Cryptography II: Symmetric Ciphers

David Aspinall
School of Informatics,
University of Edinburgh.

25th February 2003

http://www.informatics.ed.ac.uk/teaching/modules/cs
Contents

- Notions of security
- Cryptanalysis techniques
- Stream and block ciphers
- Symmetric algorithms
- Construction methods, modes
- DES and Rijndael.

"Cryptography is a mixture of mathematics and muddle, and without the muddle the mathematics can be used against you." — Ian Cassells, former Bletchly Park cryptanalyst.
Notions of security

Security of crypto primitives is judged according to various notions:

1. **Unconditional (or perfect) security.** Information-theoretic: ciphertext contributes no information about the plaintext. Even if an adversary has unlimited computational resources, he cannot break the system. Best guarantee, but the key must be as long as the plaintext.

2. **Complexity-theoretic security.** Assume adversary has polynomially-bounded computational power. Prove that wrt security parameters, problem of breaking system is intractable. Strong guarantee but tricky.

3. **Provable security.** Breaking system is proved as difficult as some well-known *supposedly* difficult problem (e.g., FACTORING or DLP).

4. **Computational security.** Make specific assumptions about the resources and methods available to an adversary, and show that they are not enough to break the system. May just consider best known method for attack (without proof), or may apply 2 or 3 above, choosing security parameters to go comfortably beyond the adversary’s abilities.
Cryptanalysis attacks [Sch96]

Typical attack is based on $C_1 = E_e(P_1)$, $C_2 = E_e(P_2)$, \ldots, $C_n = E_e(P_n)$ for some small $n$. The best outcome is to find $e$ or an algorithm for $E_e$.

<table>
<thead>
<tr>
<th>Attack style</th>
<th>Given information</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ciphertext-only attack</strong></td>
<td>the $C_i$ (deduce at least $P_i$)</td>
</tr>
<tr>
<td><strong>Known-plaintext attack</strong></td>
<td>the $C_i$ and $P_i$</td>
</tr>
<tr>
<td><strong>Chosen-plaintext attack</strong></td>
<td>$C_i$ for chosen $P_i$</td>
</tr>
<tr>
<td><strong>Adaptive-chosen-plaintext</strong></td>
<td>as above, but iterative</td>
</tr>
<tr>
<td><strong>Chosen-key attack</strong></td>
<td>mathematical key relationships (obscure)</td>
</tr>
<tr>
<td><strong>Chosen-ciphertext attack</strong></td>
<td>$C_i, P_i = D_d(C_i)$. Find decryption key $d$.</td>
</tr>
<tr>
<td>“Rubber-hose cryptanalysis”</td>
<td>Bribery, torture, or blackmail.</td>
</tr>
<tr>
<td>(not really cryptanalysis, but v successful)</td>
<td></td>
</tr>
<tr>
<td>“Purchase-key attack”</td>
<td>Bribery again.</td>
</tr>
</tbody>
</table>
Linear Cryptanalysis

- In **linear cryptanalysis** [Mat94], the analyst searches for a linear relationship across the cipher, relating input bits and output bits. This can approximate linear aspects of the cipher's construction.

- The technique is to glue together a number of probabilistic relations garnered by experimentation, e.g.,

  \[ o_1 + o_8 = i_7 + i_3 \quad \text{with probability 0.7} \]

  Internal knowledge of the way the cipher is constructed helps find and combine these relationships.

- Suppose the final relationship has probability \( p = 0.5 + \frac{1}{m} \) for some \( m \). Then we expect to start recovering key bits with \( m^2 \) known texts.

- If \( m^2 \) is greater than the number of possible known texts (e.g., \( 2^n \) for \( n \)-bit input or block size), then the cipher is considered secure against linear cryptanalysis.
Differential Cryptanalysis

- In differential cryptanalysis [BS93], the technique is to observe changes in the output based on changes in the input. This can exploit non-linear aspects of the cipher.

- Observations are based on cipher text pairs, and have the form:

  flipping bits $i_2, i_3, i_7 \Rightarrow$ bits $o_0$ and $o_8$ flip \quad with probability 0.7

- We examine all possible input pattern changes, noting those having particularly high or low probabilities. Then we try to combine the informative observations, leading to a output-difference that holds for the whole cipher with a useful probability. With enough chosen texts, we can see the expected output and make deductions about the key.

- Again, the cipher is considered secure against this kind of attack if the number of texts needed is greater than the total possible number of texts for the key.
Symmetric-key encryption schemes are often characterised as *stream ciphers* or *block ciphers* although the distinction can be fuzzy.

- **A stream cipher** is an encryption scheme which treats the plaintext symbol-by-symbol (e.g., by bit or byte). The security usually lies in a changing **keystream** rather than the encryption function, which may be simple. Typically, \( M = C = A \) and a stream of symbols \([m_1 \ m_2 \ m_3 \ \cdots]\) is encrypted using a keystream \([e_1 \ e_2 \ e_3 \ \cdots]\) to generate \([E_{e_1} (m_1) \ E_{e_2} (m_2) \ E_{e_3} (m_3) \ \cdots]\).

  Stream ciphers may be **synchronous** (keystream generated independently of the plaintext and ciphertext) or **self-synchronizing** (the keystream is generated as a function of the key and a fixed amount of previous ciphertext).

- **A block cipher** is an encryption scheme which breaks up the plaintext message into **blocks** of a fixed length (e.g., 128 bits), and encrypts one block at a time, using a complex encryption function.
The **Vernam cipher** is a stream cipher defined on the alphabet $\mathcal{A} = \{0, 1\}$, which uses a key stream also made of binary digits. Each symbol $m_i$ in the message is encoded using the corresponding symbol $k_i$ of the key stream, using exclusive-or:

$$c_i = m_i \oplus k_i.$$ 

Because $(a \oplus b) \oplus b = a$, the decryption operation is identical:

$$m_i = c_i \oplus k_i.$$ 

If the key string is randomly chosen, and never reused, then this cipher is called a **one-time pad**. Claude Shannon proved that this cipher is unconditionally secure. Unfortunately, to guarantee this, it requires a true random source for key bits (hard to come by), and a key stream as long as the message. This makes it impractical for most applications. It used to be used for high security communications between Washington and Moscow.
Feedback Shift Registers

- More practical than the one-time pad would be to use a pseudorandom keystream, which is seeded with a much shorter key. Feedback shift registers (FSRs) are the basic component of many keystream generators, used to produce pseudorandom bit streams.

- An FSR of length $n$ consists of $n$ 1-bit register stages connected together, whose contents $\vec{s}$ are inputs to a boolean function $f$. At each tick, the contents are shifted right, and $f$ calculates the feedback digit.

- If the initial state is $[s_{n-1}, \ldots, s_0]$, then the output sequence $s_0, s_1, \ldots$ is determined by the equation: $s_j = f(s_{j-1}, s_{j-2}, \ldots s_{j-n})$ for $j \geq n$. 

![Feedback Shift Register Diagram]

\[ f(s_{j-1}, s_{j-2}, \ldots s_{j-n}) \]
Linear Feedback Shift Registers

• In a LFSR of length $n$, the feedback function $f$ is determined by a $n$-degree connection polynomial $C$ with coefficients $c_i$

$$C(X) = c_nX^n + \ldots + c_2X^2 + c_1X + 1$$

this determines the feedback function, as:

$$s_j = (c_1s_{j-1} + c_2s_{j-2} + \cdots + c_n s_{j-n}) \mod 2 \quad \text{for } j \geq n.$$  

• LFSRs have an elegant algebraic theory and can be constructed to produce sequences with good properties: a large period (the maximum, $2^n - 1$), good statistical randomness properties, and a large linear complexity (a statistical “effectiveness” measure).

• Unfortunately, LFSRs are insecure. They can be broken by determining the length $n$, and then using a known-plaintext attack of length $2n$.

• In practice, some controlled non-linearity is added by either non-linear filtering or composition of LFSRs, or LFSR-controlled clocking.
A simple substitution cipher is a block cipher for arbitrary block length $t$. It swaps each letter for another letter, using a permutation of the alphabet.

- Let $\mathcal{A}$ be an alphabet, $\mathcal{M}$ be the set of strings over $\mathcal{A}$ of length $t$, and $\mathcal{K}$ be the set of all permutations on $\mathcal{A}$.
- For each $e \in \mathcal{K}$ define $E_e$ by applying the permutation $e$ to each letter in the plaintext block:

$$E_e(m) = e(m_1)e(m_2) \cdots e(m_t) = c$$

where $m \in \mathcal{M}$ and $m = m_1m_2 \cdots m_t$.

For each $d \in \mathcal{K}$ we define $E_d$ in exactly the same way,

$$D_d(c) = d(c_1)d(c_2) \cdots d(c_t).$$

- Key pairs are permutations and their inverses, so $d = e^{-1}$, and

$$D_d(c) = e^{-1}(c_1)e^{-1}(c_2) \cdots e^{-1}(c_t) = m_1m_2 \cdots m_t = m.$$
• The **Caesar cipher** is a simple substitution cipher which replaces
  A → D, B → E, C → F, ... , X → A, Y → B, Z → C.

• The **ROT-13** transformation provided in Usenet news readers is similar,
  but replaces each letter character $c$ with the character $(c + 13) \mod 26$.
  This permutation is its own inverse.

Simple substitution ciphers are **insecure**, even when the key space is large.
The reason is that the distribution of letter frequencies is preserved in the
ciphertext, which allows easy cryptanalysis with a fairly small amount of
ciphertext and known properties of plain text (for example, the relative
frequencies of letters in English text).

This emphasises what should be an obvious point: a **large keyspace does not guarantee a strong cipher**. For the alphabet A-Z, the size of the key
space for this cipher is $26! \approx 2^{88}$, large enough to prevent brute force
attacks by todays standards. But the cipher is easy to break.
A *polyalphabetic substitution cipher* is a block cipher with block length $t$. Instead of a single permutation, it uses a set of $t$ permutations, and substitutes each letter using a permutation corresponding to its position in the block.

- Let $\mathcal{A}$ and $\mathcal{M}$ be as before. Let $\mathcal{K}$ be the set of all $t$-tuples $(p_1, \ldots, p_t)$ where each $p_i$ is a permutation on $\mathcal{A}$.

- For each $e = (p_1, \ldots, p_n) \in \mathcal{K}$ define $E_e$ by applying the permutation $p_i$ to the $i$th letter in the plaintext block:

  $$E_e(m) = p_1(m_1)p_2(m_2)\cdots p_t(m_t) = c$$

  where $m = m_1m_2\cdots m_t$.

- The corresponding decryption key is $d = (p_1^{-1}, \ldots, p_t^{-1})$. 
The **Vigenère cipher** has a block-length of 3, and uses the permutations $e = (p_1, p_2, p_3)$ where $p_1$ rotates each letter of the alphabet three places to the right, $p_2$ rotates seven positions, and $p_3$ ten positions ($e$ may be represented as the word DFI). For example:

$m = \text{COM EON EVE RYB ODY}$

$E_e(m) = \text{FVW HVX HCO UFL RKI}$

Polyalphabetic substitution ciphers have the advantage over simple substitution ciphers that symbol frequencies are not preserved: a single letter may be encrypted to several different letters, in different positions. However, cryptanalysis is still straightforward, by first determining the block size, and then applying frequency analysis by splitting the letters into groups which are encrypted with the same permutation. So polyalphabetic substitutions are certainly **not secure**.
Simple transposition ciphers

The simple transposition cipher is a block cipher with block-length $t$. The idea is simply to permute the symbols in the block.

- Let $\mathcal{K}$ be the set of all permutations on the set \{1, 2, \ldots, t\}. For each $e \in \mathcal{K}$, the encryption function is defined by

$$E_e(m) = (m_{e(1)}, m_{e(2)}, \ldots, m_{e(t)}).$$

The corresponding decryption key is the inverse permutation.

This cipher again preserves letter frequencies, which allows easy cryptanalysis. So it is not secure.

These block ciphers so far are not useful by themselves, but get interesting when combined. A good cipher should add both confusion by substitution transformations and diffusion by transpositions. Confusion obscures the relationship between the key and the ciphertext. Diffusion spreads out redundancy in the plaintext across the ciphertext. Modern block ciphers apply rounds consisting of substitution and transposition steps.
Product ciphers

It is easy to combine encryption functions using function composition, because the composition of two bijections is again a bijection.

- A *product cipher* is defined as the composition of $N$ encryption transformations, $E_1^e, E_2^e, \ldots, E_N^e$, for $n \geq 0$. (We can consider a single key space wlog: each transformation may depend on a different part of the key $e$, or may be independent of the key.)

- The overall encryption function is given by composing the parts:

  $$E_e = E_1^e \circ E_2^e \circ \cdots \circ E_N^e$$

  where $\circ$ denotes function composition in the diagramatic order.

- The overall decryption function is the composition of the corresponding component decryptions: $D_1^d = D_N^d \circ \cdots \circ D_2^d \circ D_1^d$.

- Involution (functions that are their own inverse) are particularly useful in constructing product ciphers. The favourite is XOR: $f(x) = x \oplus c$. 

– 16 –
Constructing block ciphers

- Classically, block ciphers are conceptualized as being constructed from circuits of **S-boxes** and **P-boxes**, which perform *substitutions* and *permutations* of the cipher text. (An $X \times Y$ box maps $X$ bits to $Y$ bits).

- The **Feistel principle** gives a way of constructing a cipher so that the same circuit is used for both encryption and decryption. A *round* in a Feistel cipher treats the input block in two halves, $L_i$ and $R_i$. It uses the right-hand half to modify the left, and then swaps:

$$L_{i+1} = R_i \quad R_{i+1} = L_i \oplus F(K_i, R_i).$$

The inverse operation is:

$$R_i = L_{i+1} \quad L_i = R_{i+1} \oplus F(K_i, L_{i+1}).$$

- Block ciphers can be used in various **modes**, including ECB, CBC, CFB, OFB. These are described on the next slide.

**Important reading exercise**: compare the security, efficiency, inbuilt data integrity, and error recovery of these different modes.
Modes for block ciphers

**ECB** *electronic codebook mode.* Each block of plaintext $P_i$ is enciphered independently under the same key. (Obvious mode; obvious failings).

**CBC** *cipherblock chaining mode.* Each plaintext block $P_i$ is XORed with the previous ciphertext $C_{i-1}$ block before encryption. An *initialization vector* (IV) (optionally secret, fresh for each message) is used for $C_0$.

\[
C_i = E_k(P_i \oplus C_{i-1}) \quad P_i = C_{i-1} \oplus D_k(C_i)
\]

**OFB** *output-feedback mode.* Encryption function of block cipher used as a synchronous stream cipher (*internal feedback*).

\[
C_i = P_i \oplus S_i; \quad S_i = E_k(S_{i-1}) \quad P_i = C_i \oplus S_i; \quad S_i = E_k(S_{i-1})
\]

**CFB** *cipher-feedback mode.* Encryption function of block cipher used as self-synchronizing stream cipher for symbols of size up to block size.

\[
C_i = P_i \oplus E_k(C_{i-1}) \quad P_i = C_i \oplus E_k(C_{i-1})
\]
**DES**

- **DES** is a block cipher based on Feistel's principle. Block-size is 64 bits, key-size 56 bits (+8 parity bits). Invented by IBM in 1970s, tweaked by NSA. Still widely used, esp. in financial sector. Much analysed.

- An outline of DES is given on the next slide. Tables are needed to fully specify the permutations and S-boxes. See a textbook or the standard.

- Main threat isn't cryptanalytic, but (slightly optimised) **exhaustive search** in small key-space. Remedied by **3DES** (triple DES), 3 keys:

  \[ C = E_{k_3}(D_{k_2}(E_{k_1}(P))) \quad P = D_{k_3}(E_{k_2}(D_{k_1}(C))). \]

  Security of 3DES is not obvious: repeated encryption may not gain security (one-step DES is not closed, so it in fact does), and new attacks may be possible (**meet-in-the-middle attack**). With 3 independently chosen keys, security is roughly the same as expected with 2 keys.

- Several other DES variants, including **DESX**, using **whitening** keys \( k_1, k_2 \) as \( C = E_k(P \oplus k_1) \oplus k_2 \). (Used in Win2K encrypting FS).
Outline of DES internals

- DES is constructed as a 16-round Feistel cipher:
  1. 64-bit input block has initial permutation $P_0$ and partition to $L_0, R_0$;
  2. 16 Feistel rounds, each using 48-bit subkey $K_i$ to operate on the right-half of a block: $R_{i+1} = L_i \oplus F(K_i, R_i)$, then swapping $R_i, L_i$;
  3. After final round, inverse of initial permutation, $P_0^{-1}$ is applied.

- Each $K_i$ is computed by the key scheduler:
  1. start from a permutation $P_k(K)$ of the original key $K$;
  2. at each step, apply circular shifts to two 28-bit key-halves, and
  3. apply a permutation $P_s(K)$ which selects 48-bits from 56 to give $K_i$.

- The cipher function $F$ adds non-linearity; it has four layers:
  1. The 32-bit block half $R$ is expanded to 48-bits $E(R)$;
  2. The key is added, producing $E(R) \oplus K$;
  3. Eight groups of 6-bits $E(R) \oplus K = M_1 \cdots M_8$ are fed into eight $6 \times 4$ S-boxes to produce eight 4-bit outputs $S_1(M_1) \cdots S_8(M_8)$;
  4. The final result is a permutation of $S_1(M_1) \cdots S_8(M_8)$. 

- 20 – 
• In October 2000, the US NIST selected **Rijndael** as the new **AES**, to replace the aging DES. Rijndael was designed by two Belgian cryptographers, Vincent Rijmen and Joan Daemen. The algorithm was selected as a result of a 3 year worldwide review process. No proof of security, but a high level of confidence amongst cryptographers.

• Rijndael satisfied a number of requisite criteria for the AES:
  — **Security**: math basis, cryptanalytic resistance; output randomness;
  — **Efficiency**: in time/space, hardware and software implementations;
  — **Flexibility**: block sizes 128 bits, key sizes 128/192/256 bits.
  — **Intellectual property**: unclassified, published, royalty-free.

The US Federal Information Processing standard **FIPS 197** for AES was published in November 2001. Rijndael will probably be very widely used worldwide in the future.

• Rijndael is built as a network of linear transformations and substitutions, with 10, 12 or 14 rounds, depending on the key size.
Outline of Rijndael internals

A 128-bit block is split into an array of 16 bytes, sometimes considered as a $4 \times 4$ square. The round function processes this state in four layers:

1. ByteSub. A single $8 \times 8$ S-box is applied to each byte. The S-box is defined by the transformation $S(x) = M(1/x) + b$ in a finite field for a matrix $M$ and constant $b$; in practice, a 256-byte lookup table is used.
2. ShiftRow. Rows 0-3 of the square are circular left-shifted by 0-3 places.
3. MixColumn. Each column is mixed by a matrix multiplication.
4. AddRoundKey. The round key is XORed into the bytes.

The key scheduler expands $K$ to enough bits (1408) for the round:

- KeyExpansion. Uses the S-box, XOR, and a round-dependent constant.

The overall algorithm applies an initial key addition, followed by 10 rounds. The final round does not use MixColumn. Each step is invertible, so decryption is composition of inverse transformations. The design cleverly allows the same functions to be used with modified parameters.
Current symmetric crypto algorithms

Stream ciphers

- **A5**, encrypts GSM digital cellular traffic. Originally secret, but leaked and reverse-engineered. Based on three LFSRs. Very feasible attack.
- **PKZIP** has a byte-wide stream cipher. Easily broken with a small amount of plain text.
- **RC4/ARCFOUR**. RSADSI trade secret; code posted anonymously in 1994. Variable key-size, byte-wide, OFB with $8 \times 8$ S-box. Very fast & simple, widely licensed (Lotus Notes, Oracle SQL), less widely studied.

Block ciphers

- **DES, 3DES, Rijndael** outlined previously.
- **IDEA**, 64-bit blocks, 128-bit key. Efficient: uses XOR, addition and multiplication operations. Patented for commercial use. Used in PGP.
- **Skipjack**. NSA designed, once classified (key escrow and LEAF issue) and patented under a secrecy order; now public domain. Block size 64 bits, 80-bit key. Used in tamperproof **Clipper** and **Capstone** chips.
References


