

COMP3441/9441 Week2

Transposition

Transposition ciphers work by shuffling the plaintext in some way.

t r a n s p o s
i t i o n c i p
h e r s w o r k
b y s h u f f l
i n g t h e p l
a i n t e x t i
n s o m e w a y

Ciphertext: tihbian rteynis airsgns noshttm snwruhee pcofexw splkliy

Block Ciphers

Suppose the plaintext M is divided into blocks

$$M = M_1M_2\dots$$

(typically of equal length)

A *block cipher* encrypts the blocks separately:

$$E(M, K) = E(M_1, K)E(M_2, K), \dots$$

A problem with this is that if there is a repeated plaintext block $M_i = M_j$ then there is a repeated ciphertext block $E(M_i, K) = E(M_j, K)$. This gives a handle for cryptanalysis.

Cipher Block Chaining

Here the plain-text blocks $P_1P_2P_3\dots P_n$ are mapped to the ciphertext blocks $C_1C_2C_3\dots C_n$ by the equations

$$C_1 = E(P_1, K)$$

$$C_{i+1} = E(P_{i+1} \otimes C_i, K)$$

This makes each block of cipher text depend on *all* the preceding blocks of plaintext, and eliminates repeated blocks.

DES: Data Encryption Standard

Initiated in 1972 by a US standards bureau NSB (now called National Institute of Standards) to develop a standard for commercial encryption

Designed by researchers at IBM, with (at the request of NSB) input from the National Security Agency (NSA)

Adopted as a standard 1976, to be reviewed every five years,

Last recertified Nov, 1999, in “triple-DES” format

Extensively used in financial industry

A block cipher, with block size 64 bits,

key size 56 bits, so about 70,000,000,000,000 keys

uses simple operations, efficiently implementable in hardware

design objective: every bit of the ciphertext should depend on every bit of the key and every bit of the plaintext, in minimal computation time.

mixes transposition and substitution methods

can be used in a variety of modes, including cipher block chaining

DES Conspiracy theories

The NSA

- argued to have the key length reduced from 128 bits to 56 bits
- proposed changes to the s-boxes

Why?

- Did they put in a trapdoor?
- Did they pick the key length to be such that they could crack it given their computation resources?

Contra Conspiracy

NSA thought DES would be released in hardware implementations only; not published

NSA has admitted DES was its “biggest mistake”

DES is secure against “differential cryptanalysis” a type of chosen plaintext cryptanalysis first published in 1990

(theoretical attack, data construction requires encrypting a 1.5 mega-bit stream of chosen plaintext for 3 years.)

Designers knew about differential analysis, and designed against it

But is it secure?

Computer processor power has advanced exponentially since 1983, making brute force attacks feasible.

Cracked by the Electronic Frontier Foundation using a specially constructed \$US 250,000 machine in 56 hours (beating record of 39 days) using a brute force search (88 billion keys per second).

Strengthening DES

current DES standard recommends triple-DES: encrypt/decrypt 3 times using 3 different 56 bit keys:

$$DES_3(K_1 \cdot K_2 \cdot K_3, M) = E_{K_3}(D_{K_2}(E_{K_1}(M)))$$

Is this better than DES? Or is there a key K such that $DES_3(K_1 \cdot K_2 \cdot K_3, M) = E_K(M)$?

No, but it took a while to prove!

K.W. Campbell and M.J. Wiener, DES is not a group, *Advances in Cryptology - Crypto '92*, Springer-Verlag (1993), 512-520.

Advanced Encryption Standard

Advanced Encryption Standard development process initiated with call for proposals in 1997.

5 finalists announced August 1999

Rijndael algorithm adopted as AES in Nov 2001

Key Distribution

The problem: Alice and Bob, who have never met, wish to communicate over an insecure line. Eve, an eavesdropper, is listening in on the line. To use shared key cryptography, they need to establish a shared key. How can they do this?

A non-solution:

A \longrightarrow B: K

A \longrightarrow B: $E_K(M)$

B knows $K, E_K(M)$, computes $D_K(E_K(M)) = M$

Problem: E also knows $K, E_K(M)$, so can compute K

An idea suggesting the problem is solvable

Suppose that $E_{K_A}(E_{K_B}(M)) = E_{K_B}(E_{K_A}(M))$

A \longrightarrow B: $E_{K_A}(M)$

B \longrightarrow A: $E_{K_B}(E_{K_A}(M))$

A knows $K_A, E_{K_B}(E_{K_A}(M))$, computes

$$D_{K_A}(E_{K_B}(E_{K_A}(M))) = D_{K_A}(E_{K_A}(E_{K_B}(M))) \\ E_{K_B}(M))$$

A \longrightarrow B: $E_{K_B}(M)$

B knows $K_B, E_{K_B}(M)$, computes $D_{K_B}(E_{K_B}(M)) = M$

Eve knows $E_{K_A}(M), E_{K_B}(E_{K_A}(M)), E_{K_B}(M)$ and should not be able to compute M from this.

Problem: It is hard to find a good cipher E such that $E_{K_A}(E_{K_B}(M)) = E_{K_B}(E_{K_A}(M))$

(Implementable if A and B are communicating by snail mail and each has a padlock.)

Modular Arithmetic

Let x, y, n be integers, $n > 0$. $x \equiv y \pmod n$ if there exists an integer k such that $x = k \cdot n + y$.

For every integer x , there is an integer $y \in \{0, 1, \dots, n - 1\}$ such that $x \equiv y \pmod n$. Write this y as $x \pmod n$.

If $x \equiv x' \pmod n, y \equiv y' \pmod n$ then

- $x + y \equiv x' + y' \pmod n$
- $x \cdot y \equiv x' \cdot y' \pmod n$
- $x^y \equiv x'^y \pmod n$

NB: it does not follow that $x^y \equiv x^{y'} \pmod n$.

Example: $x = 2, y = 1, y' = 4, n = 3$

n is *prime* if it has no divisors other than 1 and n .
 a is a *generator* modulo n if $a, a^2, a^3, \dots, a^{n-1} \pmod n$ includes all numbers $1, 2, 3, \dots, n - 1 \pmod n$

Fact: If q is prime then generators mod q exist.

Example: 2 and 3 are generators mod 5 (and the only ones)

Diffie-Hellman-Merkle's solution

Let q be a prime and $2 \leq a \leq q - 1$

- A generates a number X_A , B generates X_B
- A \longrightarrow B: $a^{X_A} \pmod q$
- B \longrightarrow A: $a^{X_B} \pmod q$

A knows X_A and $a^{X_B} \pmod q$ and computes

$$(a^{X_B} \pmod q)^{X_A} \pmod q = a^{X_A \cdot X_B} \pmod q$$

B knows X_B and $a^{X_A} \pmod q$ and computes

$$(a^{X_A} \pmod q)^{X_B} \pmod q = a^{X_A \cdot X_B} \pmod q$$

shared key is $a^{X_A \cdot X_B} \pmod q$

E knows $a^{X_A} \pmod q$ and $a^{X_B} \pmod q$.

No efficient way to compute $a^{X_A \cdot X_B} \pmod q$ from these is known.

In particular, no efficient way is known to compute the *discrete*

logarithm: i.e., given a, b, q , solve $a^x \equiv b \pmod q$ for x .

Note: to increase Eye's set of possible candidates for $a^{X_A \cdot X_B}$, want $a^x \pmod q$ to have a large set of possible values as x varies.

So, choose a to be a generator mod q , giving $q - 1$ different values.