Computer Science 418

Product Ciphers, Block Ciphers, DES

Mike Jacobson

Department of Computer Science
University of Calgary

Week 4

Outline

Product Ciphers

Shannon also introduced the idea of product ciphers (multiple encryption):

Definition 1 (Product cipher)

The *product* of two ciphers is the result of applying one cipher followed by the other.

AKA *superencipherment* and various other names.

Properties of Product Ciphers

If different ciphers are used in a product cipher, ciphertexts of one cipher need to have the correct format to be plaintexts for the next cipher to be applied.

- This is composition of encryption maps.

Applying a product cipher potentially increases security. E.g. n-fold encryption with one cipher and n keys potentially corresponds to a cipher that has n times longer keys.

Of course it also results in a loss of speed by a factor of n, but this might be worth it for added security.

Caveat

Be careful with this reasoning!

Note 1

The product of two substitution ciphers is a substitution cipher. The product of two transposition ciphers is a transposition cipher.

Such ciphers are closed under encryption, so multiple encryption under different keys provides no extra security:

E.g. double encryption $C = E_{K_1}(E_{K_2}(M)) = E_{K_3}(M)$ for a third key K_3

Confusion and Diffusion

Shannon suggested applying two simple ciphers with a fixed mixing transformation (transposition) in between to:

- *diffuse* language redundancy into long term statistics, and to
- *confuse* the cryptanalyst by making relation between redundancy of the ciphertext C and the description of the key K very complex.

Definition 2 (Confusion)

Make the relationship between the key and ciphertext as complex as possible (accomplished by applying substitutions or *S-boxes*).

Definition 3 (Diffusion)

Dissipate the statistical properties of the plaintext across the ciphertext (accomplished by applying applying transpositions or *P-boxes*).

Examples of Product Ciphers

• ADFGX/ADFGVX Ciphers (employed by the Germans in WW I).

• Hayhanen Cipher — Hayanan was a Russian spy caught in New York in the 1950's. The FBI couldn't break it!

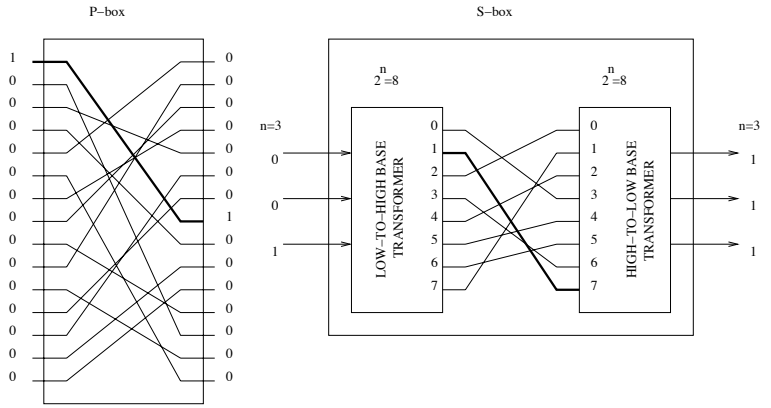
Example 4

IBM's Lucifer system, uses permutations (transpositions) on large blocks for the mixing transformation, and substitution on small blocks for confusion.

Feistel originally wanted to call the product cipher "Dataseal". IBM instead shortened the term *demonstration cipher* to "Demon." Later, it was changed to *Lucifer*, because it retained the "evil atmosphere" of Demon, and contained the word *cipher*.

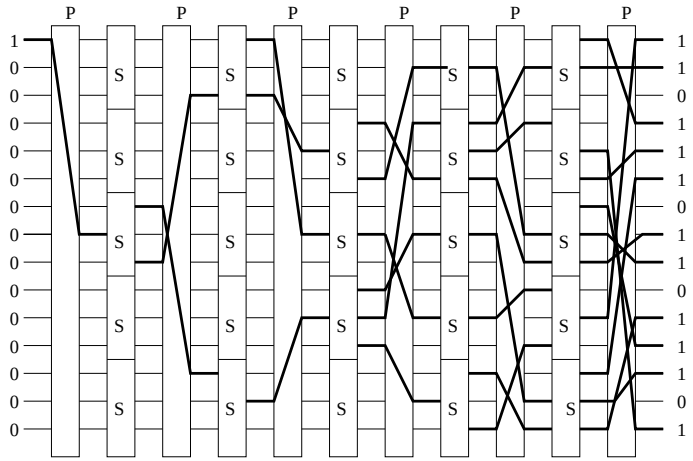
Lucifer: P-boxes and S-boxes

Since Lucifer was set up in hardware, they called the chips which did the permutation "P-boxes" and those that did the substitution "S-boxes."



Lucifer: Complete Cipher

The Lucifer system simply consisted of a number of P and S boxes in alternation.



Error Propagation

Lucifer demonstrates graphically not only the difffusion properties of Lucifer, but also the idea of *error propagation*.

Definition 5 (Error Propagation)

The degree to which a change in the input leads to changes in the output.

Good error propagation is a desirable property of a cryptosystem (a user can easily tell if a message has been modified).

Not good for decryption though (one error in the process should still mostly decrypt correctly).

Error Propagation in Lucifer

Example 6

Lucifer has good diffusion and error propagation. The thicker lines in the graphic indicate '1' bits. The '1' input bit dissipates over the entire ciphertext.

Aso, if the first '1' bit is changed to a '0', then half the bits in the output are affected.

Note: All modern symmetric key ciphers in use are product ciphers.

Block Ciphers

All modern ciphers in use are block ciphers (although not necessarily used as such — we'll talk about modes of operation of block ciphers later).

Definition 7 (Block cipher)

Encrypts plaintext blocks of some fixed length to ciphertext blocks of some fixed (possibly different) length.

Usually, a message M will be larger than the plaintext block length, and must hence be divided into a series of sequential message blocks $M_1, M_2, \ldots, M_n$ of the desired length.

- A block cipher operates on these blocks one at a time.

Examples of Block Ciphers

Example 8

The shift cipher is a block cipher where the blocks consists of one character (*i.e.* 8 bits on 32-bit architecture, 16 bits on 64-bit architecture).

Two main block ciphers in use today:

1. DES (obsolete, so now used in triple encipherment as 3DES)
2. AES

NIST Publications

- NIST: National Institute of Standards and Technology (formerly National Bureau of Standards (NBS))
- FIPS: Federal Information Processing Register

Everything about NIST's cryptographic standards, recommendations, and guidance can be found at the NIST Cryptographic Toolkit website `http://csrc.nist.gov/CryptoToolkit`.

- Extremely useful website for the practicing cryptographer.
- There is a link on the "external links" page on the course website.

DES Documents

- Original DES publication: FIPS 46 (January 15, 1977)
- Second DES publication: FIPS 46-2 (December 30, 1993)
- Current DES publication: FIPS 46-3 (October 25, 1999)

History of DES

- DES was developed by IBM around 1972.
- NBS made solicitations to IBM in 1973/1974 concerning the use of this as a public standard, in response to corporate needs for securing information.
- IBM and the National Security Agency (NSA) secretly evaluated DES for security.
- DES was approved in 1978, and it is still in use, although no longer endorsed.
- The *Advanced Encryption Standard* (AES), approved in October 2000, has now replaced it.

Specifications

• DES is a block cipher that encrypts 64-bit plaintext blocks to 64-bit ciphertext blocks using using 64-bit keys.

• Note that 8 of the key bits are parity bits, resulting in 56 actual bits of the key.

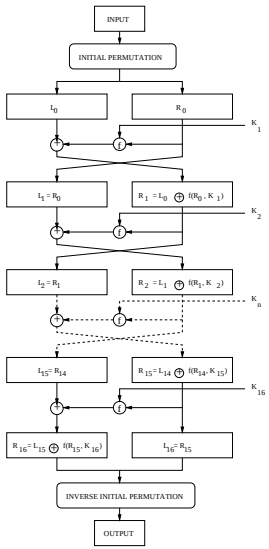
• So $\mathcal{M} = \mathcal{C} = \{0,1\}^{64}$ and $\mathcal{K} = \{0,1,\}^{56}$.

Overview

$DES_{key}(M) = IP^{-1}(S_{16}(S_{15}(\ldots(S_2(S_1(IP(M))))\ldots)))$

1. The 64 plaintext bits are permuted in a fixed order (transposition cipher).
2. The block is divided into two 32-bit words L_0 and R_0.
3. The block undergoes 16 substitution "rounds." In each round, one word is transformed using *XOR* and a substitution function, after which the two words are swapped.
4. In the last round, the two words are not swapped.
5. The original permutation is reversed.

Diagram of DES



Initial Permutation *IP*

See FIPS publication for precise specification.

Notation: first bit of the output is the 58th bit of the input.

Substitution Rounds

In round i :

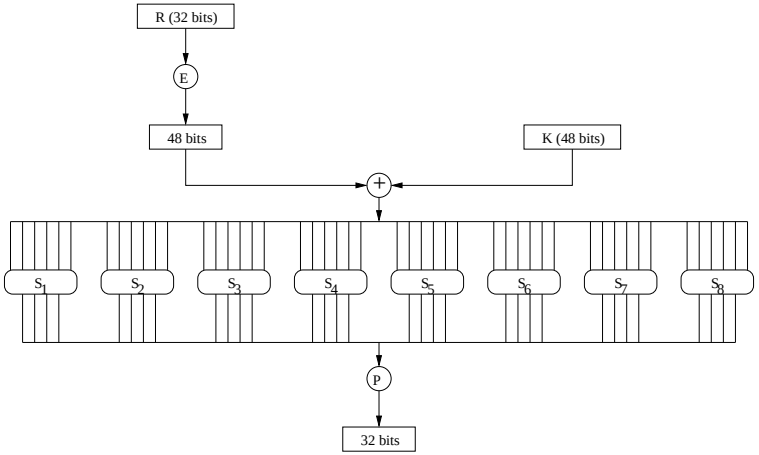
- The right word R_{i-1} is combined with the ith subkey K_i via the function f.
- The output of f is XORed with L_{i-1} to form R_i.
- The next left word L_i is the previous right word ($L_i = R_{i-1}$).

The Function f

f accepts as input R_i (32 bits) and the ith round subkey K_i (48 bits). The subkey generation is described below. The function f works as follows:

1. R_i is expanded to the 48 bit R_i' via the expansion function E, which simply repeats some of the bits of R_i in generating R_i' (see the FIPS publication for the specification of E).
2. R_i' is XORed with K_i (both are 48 bits long).
3. $R_i' \oplus K_i$ is broken into 8 6-bit words. Each of these words is replaced by a 4-bit word according to the 8 (different) S-boxes $S_1, S_2, \ldots, S_8$. The result of applying the S-boxes is a 32-bit string.
4. The 32-bit string is permuted according to the fixed permutation P (see the FIPS publication).

Diagram of f



Key Schedule

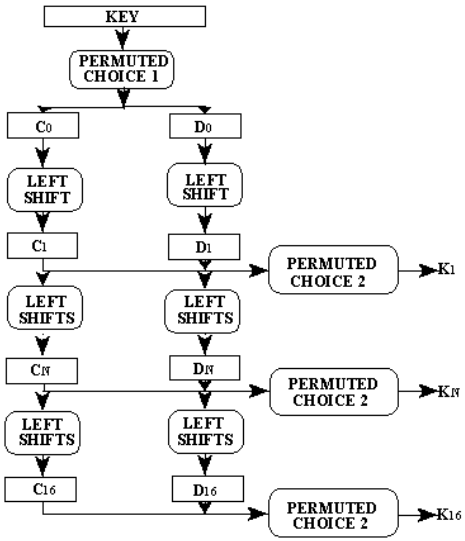
Generation of the Subkeys K_i :

1. The key K (56 bits) is permuted according to the fixed permutation "PERMUTED CHOICE 1" (see FIPS publication) and separated into two 28-bit words C_0 and D_0.
2. Each word is rotated either one or two places to the left according to the fixed schedule below, yielding C_1 and D_1.

Iteration	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
# of left shifts	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

3. K_1 is obtained from C_1 and D_1 via "PERMUTED CHOICE 2," which selects 48 bits from C_1 and D_1 according to a fixed ordering.
4. Steps 2 and 3 are repeated with C_i and D_i to obtain the remaining 15 subkeys.

Diagram of the Key Schedule



Decryption

Let $K_n = KS(n, key)$ (K_n is the nth subkey, KS is the key schedule)

Note:

$$\begin{aligned} L_{i+1} &= R_i \\ R_{i+1} &= L_i \oplus f(R_i, K_{i+1}) \quad i = 0, 1, \ldots, 15 \\ L_i &= R_{i+1} \oplus f(R_i, K_{i+1}) \\ C &= IP^{-1}(R_{16}, L_{16}) \\ IP(C) &= (R_{16}, L_{16}) \end{aligned}$$

(IP^{-1} is the inverse of the initial permutation function)

Decryption, cont.

Suppose $DES_{key}(M) = C$. Denote by

$$K_i' = KS(17 - i, key), \quad i = 1, 2, \ldots, 16$$

(the key schedule in reverse order). Now run the DES device on C, using K_i' instead of K_i. We have $IP(C) = (R_{16}, L_{16})$, and thus

$$\begin{aligned} L_0' &= R_{16} \\ L_1' &= R_0' = L_{16} \\ &= R_{15} \end{aligned} \qquad \begin{aligned} R_0' &= L_{16} \\ R_1' &= L_0' \oplus f(R_0', K_1') \\ &= R_{16} \oplus f(L_{16}, K_{16}) \\ &= R_{16} \oplus f(R_{15}, K_{16}) \\ &= R_{16} \oplus R_{16} \oplus L_{15} \\ &= L_{15} \end{aligned}$$

Decryption, cont.

In fact, by continuing this argument we get

$$L_i' = R_{16-i} \quad R_i' = L_{16-i}$$

and hence

$$IP^{-1}(R_{16}', L_{16}') = IP^{-1}(L_0, R_0) = M$$

Decryption of DES is simply running the DES algorithm on C with the reverse key schedule.

Note 2

The invertibility of DES is *independent* of the function f. Regardless of what function is used for f, decryption works exactly as described above.

- Works largely because the individual parts of DES are *involutions* — functions that are their own inverses ($g(g(x)) = x$)

DES: Assessment

DES was a great cipher at the time:

- highly efficient (for the time)
- lends itself very well to hardware implementation
- same encryption and decryption algorithm (only one chip needed for encryption and decryption)
- withstood all known attacks

Breaking DES

The problem is that for today's computers, the key space of DES is simply too small to foil exhaustive search ($2^{56} \approx 10^{17}$ keys).

- In 1998, the *Electronic Frontier Foundation* built a *DES Cracker* for \$250,000 which finds a single DES key in 56 hours (tests 8800 keys/μ-sec).
- A combination of the DES cracker and 100,000 PCs on the internet has found a DES key in 22.25 hours (tested 245,000 keys/μ-sec).

Multiple DES Encryption

What about multiple DES encryptions? Does this foil exhaustive attacks due to longer key sizes?

Campbell and Wiener (1992) proved that DES is *not* closed, so multiple DES encryptions/decryptions could potentially provide additional security.

- size of the group generated by all the keys (*i.e.* the number of distinct encryptions obtained by applying repeated DES encryptions) has been shown to have size at least $10^{2499} \approx 2^{8302}$. (Estimated number of atoms in the universe: 2^{240}.)

Later, we will show that on double encryption is essentially no more secure than single encryption (but twice as slow).

What about three DES encryption? 3DES (triple DES) is still used.

Triple DES

Use three successive DES operations: $C = DES_{K_1}(DES_{K_2}^{-1}(DES_{K_3}(M)))$

- See NIST Special Publication SP 800-67.

Advantages:

- Same as single key if $K_2 = K_1$ or $K_2 = K_3$.
- Exhaustive search has complexity 2^{112} via the meet-in-the-middle attack (see next week), but with a 168-bit key and a factor of 3 in speed.
- Can use $K_1 = K_3$ with no loss of security.
- No other known practical attacks.

The main disadvantage is that 3-DES is three times slower than single key DES while only doubling the key size.