Cryptology

Two essential parts:

- Cryptography
  - encryption of plaintext to ciphertext (cryptogram)
  - decryption of ciphertext to plaintext
- Cryptanalysis
  - breaking cryptography
  - necessary when developing cryptographic systems!

Crypto systems

A crypto system S can be defined as a tuple

$S = <\boldsymbol{M}, \boldsymbol{C}, \boldsymbol{K}, \boldsymbol{E}, \boldsymbol{D}>$

$\boldsymbol{M}$ – set of plaintexts (messages)

$\boldsymbol{C}$ – set of ciphertexts (cryptograms)

$\boldsymbol{K}$ – set of keys

$\boldsymbol{E}$ – set of encryption functions

$\boldsymbol{D}$ – set of decryption functions

– for every $k \in \boldsymbol{K}$ there are $E_k \in \boldsymbol{E}$, $D_k \in \boldsymbol{D}$ s.t. for all $m \in \boldsymbol{M}$, $D_k(E_k(m)) = m$

– for every $k \in \boldsymbol{K}$ and $c \in \boldsymbol{C}$, there is only one $m \in \boldsymbol{M}$ s.t. $E_k(m) = c$

(i.e., different m can not be encrypted to the same c.)

Properties of crypto systems

1. E_k and D_k must be efficient and easy to use.
2. Algorithms $\boldsymbol{E}$ and $\boldsymbol{D}$ should be assumed known.
3. Without k, it should be infeasible to deduce
    1. m from c, where $c=E_k(m)$
    2. D_k from c even if m is known, for $m=D_k(c)$
    3. E_k from m even if c is known, for $c=E_k(m)$
    4. a (false) c' s.t $E_k(m)=c'$ for any m, unless E_k and m are known.

Classification of crypto systems

1. type of operations used
   - substitution
   - transposition
2. number of keys used by sender/receiver
   - one: symmetric, single–key, secret–key, conventional
   - two: asymmetric, two–key, public–key
3. how the plaintext is processed
   - block cipher: one block (e.g. 64 bits) at a time
   - stream cipher: one element (e.g. 8 bits) at a time

Cryptanalysis attacks

1. Ciphertext only: c (and $\boldsymbol{E}$) known
    - try all keys ("brute force") – impractical with large $\boldsymbol{K}$
    - statistical analysis, given type of plaintext (English etc)
2. Known plaintext: one or more (m,c) pairs known
    - e.g. "login: " prompt, "%PS−" header
3. Chosen plaintext
    - analyst can adapt plaintexts to results of analysis
4. Chosen ciphertext
5. Chosen text
    - useful e.g. for asymmetric crypto systems

Crypto system security

- Unconditionally secure
  - unbreakable regardless of how much ciphertext, time or computing resources
  - **one** such known: one–time–pad
- Computationally secure
  - the cost of breaking the cipher exceeds the value of the encrypted information
  - the time required to break the cipher exceeds the useful lifetime of the information

Computational security

Speed of computation increases rapidly!

ex: brute force average time required

Key size	1 encr/us	10M encr/us
32 bits	36 min	2.15 ms
56 bits	1142 years	10 hrs
128 bits	5.4 · 10e24 years	6.4 · 10e18 years

DES (Data Encryption Standard, 1977) was broken in 22 hrs 15 minutes in January 1999!

(using special hardware + 100,000 PC/workstations)

Classical crypto systems

- Caesar cipher (shift cipher)
  - substitute each element (letter) with the one *n* positions later in the alphabet (modulo length of alphabet)
  - ex: shift 3, English alphabet

```
meet me after the toga party
PHHW PH DIWHU WKH WRJD SDUWB
```

  - Caesar: $n = 3$
  - rot13: $n = 13$

    (used on Internet for possibly offensive jokes)

Shift cipher

- Interpret letters A,B,C,...,Z as numbers 0,1,2,...,25 $= \boldsymbol{Z}_{26}$
- Define

$$c_i = E_k(m_i) = (m_i + k) \bmod 26$$

$$m_i = D_k(\mathrm{c}_i) = (\mathrm{c}_i - k) \bmod 26$$

- Exercise: check that this forms a crypto system!
- Homework: write a program (for Swedish!)

Breaking shift ciphers

- Brute force: try all keys
  - feasible for simple shift cipher, because
    - the algorithm is known
    - there are few keys
    - it is easy to recognize the plaintext

General substitution cipher

- Instead of just shifting the alphabet, mix it:

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

- Now: 26! different keys ($> 4 \cdot 10^{26}$)
  - brute force not feasible

Breaking substitution ciphers

- Easy to break if we know the language
  - analyse the relative frequencies of letters
  - compare and match
- More difficult for short ciphertexts
  - use digram analysis: relative frequencies of pairs of letters
    - trigrams...
- Generally useful method for recognizing plaintext

Improving substitution ciphers

- Multiliteral ciphers
  - Playfair cipher: encrypt digrams to digrams
    - considered "unbreakable", but easily broken with a few hundred letters of ciphertext
  - Hill cipher: encrypt *n* letters to *n* letters
    - difficult with ciphertext–only attack, but easy with known plaintext
- Polyalphabetical ciphers
  - use several different monoalphabetical substitutions

Polyalphabetical ciphers

- Model:

  for $m = m_1,\ldots,m_n$, $c = c_1,\ldots,c_n$, $k = k_1,\ldots k_d$,

  $$c = E_k(m) = f_1(m_1)\cdots f_d(m_d)\cdot f_1(m_{d+1})\cdots f_d(m_{2d})\cdots$$

- Example: Vigenère cipher (1500, "unbreakable" until 1800s)
  - $f_i(m) = m+k_i \pmod{n}$

  i.e., d shift ciphers combined

```
m = R E N A I S S A N C E
k = B A N D B A N D B A N
c = S E A D J S F D O C R
```

- Previous years exercise to break

Breaking Vigenère

- The cipher can be broken because of the periodicity of the key
  - if two identical strings happen to be at the same place relative to the key, they are encrypted the same
  - the distance between repetitions in the ciphertext is a clue to the length of the key
    - common factors give good initial values
  - with the key length d, break d shift ciphers using frequency analysis

Strengthen polyalphabetic ciphers

- Lengthen the key
  - Book ciphers: use a predefined part of a book as key
  - More difficult to break
  - still possible: the key has language structure
- Remove structure of key: one–time–pad
  - use a random key as long as the message
  - use each key only once
  - UNBREAKABLE!

Encrypting binary data

- Vernam cipher:
  - $c_i = E_k(m_i) = m_i \oplus k$
  - $m_i = D_k(c_i) = c_i \oplus k$

  $m = (m \oplus k) \oplus k = m \oplus (k \oplus k) = m \oplus 0 = m$
- Very good for one−time−pad
- Very bad if you use the same key twice
  - known plaintext attack:
    $c_1 \oplus m_1 = (m_1 \oplus k) \oplus m_1 = (m_1 \oplus m_1) \oplus k = k$

Generalising shift ciphers

- Affine ciphers
  - for $\boldsymbol{M}=\boldsymbol{C}=\boldsymbol{Z}_n$, define
    $c = E_k(m) = (a\ m + b) \bmod n$
    $m = D_k(c) = (a'\ m + b') \bmod n$
    where k is (a,b) resp (a',b')
  - Must choose k s.t. E_k is injective (reversible)
    ex: (2,1) is not a good key in $\boldsymbol{Z}_{26}$
    - $(2 \cdot 13 + 1) \bmod 26 = 1$
    - $(2 \cdot 0 + 1) \bmod 26 = 1$

Modular arithmetic

- $a \equiv b \pmod{n}$ if $a-b = kn$ for some k
  - e.g. $17 \equiv 7 \pmod{5}$
- Write $a \bmod n = r$
  if r is the (positive) residue of a/n
  - implies $a \equiv r \pmod{n}$
- Let $\Diamond$ be an operation: $+, -, \cdot$. Then

  $(a \Diamond b) \bmod n = ((a \bmod n) \Diamond (b \bmod n)) \bmod n$
- $(\mathbf{Z}_n, \{+, -, \cdot\})$ is a commutative ring:
  usual commutative, associative, distributive laws

Modular arithmetic (cont)

x is the multiplicative inverse of a modulo n, written a^{-1}, if $ax \equiv 1 \pmod{n}$

- Ex: $3 \cdot 5 \equiv 1 \pmod{14}$

The reduced set of residues modulo n is $\boldsymbol{Z}^{*}_{n} = \{\, x \in \boldsymbol{Z}_{n} - \{0\} : \gcd(x,n) = 1 \,\}$

Euler's totient function $\phi(n)$ is the cardinality of $\boldsymbol{Z}^{*}_{n}$

Ex: $\boldsymbol{Z}^{*}_{24} = \{ 1, 5, 7, 11, 13, 17, 19, 23 \}$, $\phi(24)=8$

Affine ciphers (cont)

- To choose a key s.t. encryption is injective,
  $a\,m + b \equiv c \pmod{n}$
  $a\,m \equiv c-b \pmod{n}$
  must have one solution wrt. m.
- If $\gcd(a,n) > 1$, then, e.g., $a\,m \equiv 0 \pmod{n}$ has two solutions $m = 0$ and $m = n/(\gcd(a,n))$
- If $\gcd(a,n) = 1$, then if there are two solutions d,d' so $a\,d \equiv a\,d' \pmod{n}$. Then $a(d-d') \equiv 0 \pmod{n}$, so $n | a(d-d')$, and since $\gcd(a,n)=1$, $n | (d-d')$, and $d \equiv d' \pmod{n}$, so the solution is unique!

Affine ciphers (cont)

- Furthermore, as x varies over $\boldsymbol{Z}_n$ and $\gcd(a,n)=1$, $(a\,x + b) \bmod n$ will have n different values, so $a\,x \equiv b \pmod{n}$ has a unique solution for every value of b.
- The number of keys a s.t. $\gcd(a,n)=1$ is $\phi(n)$, so the number of keys (a,b) of an affine cipher is $n \cdot \phi(n)$.
  - ex: an affine cipher on $\boldsymbol{Z}_{26}$ has $26 \cdot 12=312$ keys.

Transposition ciphers

- Columnar transposition
  - write the plaintext row by row in a rectangle, read the ciphertext column by column (for some permutation of columns)
- Periodic permutation
  - permute the columns, read row by row (easier to encrypt row–by–row, don’t need all cleartext)
- Broken by frequency analysis of digrams/trigrams

Strengthening ciphers

- Try combining several keys: *product ciphers*
  - $c = E_n(E_{n-1}(\ldots(E_1(m))\ldots)) = E'_k(m)$
  - $m = D_1(D_2(\ldots(D_n(c))\ldots)) = D'_k(c)$
- If $E'_k = E_{k'}$ for some single k', the cipher is *idempotent* (ex: shift cipher)
- Two crypto systems S_1 and S_2 *commute* if $S_1\,S_2 = S_2\,S_1$
- A crypto system S is *idempotent* if $S = S\,S$
- A function f is an *involution* if $f(f(x)) = x$

Product ciphers

- If S_1 and S_2 are idempotent and commute, then their product is also idempotent.
  - so no strength is won
  - e.g. Caesar cipher
- So: for a product cipher to increase cryptographic strength, its parts must not be idempotent!
- The composition of two involutions is not necessarily an involution.