

CSc 466/566

Computer Security

14 : Cryptography — Symmetric Key

Version: 2019/10/23 11:53:36

Department of Computer Science
University of Arizona

collberg@gmail.com
Copyright © 2019 Christian Collberg

Christian Collberg

1/74

Outline

- 1 Introduction
- 2 Feistel Network
- 3 DES
 - Data Permutations
 - Split input
 - The Rounds
 - The f Function
 - Key Schedule
 - Security
 - Triple DES
 - Software
- 4 Summary

Introduction

2/74

Learning Outcomes

- Structure of block ciphers
- Overall structure of DES
- Triple DES
- Feistel Networks: how do they encrypt and decrypt?
- Algorithm for message padding
- Modes of operations: what are they, how are they used, show how they can fail
- Use of initialization vectors

Introduction

3/74

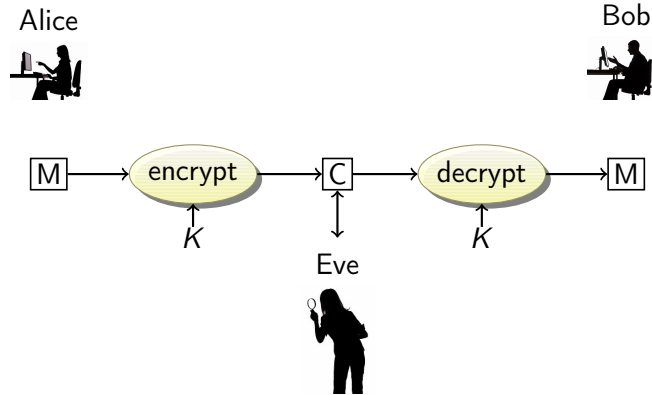
Block Ciphers

- **Block ciphers** work on one block of data at a time.
- A long message is broken up into pieces (blocks) and sent one at a time.
- Block ciphers are **symmetric**, i.e. both the sender and receiver use the same key.
- **Modes of operation** are used to chain message blocks together.

Introduction

4/74

Symmetric Encryption Protocol



XTEA: Extended Tiny Encryption Algo.

<https://en.wikipedia.org/wiki/XTEA>

```
void encipher(
    unsigned int rounds, ← security parameter
    uint32_t block[2], ← block is 64 bits
    uint32_t const key[4]) { ← key is 128 bits

}

}
```

XTEA: Extended Tiny Encryption Algo.

<https://en.wikipedia.org/wiki/XTEA>

```
void encipher(
    unsigned int rounds,
    uint32_t block[2],
    uint32_t const key[4]) {
    unsigned int i;
    uint32_t v0=block[0], v1=block[1], ← init

    for (i=0; i < rounds; i++) { ← iterate
        v0+=... ← update 1st half of block

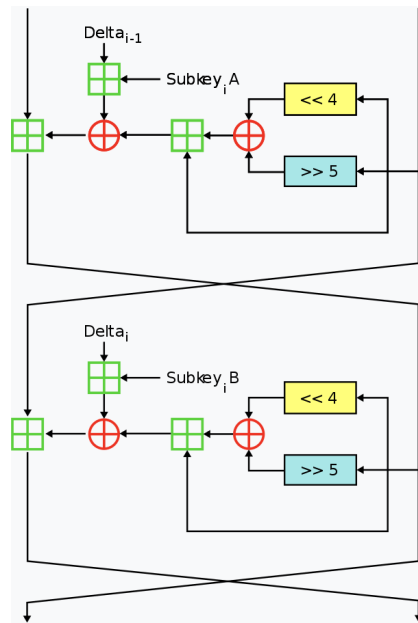
        v1+=... ← update 2nd half of block
    }
    block[0]=v0; block[1]=v1; ← Return block
}
```

XTEA: Extended Tiny Encryption Algo.

<https://en.wikipedia.org/wiki/XTEA>

```
void encipher(
    unsigned int rounds,
    uint32_t block[2],
    uint32_t const key[4]) {
    unsigned int i;
    uint32_t v0=block[0], v1=block[1],
    sum=0, delta=0x9E3779B9;
    for (i=0; i < rounds; i++) {
        v0+=(((v1<<4)^(v1>>5))+v1)^
            (sum+key[sum & 3]);
        sum+=delta;
        v1+=(((v0<<4)^(v0>>5))+v0)^
            (sum+key[(sum>>11)&3]);
    }
    block[0]=v0; block[1]=v1;
}
```

XTEA: Extended Tiny Encryption Algo.

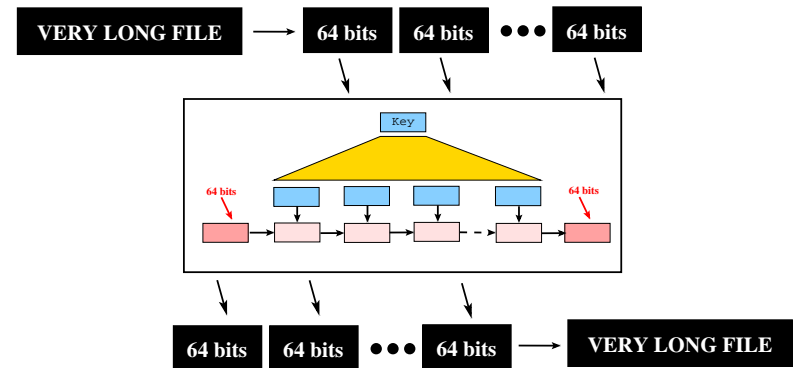


Introduction

<https://en.wikipedia.org/wiki/XTEA>

9/74

Encrypting Large Plaintext

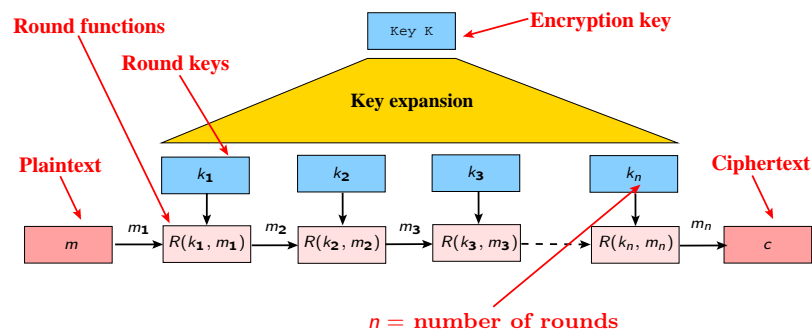


- A large message is encrypted piece-by-piece, each piece of size *blocksize*.
- More later about **modes of operation**, how to assemble/disassemble the sequence of blocks.

Introduction

10/74

Block Ciphers

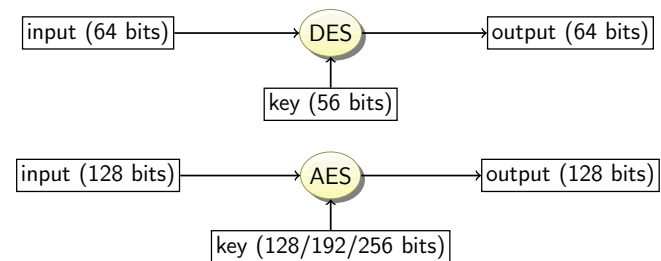


Introduction

11/74

Block Ciphers

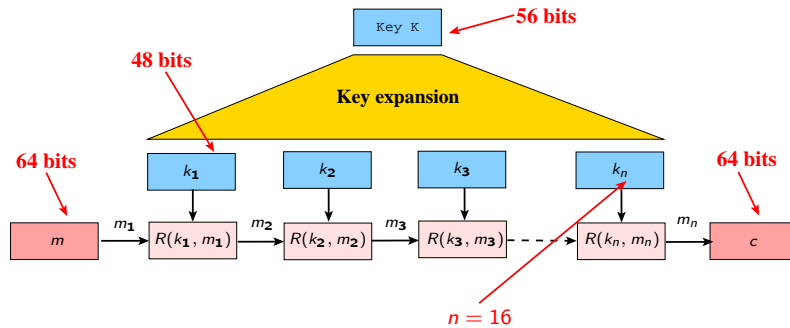
- Different block ciphers use different block size and key length:



Introduction

12/74

DES



Well-known Ciphers

Cipher	Structure	Block Size	Key Size	# S Boxes	# Rounds
DES	Feistel	64	56	8	16
3DES	Feistel	64	168	8	48
Blowfish	Feistel	64	128-448	4	16
IDEA	S-P	64	128	0	8
TEA	Feistel	64	128	0	32
CAST	Feistel	64	40-128	4	12-16
AES	S-P	128	128,192,256	1	10,12,14
RC6	Feistel	128	128,192,256	0	20
Serpent	Feistel	128	128,192,256	8	32
Twofish	Feistel	128	128,192,256	4	16
MARS	Feistel	128	128-448	1	32

Outline

- 1 Introduction
- 2 Feistel Network
- 3 DES
 - Data Permutations
 - Split input
 - The Rounds
 - The f Function
 - Key Schedule
 - Security
 - Triple DES
 - Software
- 4 Summary

Feistel Network

- Many block ciphers are based on the properties of a **Feistel Network**.
- f (see next slide) is called the **round function**, and mixes the key and (part of) the ciphertext (i.e. $f = f(\text{ciphertext}, \text{key})$).
- Each round works on two halves of the plaintext block. We call them L_i and R_i (for left and right half).
- Named after German-born physicist and cryptographer **Horst Feistel**.

Properties of XOR

- We rely on these identities:

$$(A \oplus B) \oplus C = A \oplus (B \oplus C)$$

$$A \oplus B = B \oplus A$$

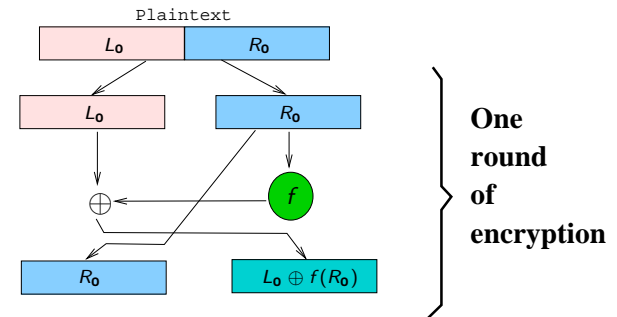
$$A \oplus A = 0$$

$$A \oplus 0 = A$$

- From these, it follows that

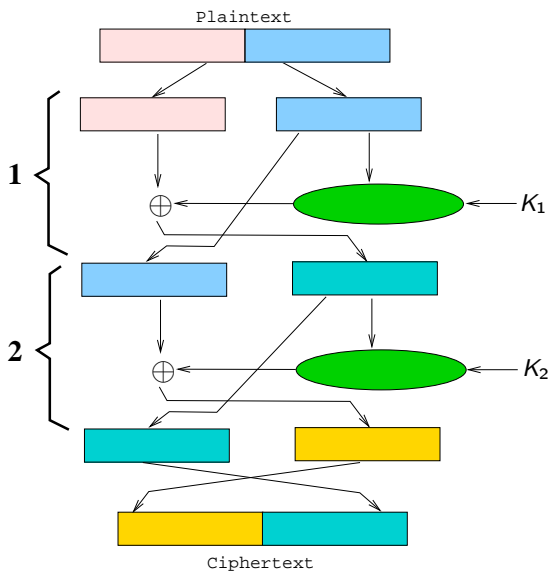
$$(A \oplus A) \oplus A = 0 \oplus A = A$$

1 Round Feistel Network



- Encrypt the plaintext block M , which is broken into two halves (L_0, R_0) .

2 Round Feistel Network



n Round Encryption

- Let $i = 0, \dots, n$ where n is the number of rounds.
- In each round, do:

$$\begin{cases} L_{i+1} = R_i \\ R_{i+1} = L_i \oplus f(R_i, K_i) \end{cases}$$
- The ciphertext becomes (R_{n+1}, L_{n+1}) .

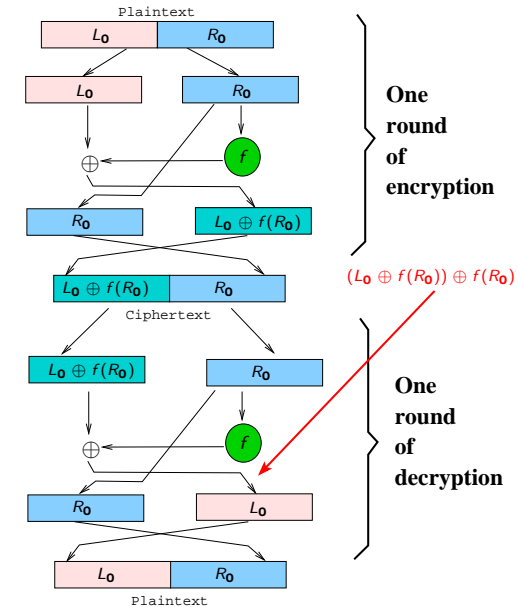
Decryption

- Decrypt the ciphertext $C = (R_{n+1}, L_{n+1})$.
- Let $i = n, \dots, 0$.
- In each round, do:

$$\begin{cases} R_i = L_{i+1} \\ L_i = R_{i+1} \oplus f(L_{i+1}, K_i) \end{cases}$$

The cleartext becomes (L_0, R_0) .
- Same algorithms as encryption, but the keys are applied in the reverse order!

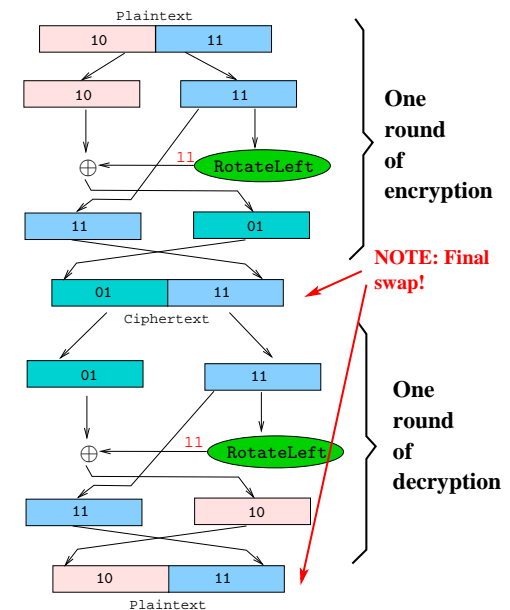
Encryption, then Decryption



Example 1

- Let's try an example. Let's encrypt the 4-bit block 1011.
- In a Feistel network, the function f can be **any function**!!! It doesn't have to be invertible!!!
- Let's choose "RotateLeftBy1" as our function f .

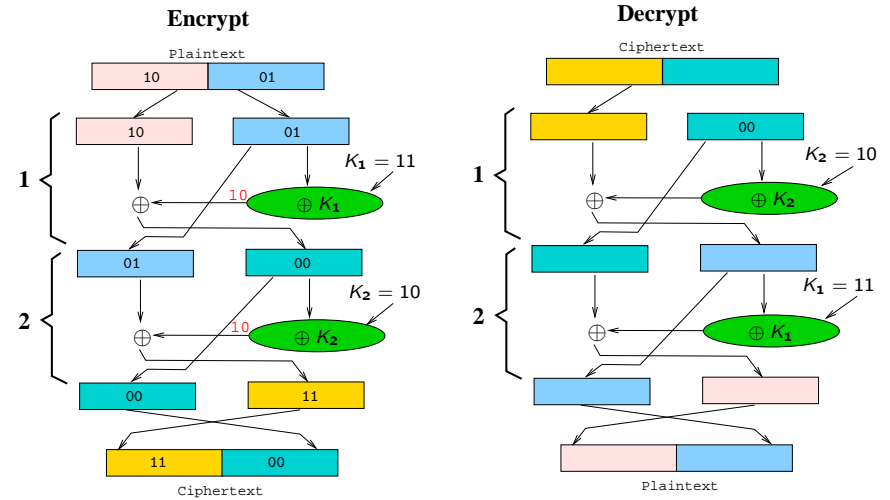
Example 1



Example 2

- In the next example, let's use 2 rounds instead of 1.
- Let's also include a round key K_i .
- Let's choose " $\oplus K_i$ " as our function f .
- Note that we're using the keys in the reverse order. Other than that, decryption is the same as encryption!

Example 2...



Exercise 1

- Use 3 rounds.
- Let $K_1 = 00, K_2 = 11, K_3 = 10$.
- Let $f = \oplus K_i$, as in the previous example.
- Encrypt the 4-bit block 0001!
- Decrypt the resulting ciphertext!

Exercise 1: Encrypt



Exercise 1: Decrypt



Outline

- 1 Introduction
- 2 Feistel Network
- 3 DES
 - Data Permutations
 - Split input
 - The Rounds
 - The f Function
 - Key Schedule
 - Security
 - Triple DES
 - Software
- 4 Summary

DES

- *Data Encryption Standard*. Designed by IBM and “certified” by NSA. Widely used in industry. The NSA is rumored to be able to break it in 3–15 minutes.
- The running example has been taken from J. Orlin Grabbe, *The DES Algorithm Illustrated*, <http://orlingrabbe.com/des.htm>.

DES

- RSA Conference 2011 Keynote - The Cryptographers' Panel:
<http://www.youtube.com/watch?v=0N1Zpyk3PKI>
- *DESVisual: A visualization tool for the DES Cipher*
<http://www.cs.mtu.edu/~shene/NSF-4/ccsc2011.pdf> <http://www.cs.mtu.edu/~shene/NSF-4>
- The Data Encryption Standard -Cryptography-Professor Dan Boneh: <http://www.youtube.com/watch?v=UgFoqxKY7cY>

DES

- The book *Understanding Cryptography: A Textbook for Students and Practitioners* by Christof Paar, Jan Pelzl has a nice description of DES.
- <https://books.google.com/books?id=f24wFELSzkoC>

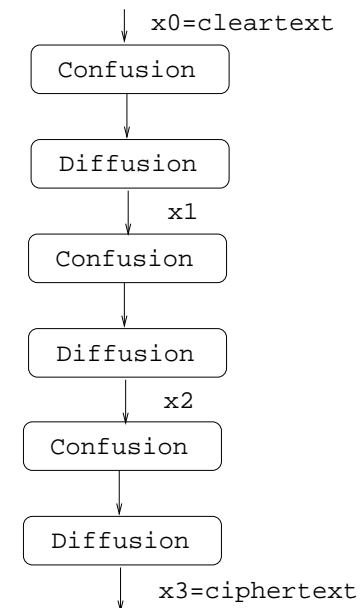
Confusion and Diffusion

- DES is a combination of two basic principles:
 - confusion:** Confusion scrambles up the letters of the plaintext so that there is no direct relationship between the key and the ciphertext. Substitution ciphers do this.
 - diffusion:** Diffusion spreads the plaintext out over the ciphertext. Transposition (AKA permutation) ciphers do this.

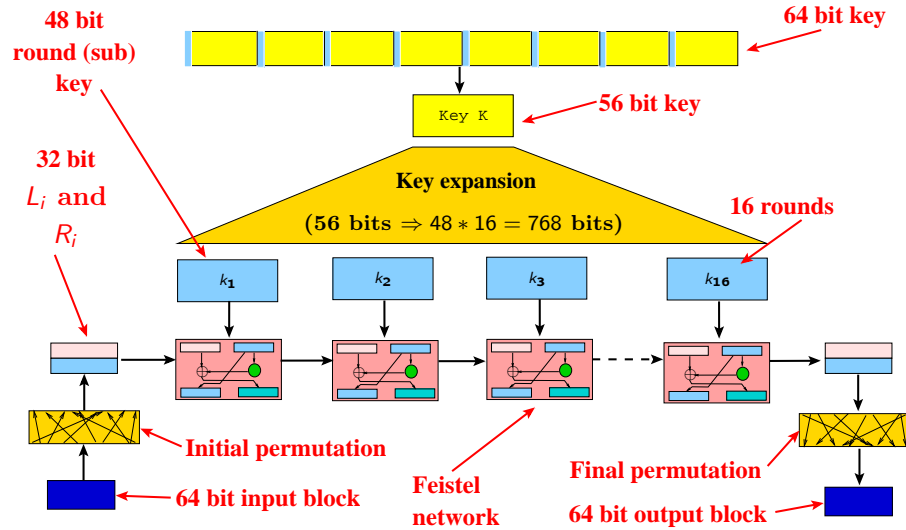
Confusion and Diffusion...

- Ciphers based on confusions only are not secure.
- Ciphers based on diffusion only are not secure.
- Ciphers that use substitution followed by a permutation are called **product ciphers**.
- DES has 16 rounds. Each round consists of a substitution and a permutation.

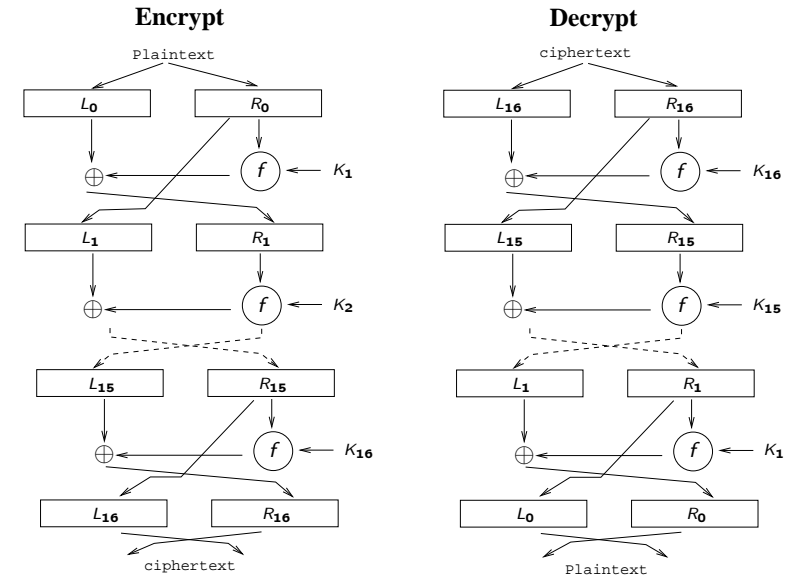
Confusion and Diffusion...



DES Overview



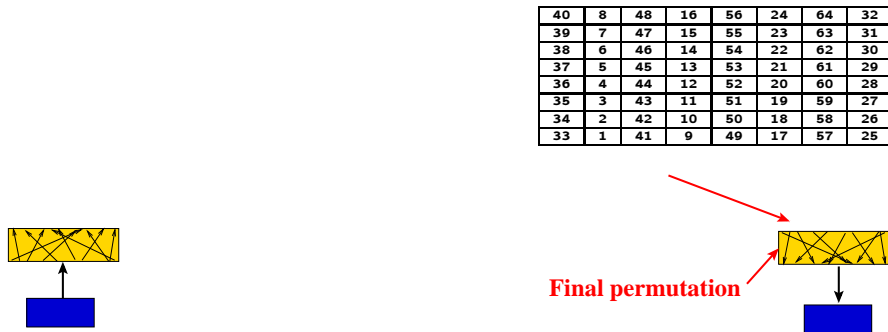
DES: 16 rounds



DES

38/74

Initial and Final Data Permutations



Initial Data Permutation

- We first permute the 64-bit data block with permutation table IP:

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

- Bit 58 of M becomes the bit 1.
- Bit 50 of M becomes the bit 2.
- The final data permutation is just the inverse of the initial one.

DES

40/74

Splitting the Input Block

- Next, we split the permuted 64-bit data block into a left and a right half, L_0 and R_0 :

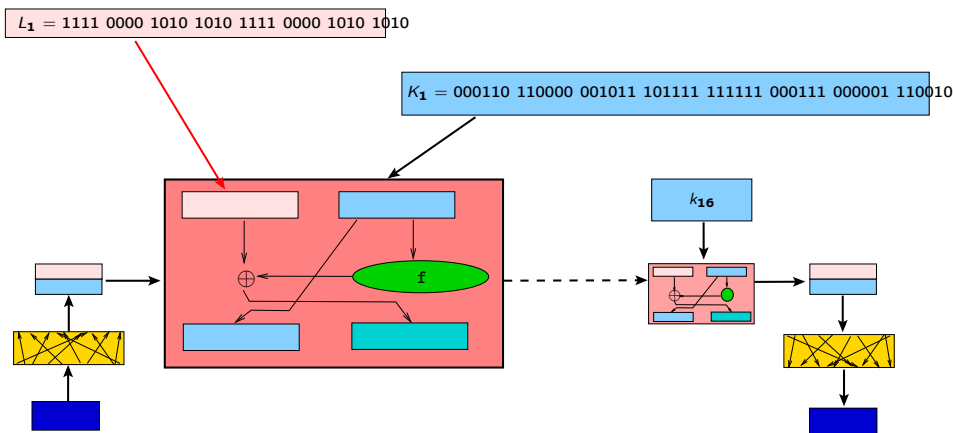
The Rounds

- We iterate 16 times, for $1 \leq i \leq 16$, computing:

$$\begin{aligned} L_n &= R_{n-1} \\ R_n &= L_{n-1} \oplus f(R_{n-1}, K_n) \end{aligned}$$

- In each iteration, the right 32 bits of the previous result become the left 32 bits of the current step.
- We XOR the left 32 bits of the previous step with the calculation f on the key and the right 32 bits.
- At the end, we have a final block $L_{16}R_{16}$.

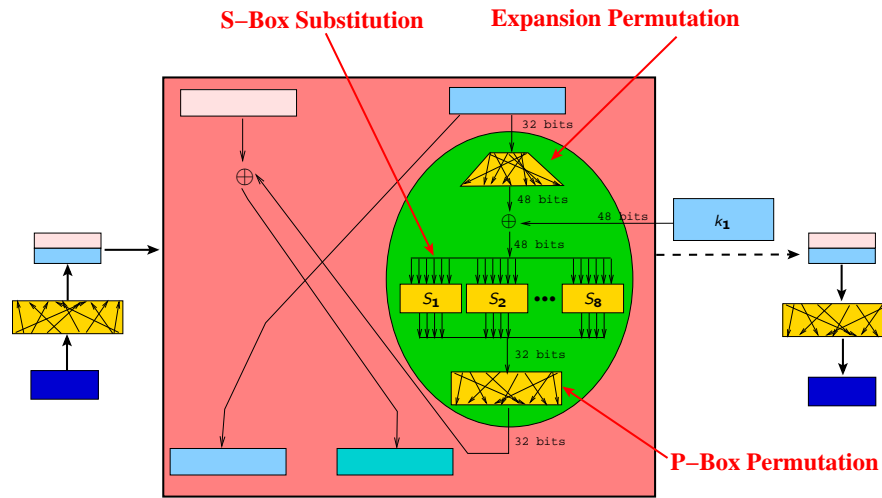
The Rounds...



The f Function

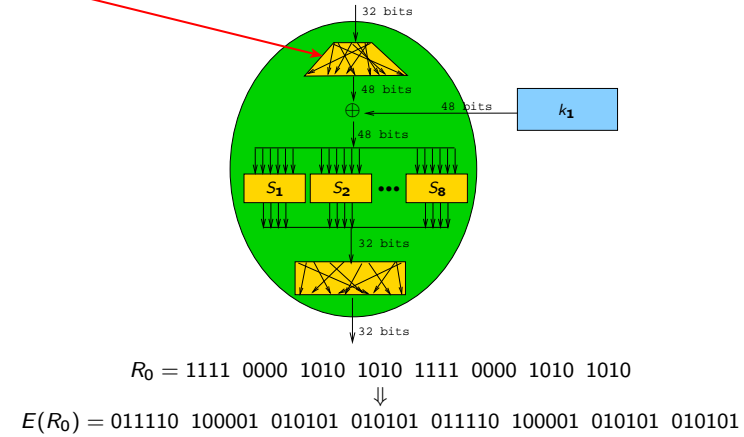
- f is the function that combines the data (the right 32-bit half, R) with the key during each round:
 - Get 48 bits of the key,
 - Expand R to 48 bits using the *expansion permutation*,
 - XOR the expanded R and the compressed key,
 - Send the result through 8 S-boxes using the *S-Box Substitution* to get 32 new bits,
 - Permute the result using the *P-Box Permutation*,
- The result of the f function is XOR:ed with the left half (L) to get the new right half.

The f Function...



The f Function: Expansion Permutation

32	1	2	3	4	5	6	7	8	9
8	9	10	11	12	13	14	15	16	17
16	17	18	19	20	21	22	23	24	25
24	25	26	27	28	29	30	31	32	1



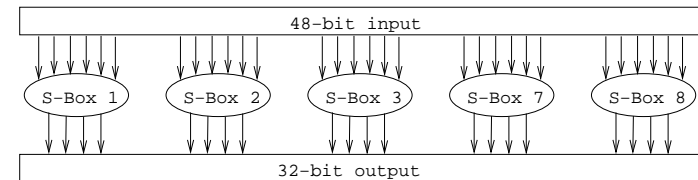
Expansion Permutation

- The expansion permutation takes the 32 bits of R , permutes these bits, and, by copying some of them, expands R into 48 bits.
- The table E below says that bit 32 of R moves into position 1, bit 1 moves to 2, 2 to 3, 4 to 5, 5 to 6, 4 to 7, 5 to 8, etc.
- Notice how every 4-bit block becomes 8 bits.

32	1	2	3	4	5	4	5	6	7	8	9
8	9	10	11	12	13	12	13	14	15	16	17
16	17	18	19	20	21	20	21	22	23	24	25
24	25	26	27	28	29	28	29	30	31	32	1

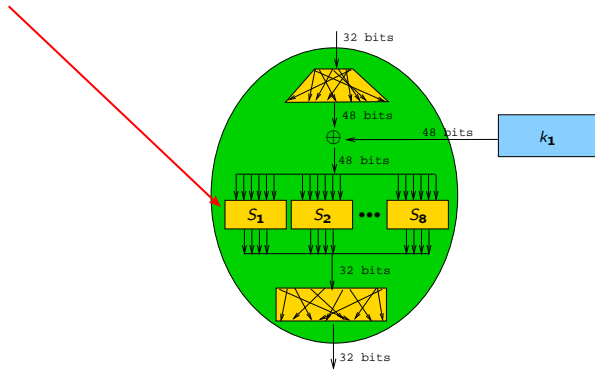
S-Box Substitution

- The expanded right (R_{i-1}) and the compressed key (K_i) are XORed together. The 48-bit result is then passed through 8 S-Boxes (Substitution Boxes) to form 32 bits.
- Each one of the 8 S-Boxes is different. Each takes 6 bits of input and produces 4 bits of output:



The f Function: S-Box Substitution

	column number															
row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

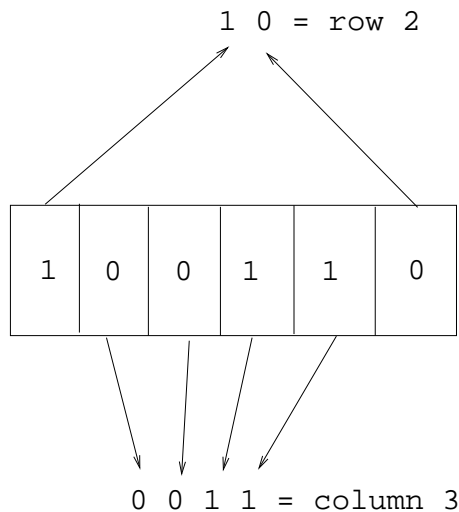


S-Box Substitution Tables

- Each S-Box is given by a table consisting of 4 rows of 16 numbers. Each number is a 4-bit quantity.
- The input to each S-Box is 6 bits:

b_1	b_2	b_3	b_4	b_5	b_6
-------	-------	-------	-------	-------	-------
- The S-Box table is indexed by taking the outer 2 bits (b_1 and b_6) and forming a number between 0 and 3. This is used to get the row number, The middle 4 bits ($b_2 \dots b_5$) get the column number.
- The S-Boxes are what gives DES its security.

S-Box Substitution Tables...



S-Box Substitution: S-Box 1

	column number															
row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

- Input block $B = b_1 b_2 b_3 b_4 b_5 b_6 = 011011$.
- Row index = $b_1 b_6 = 01_2 = 1_{10}$
- Column index = $b_2 b_3 b_4 b_5 = 1101_2 = 13_{10}$.
- $\Rightarrow S_1(011011) = 5_{10} = 0101_2$.

Exercise: S-Box Substitution

row	column number															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

• Input block $B = b_1b_2b_3b_4b_5b_6 = 100001$.

• Row index = $b_1b_6 =$



• Column index = $b_2b_3b_4b_5 =$



• $\Rightarrow S_1(100001) =$



S-Box Substitution

S-Box 1															
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S-Box 2															
15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S-Box 3															
10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S-Box Substitution...

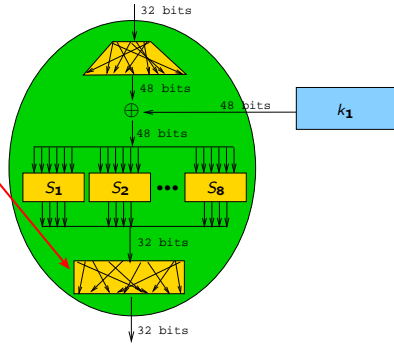
S-Box 4															
7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S-Box 5															
2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S-Box 6															
12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S-Box Substitution

S-Box 7															
4	11	2	14	15	0	8	13	3	12	9	7	6	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S-Box 8															
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

The f Function: P-Box Permutation Tables

16	7	20	21	29	12	28	17	1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9	19	13	30	6	22	11	4	25

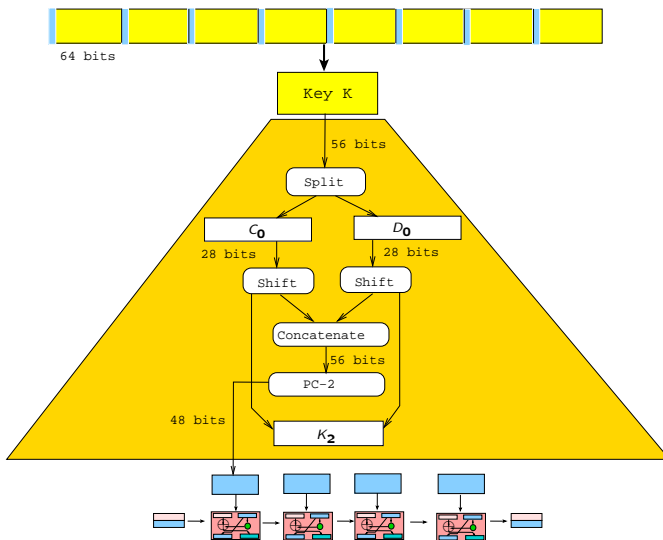


P-Box Permutation Tables

- The final operation in the f function is the P-Box.
- It's a straight permutation of the 32 bits that come out of the S-Box.
- Bit number 16 moves into position 1, bit 7 into 2, 20 into 3, etc:

16	7	20	21	29	12	28	17	1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9	19	13	30	6	22	11	4	25

Key Schedule



Key Schedule...

- DES runs in 16 rounds.
- At each round we need a 48 bit key.
- We take the original 64 bit key, and extract 56 bits.
- In each round a new version of the key is generated to be used in the next round:
 - Split the key into two halves, each 28 bits.
 - Shift each half by 1 or 2 steps.
 - Compress the two halves into 48 bits using the *Compression Permutation*,
 - Merge the two shifted halves into the key for the next round.

Key Schedule — Final Keys

i	K_i							
1	000110	110000	001011	101111	111111	000111	000001	110010
2	011110	011010	111011	011001	110110	111100	100111	100101
3	010101	011111	110010	001010	010000	101100	111110	011001
4	011100	101010	110111	010110	110110	110011	010100	011101
5	011111	001110	110000	000111	111010	110101	001110	101000
6	011000	111010	010100	111110	010100	000111	101100	101111
7	111011	001000	010010	110111	111101	100001	100010	111100
8	111101	111000	101000	111010	110000	010011	101111	111011
9	111000	001101	101111	101011	111011	011110	011110	000001
10	101100	011111	001101	000111	101110	100100	011001	001111
11	001000	010101	111111	010011	110111	101101	001110	000110
12	011101	010111	000111	110101	100101	000110	011111	101001
13	100101	111100	010111	010001	111110	101011	101001	000001
14	010111	110100	001110	110111	111100	101110	011100	111010
15	101111	111001	000110	001101	001111	010011	111100	001010
16	110010	110011	110110	001011	000011	100001	011111	110101

Weak keys

These keys are weak or semi-weak and should be avoided (p is 0 or 1, P is e or f):

0x0p0p0p0p0p0p0p	0x0p1P0p1P0p0P0p0P
0x0pep0pep0pfp0pfP	0x0pfP0pfP0pfP0pfP
0x1P0p1P0p0P0p0P0p	0x1P1P1P1P0P0P0P0P
0x1Pep1Pep0Pfp0Pfp	0x1PfP1PfP0PfP0PfP
0xep0pep0pfp0pfp0p	0xep1Pep1pfP0PfP0P
0xepepepepepepepepep	0xepfPepfPfPpfPfPpfP
0xfP0pfP0pfP0pfP0p	0xfP1PfP1PfP0PfP0P
0xfPepfPepfPepfPep	0xfPfPfPfPfPfPfPfPfP

DES Timeline

en.wikipedia.org/wiki/Data_Encryption_Standard

- 1973: NBS request for a standard encryption algorithm
- 1976: DES is approved as a standard
- 1998: The EFF's DES cracker (Deep Crack) breaks a DES key in 56 hours.
- 1999: Deep Crack + distributed.net break a DES key in 22 hours and 15 minutes.

DES Timeline...

- 2001: The Advanced Encryption Standard is published in FIPS 197
- 2006: The FPGA based parallel machine COPACOBANA breaks DES in 9 days at \$10,000 hardware cost.
- 2007: COPACOBANA: software improvements reduced the average time to 6.4 days.
- 2008: The RIVYERA machine reduced the average time to less than a single day.

DES Crack

To brute-force DES keys, we use a set of field-programmable gate arrays (FPGA), which became trendy for Bitcoin mining a couple of years ago and got cheaper after the hype was over. The speed of our 8 modules *ZTEX 1.15y board with the price tag of 2,000 Euro is 245.760 Mcrypt/sec. It is enough to obtain the key within 3 days.

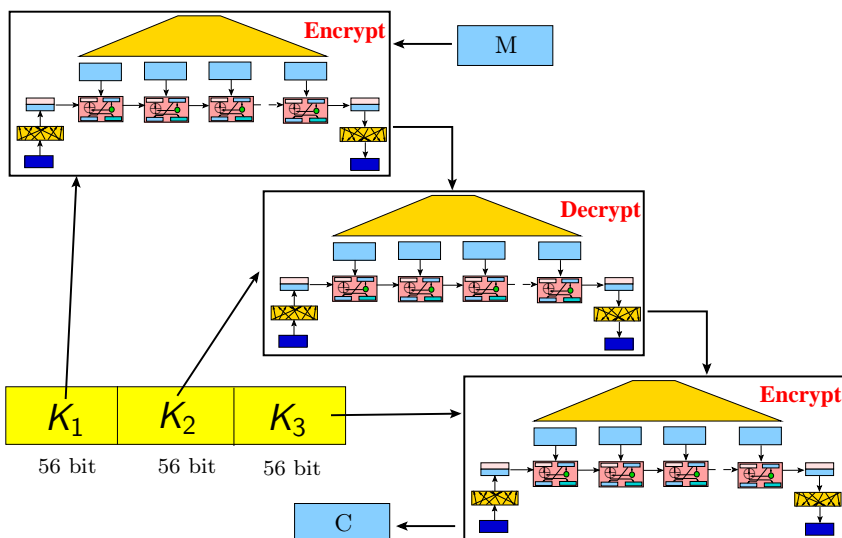
<http://threatpost.com/majority-of-4g-usb-modems-sim-cards-exploitable/110139>

DES Crack...



<https://crack.sh>

Triple DES



Triple DES...

- Three DES keys, K_1 , K_2 and K_3 , each 64 bits (56 bits of actual key + 8 parity bits).
- Blocks are still 64 bits.
- Encryption:

$$C = E_{K_3}(D_{K_2}(E_{K_1}(M)))$$


DES encrypt with K_1 , DES decrypt with K_2 , then DES encrypt with K_3 .

- Decryption:

$$M = D_{K_1}(E_{K_2}(D_{K_3}(C)))$$

Decrypt with K_3 , encrypt with K_2 , then decrypt with K_1 .

Triple DES — Picking keys

- There are three options for picking keys:
 - ① All three keys are independent (168 key bits)
 - ② K_1 and K_2 are independent, $K_3 = K_1$ (112 key bits).
 - ③ $K_1 = K_2 = K_3$ (56 key bits).
- Option 3 is good for backwards compatibility. Why? 
- 3DES is still used in banking applications.

Software – openssl

```
> cat message
Attack at dawn, actually, no, let's make it 3 am
since, you know, they expect us at dawn, but,
maybe they will expect us at 3 am since, we
said we'd attack at dawn, so let's attack at
dawn anyway! Unless...
> openssl enc -e -des -in message -a \
-out message.asc -pass pass:Alice4Evah
> cat message.asc
```

```
U2FsdGVkX1/hHZThWEArEyxc0Z6dT4s1zfFDUovG+yCBNEpk1M+m7jevbg+bJpE9
IgxkZLVzncgIWYKQqPLjbmBJfKVRBHCFdfbS+oi4dxj3PfMygm/HN5QjeC7TIJL
muXX+c0SExJ4GKifxB08VBT8baj8a06k2Qfex9JuXh0dHSqCf02GHgr4b7s/5us0
OHgrVZP9RIaCMBEB+9uu6oDVlApK/GoWipf8/S5jz7qbdjjaT6Cw00+HImNPEE9w
k1xYs5CkNU6fVHH3b1SIyJHxxf6REUkxA3immtFVidsn2sOHJ0g36A==
```

Software – openssl

```
> openssl enc -d -des -in message.asc \
-a -pass pass:Alice4Evah
Attack at dawn, actually, no, let's make it 3 am
since, you know, they expect us at dawn, but,
maybe they will expect us at 3 am since, we
said we'd attack at dawn, so let's attack at
dawn anyway! Unless...
```

Outline

- 1 Introduction
- 2 Feistel Network
- 3 DES
 - Data Permutations
 - Split input
 - The Rounds
 - The f Function
 - Key Schedule
 - Security
 - Triple DES
 - Software
- 4 Summary

Readings and References

- Chapter 8.1.6 in *Introduction to Computer Security*, by Goodrich and Tamassia.
- The Enigma Secret: <http://www.youtube.com/watch?v=IJToxIZMbZQ&feature=related>
- J. Orlin Grabbe, The DES Algorithm Illustrated, <http://orlingrabbe.com/des.htm>.

Acknowledgments

Additional material and exercises have also been collected from these sources:

- 1 Igor Crk and Scott Baker, *620—Fall 2003—Basic Cryptography*.
- 2 J. Orlin Grabbe, The DES Algorithm Illustrated, <http://orlingrabbe.com/des.htm>.
- 3 Andrea Sanchez, *DES Algorithm*, <https://www.youtube.com/watch?v=dRH585Ctp3E>.
- 4 Dan Boneh, *The Data Encryption Standard -Cryptography*, <https://www.youtube.com/watch?v=UgFoqxKY7cY>.