : k = \begin{array}{l}\texttt{ABCDEFGHIJKLMNOPQRSTUVWXYZ}\\ \texttt{xeuadnbkvmrocqfsyhwglzijpt}\end{array}$
2. $Enc : c = \texttt{sdhclgxgvfq}, Dec : m = \texttt{PERMUTATION}$

Classical cryptographic protocols

Mono-alphabetic replacement

1. Gen : k a random permutation of the 26 elements
2. Enc, Dec : apply permutation/inverse caharecter by character

Properties

- key space: $26! \approx 2^{88}$ large enough, but ...
- Fixed equivalent of each letter $\Rightarrow$ statistics-based attack using frequencies
- Even short texts (10 words!) usually show statistics close to the global language stats
- Some letters easy to spot, brute force on the rest (hard to automize)
- Only on text that „makes sense”

Classical cryptographic protocols

Vigenère cipher

- principle: hide letter frequencies
- same letter shifted by variable amounts based on the position

Poly-alphabetic shift

1. $Gen : k = (k_1, \ldots, k_t)$ random (t not fixed)
2. $Enc : c_{i+jt} \equiv m_{i+jt} + k_i \pmod{26} : i = 1, \ldots, t, j = 0, \ldots$
3. $Dec : m_{i+jt} \equiv c_{i+jt} - k_i \pmod{26} : i = 1, \ldots, t, j = 0, \ldots$

Example $m = \texttt{THISISIMPOSSIBLE}$

1. $Gen : k = false$ (means a number sequence!)
2. $Enc :$ $$\begin{aligned} m &= \texttt{THISISIMPOSSIBLE} \\ k &= \texttt{falsefalsefalsef} \\ c &= \texttt{YHTIMXIXHSXSTTPJ} \end{aligned}$$

Classical cryptographic protocols

Poly-alphabetic shift

1. $Gen : k = (k_1, \ldots, k_t)$ random (t not fixed)
2. $Enc : c_{i+jt} \equiv m_{i+jt} + k_i \pmod{26} : i = 1, \ldots, t, j = 0, \ldots$
3. $Dec : m_{i+jt} \equiv c_{i+jt} - k_i \pmod{26} : i = 1, \ldots, t, j = 0, \ldots$

Properties

- If t is known $\Rightarrow$ broken (statistics on t-separated subsequences)
- Kasiski's method look for repetitions of 2,3: distances are multiple of $t \Rightarrow t =$ gcd
- count identicals: stats for $c_1, c_{1+k}, c_{2+k}, \ldots \Rightarrow$ ok for long text, short key
- Broken e.g. in American civil war ($k = \texttt{completevictory} \Rightarrow$ easy)

Wrap-up

Summary

- Kerchkhoffs's principle: make system public
- Make practical brute force attack infeasible with large key space
- It's hard to build a secure protocol
- Need science rather than ad-hoc ideas