Lecture topics

- Cryptology
- Encryption
- Secure hash

Security and cryptology

- Information security = crypto?
- A significant art by itself
  ⇒ not something average M.Sc (Eng) must master
- Cryptography only a part of solution
- One need to know
  - which cryptographic methods to select
  - how to use those (and how not to use)
- Significant development in last 30, 10 years

What we like to do?

1. Conceal information
2. Verify information integrity
3. Make sure that we have access to information
- The first two are (somewhat) served with cryptology
- For the third one cryptography may help — and may harm
  - because of how many security protocols are designed, DoS with cryptographic methods can be easy: for example in initial handshake the server may need to do complex calculations to determine that the other party is not authorised.

Terminology

**Plaintext** $M$ is information we want to protect

**Ciphertext** $C$ is protected form of $M$

**Enciphering** transforms plaintext to ciphertext

**Deciphering** transforms ciphertext to plaintext

**Key** $K$ is used to decipher ciphertext

**Public key** $K_p$, anyone can have access to
Secret key $K_s$ only owner may have access to

Initialisation Vector $IV$ is known parameter

Message digest $H = h(M)$, fixed length value

**Terminology: Attacks**

Attacks to defeat cryptography

**Ciphertext only** is known, and one tries to find the corresponding plaintext (and the key)

**Known plaintext** and corresponding ciphertext is known: one tries to find the key

**Chosen plaintext** attack: attacker can feed plaintexts of one’s choice to the system and learn corresponding ciphertexts

**Kerckhoffs’ six design principles**

1. The system must be practically, if not mathematically, indecipherable;
2. It must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience;
3. Its key must be communicable and retainable without the help of written notes, and changeable or modifiable at the will of the correspondents;
4. It must be applicable to telegraphic correspondence;
5. It must be portable, and its usage and function must not require the concourse of several people;
6. Finally, it is necessary, given the circumstances that command its application, that the system be easy to use, requiring neither mental strain nor the knowledge of a long series of rules to observe.

The more secrets the system has, the more brittle it is. A key is the easiest component to change. This principle is also known as Shannon’s Maxim [3].

**Design principles**

- **Confusion**
  - complex relationship between $K$ and $C$
  - e.g. substitution

- **Diffusion**
  - no statistical relationship between $M$ and $C$
  - one-bit change in $M$ results change in every bit in $C$ with $P = \frac{1}{2}$
  - avalanche effect
  - e.g. transposition

**Ciphers**

- **Substitution cipher**
  - plaintext $M$ enciphers always to $C$ with key $K$
  - Caesar cipher: $C = (M + K) \mod 26, K = 3$
  - modern block ciphers (block size 64 bits(8 bytes: $1.8 \times 10^{19}$ different blocks) or more)

- **One-time pad**
- unbreakable cipher (Vernam cipher)
  - if $K$ used only once
  - $K$ as many bytes as $M$
  - stream ciphers emulate

- Message digests
  - take arbitrary long $M$ producing fixed-length digest $D$

**Block ciphers**

- Few basic types
  - SP-networks (substitution-permutation networks)
  - Feistel ciphers
    * data halved, halves mixed with round function

**Block ciphers: operation modes**

**Electronic code book** (ECB) used as substitution cipher

- same $M$ encrypts to the same $C$ with same $K$
- vulnerable to cut-and-splice

**Cipher block chaining** (CBC) uses previous ciphertext $C_{i-1} \oplus M_i$

- initialisation vector $IV$
  $\Rightarrow$ randomises first block $IV \oplus M_1$
- still possible to defeat integrity

**Output feedback** (OFB) used as stream cipher $K_i = \{IV\}_k$, $K_i = \{K_{i-1}\}_k$

**Counter encryption** $K_i = \{IV + i\}_k$

- possible to parallelise for high-speed processing

**Cipher feedback** $C$ is encrypted with $K$ and XOR with plaintext

- recovers from transmission errors

**Message authentication code** (MAC) to verify integrity

- CBC mode, all but latest block discarded: keyed hash function

**Data Encryption Standard (DES)**

- Feistel cipher

- Developed by IBM in 1970s, modified by NSA, federal standard (FIPS-46) 1976

- Key length 56 bits
  $\Rightarrow$ too short nowadays

  - 1998 EFF “Deep Crack” (cost USD250,000) broke DES challenge in 56 hours with brute force

- Four weak and 16 semi-weak keys

- Still usable as Triple DES (3DES)

  - $C = DES_{K_3}(DES_{K_2}^{-1}(DES_{K_1}(M)))$
  - efficient key length 112 bits, while some advertise 168 bit key The second step could be also DES encryption, but on some hardware-based systems decrypting gives a better performance.

---

$^4$Can be embedded into message
Advanced Encryption Standard (AES)

- SP-network
- Subset of Rijndael (fixed block length 128 bits)
- Efficient also on small systems (smartcards etc.)
  - AES-128 about as fast as DES
  - 3 times faster than Triple DES
- Key length 128, 196 or 256, the shortest not for Top Secret in US
- FIPS-197

Stream ciphers

- Cryptographic secure pseudo-random number generator
- XOR by bit or by byte (synchronous stream cipher)
- Popular in communications
  - byte-sized: no need to pad blocks
  - simple implementation on hardware: for example A5/1 needs only three shift registers
    (19, 22, and 23 bits) and some XOR ports
  - fast: RC4 about 5 times faster than AES-128 on software
- Vulnerable to bit-fiddling: if one knows that an interesting bit at position $N$ should be
  inverted, one can just change it from the bit stream. On the other hand this provides some
  protection from transmission errors: with block ciphers one will end with a large block of
  invalid data.
- RC4 used in SSL, WEP
- A5/1 and A5/2 in GSM
- Both have security problems, A5/2 very weak

Asymmetric ciphers

- Symmetric ciphers provides secrecy only if one can communicate the key to other party
  secretly
  - key management becomes problem
- Use a problem that is
  - easy to construct
  - hard solve without
  - specific knowledge (= private key)
- NP-complete problems are good candidates. However, not every NP-complete problem is
  suitable for asymmetric cipher. For example, knapsack problems were thought to be good
  algorithms, but they have been broken.
- Can be used to provide a digital signature without third party

\[2\text{On modern 32-bit computer.}\]
RSA

- Factoring large numbers is hard
- Public key:
  - \( n = pq \), \( p \) and \( q \) large primes
  - \( e \) relatively prime for \((p-1)(q-1)\)
- Private key:
  - \( d = e^{-1} \mod ((p-1)(q-1)) \)
- Encrypting: \( c = m^e \mod n \)
- Decrypting: \( m = c^d \mod n \)

ElGamal

- Discrete logarithm in a finite field
- \( y = g^x \mod p \), prime \( p \), random numbers \( g < p, x < p \)
- Public key: \( y, g, \) and \( p \)
- Private key: \( x \)
- Signature: random \( k \) (relatively prime for \( p - 1 \), must be kept secret)
  - \( a = g^k \mod p \), solve \( b \) from \( M = (xa + kb) \mod (p-1) \)
  - verify: \( g^a b^k \mod = g^M \mod p \)
- Encrypting: \( a = g^k \mod p \), \( b = g^k M \mod p \)
- Decrypting: \( M = b/a^x \mod p \)

Message digest functions

- Calculating a signature for a long document
  - time-consuming
  - as large (or larger) than the original document
- Verifying document integrity
  - signed digest
  - digest stored or communicated securely. For example, there can be a list of hashes of all system files on read-only media. If any of those is modified, it may be detected by comparing hashes.
- Cryptographic checksum function
  1. \( \mathcal{H} = h(\mathcal{M}) \) easy to compute
  2. infeasible to find \( \mathcal{M} \) for given \( \mathcal{H} \)
  3. infeasible to find \( \mathcal{M}, \mathcal{M}' \) such that \( h(\mathcal{M}) = h(\mathcal{M}') \)
  4. an one-bit change in \( \mathcal{M} \) should result every bit in \( \mathcal{H} \) to change with \( P = \frac{1}{2} \)
- Birthday attack: \( 2^{\frac{n}{2}} \)

---

3Ron Rivest, Adi Shamir and Len Adleman
Secure hash algorithms in use

**MD5** designed in 1991
- 128-bit
- some weakness found, maybe insecure for demanding applications

**SHA-1** federal standard 180-2
- 160-bit
- original SHA (1993, SHA-0) vulnerable to $2^{39}$
- SHA-1 vulnerable to a collision at $2^{63}$
- longer versions (SHA-2) to 512 bits; not analysed in depth
- NIST soliciting new proposals to replace

**RIPEMD-160** European algorithm
- 160-bit, also longer ones

**HMAC: keyed hash**
- Used for authentication with shared secret
- $h(K \oplus \text{opad})||h(K \oplus \text{ipad}||M))$
  - $\text{ipad}$ and $\text{opad}$ select different bits from $K$
- Protects $K$ from eavesdropping

What cipher to choice
- How to distribute keys
- What trust model one has
- Any performance constrains
- Using public algorithms gives comfort, as if there is a weakness, it will be publicly known with good probability.
- Beware snake oil: unbreakable, certified, technobabble, secret, military grade, . . .

Failures on ciphers
- Even if you take a good algorithm, wrong use may result bad security
- A bad algorithm is always bad security
- *Do not modify* a cryptographic algorithm: adding rounds or increasing the key length may result in a weaker algorithm.
- Check that you use cryptographic as planned
  - stream ciphers: use different IV each time
  - MS Office uses same IV for all saves of same document

\[ \text{ipad}=0x36 \ldots, \text{opad}=0x5c \ldots \]
**WEP: Wired Equivalent Privacy**

- Use of encryption optional
  - ⇒ system administration failures
- No key management: use of shared key
- CRC-32 used for integrity check
  - linear algorithm: possible to fix changes with stream cipher bit-fiddling
- $IV$ only 24 bit
  - wraps around in a day (or faster: 5–10 GiB of typical network data)
- shared key ⇒ same $IV$ by multiple hosts
- birthday paradox results same $IV$ every few thousand packets

**Attacks on WEP**

- Statistical analysis for packets with the same $IV$
- Injecting known traffic e.g. from the Internet enables decrypting packets with the same $IV$
- Using replay attacks to generate traffic, for example ARP packets
- If the RC4 stream for one packet is known, it is possible to send encrypted packets with the same $IV$
- Bit-fiddling attacks to change the content or the destination of packets
- Bad software key generators: key space may be $2^{21}$, not $2^{64}$
- Dictionary attack on keys, like for passwords. You can make attack passively just by capturing a number of packets and trying different passphrases to find out the key.

**Key lengths: how long is safe**

- How long time the information must stay secret
- Longer key results in more computational load: limits available communication speed or increases energy consumption on mobile devices.

- **Symmetric ciphers**
  - risk: a fundamental weakness will be found or advances in computing
  - 64 bit cipher broken: RSA RC5 challenge
  - 128 bits should be OK
  - 196–256 bit AES key for Top Secret

- **Asymmetric ciphers**
  - risk: advances in mathematics or in computing
  - 576-bit key factored
  - RSA key lengths and same-level symmetric keys
    - prime bits symmetric
    - 1024 80
    - 2048 112
    - 3072 128
    - 15360 256
  - elliptic curves: double to symmetric keys

- **Message digests**
  - SHA-1 currently used, retired by 2010
  - SHA-2 algorithms unsure
Some performance figures for 1 KiB blocks

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>relative speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>DES CBC</td>
<td>1000</td>
</tr>
<tr>
<td>RC4</td>
<td>3638</td>
</tr>
<tr>
<td>AES 128</td>
<td>921</td>
</tr>
<tr>
<td>AES 196</td>
<td>796</td>
</tr>
<tr>
<td>AES 256</td>
<td>705</td>
</tr>
<tr>
<td>RSA 1024 sign</td>
<td>7 / 1000</td>
</tr>
<tr>
<td>RSA 1024 verify</td>
<td>132 / 1000</td>
</tr>
<tr>
<td>RSA 4096 sign</td>
<td>0.2 / 1000</td>
</tr>
<tr>
<td>RSA 4096 verify</td>
<td>12 / 1000</td>
</tr>
<tr>
<td>MD5</td>
<td>4992</td>
</tr>
<tr>
<td>SHA-1</td>
<td>3360</td>
</tr>
</tbody>
</table>

Summary

- Bits do not matter (much)
- Important
  - to select the right algorithm for the right use
  - to use algorithm in the right way
- Hardware used may impose some limitations
- For many uses, the performance is not the real problem

References

