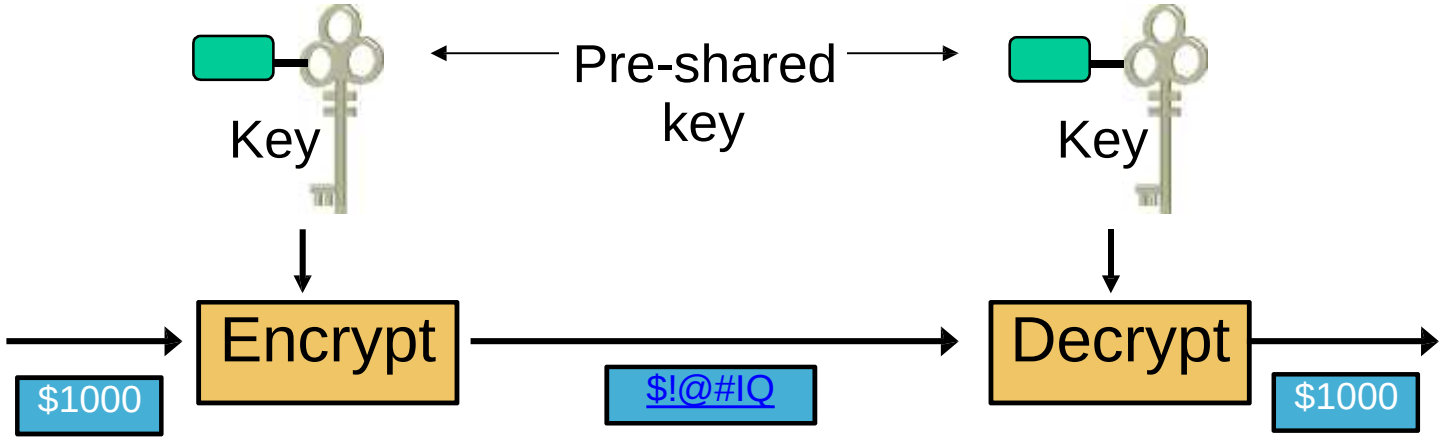
COM3020J: Cryptography
Symmetric Crypto:

Dr. Anca Jurcut
E-mail: anca.jurcut@ucd.ie

School of Computer Science and Informatics
University College Dublin

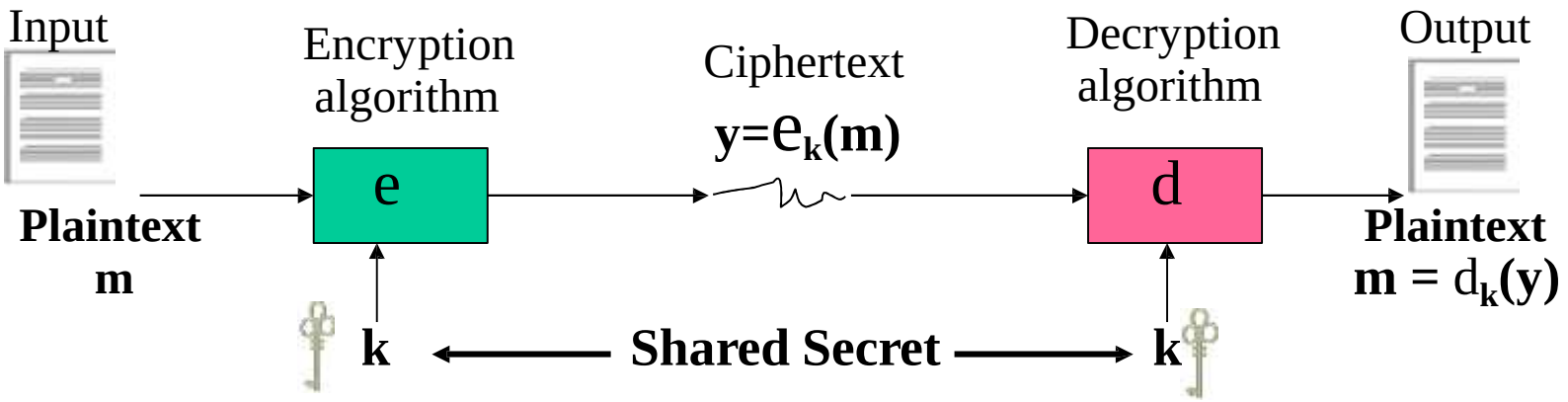
Beijing-Dublin International College

Symmetric Key (Conventional) Crypto



- **Unique key for correspondence**
- **A sender and receiver must share a secret key**
- **Keys must be delivered/distributed in a secure manner**
- **Faster processing because they use simple mathematical operations.**

Symmetric Key (Conventional) Crypto



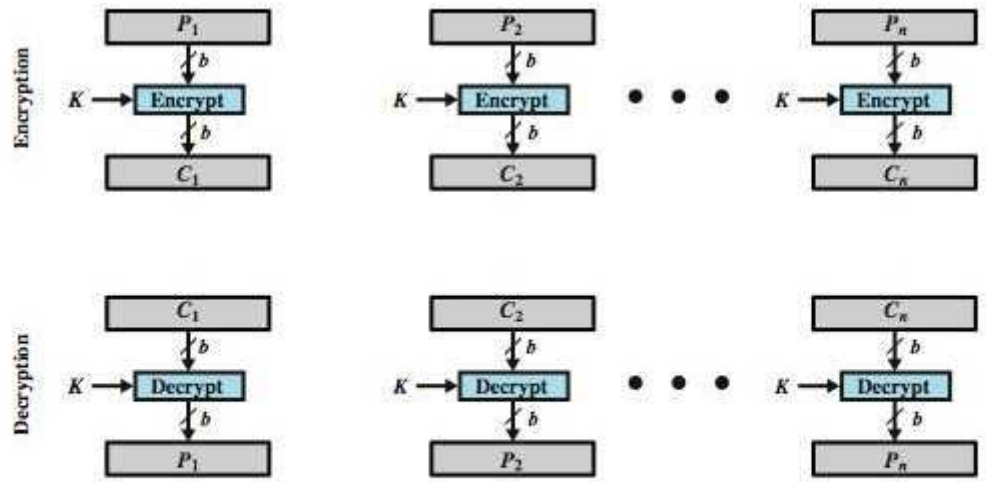
- **Communicating parties share a secret key (k)**
- **Encryption followed by decryption, using the same key, causes the original message to be recovered [$m = d_k(e_k(m))$]**
- **For secrecy/confidentiality between A and B, only A and B must know the shared key (k).**

Symmetric Key Crypto

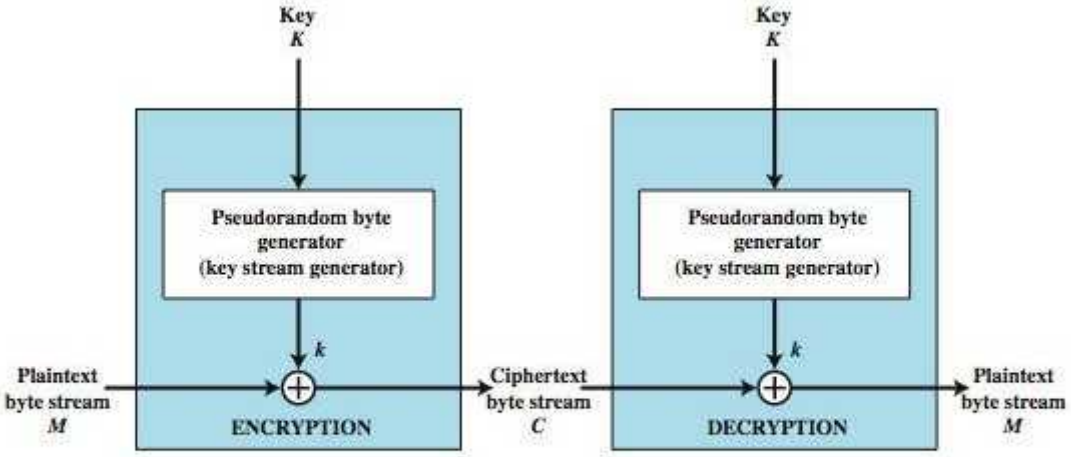
2 Types:

- **Stream cipher** — generalize one-time pad
  - Except that key is relatively short
  - Key is stretched into a long **keystream**
  - Keystream is used just like a one-time pad
- **Block cipher** — generalized codebook
  - Block cipher key determines a codebook
  - Each key yields a different codebook
  - Employs both “confusion” and “diffusion”

Block verses Stream Ciphers



(a) Block cipher encryption (electronic codebook mode)



(b) Stream encryption

Stream Ciphers

- Once upon a time, not so very long ago… stream ciphers were the king of crypto
- Today, not as popular as block ciphers
- Operate on the plaintext a single bit (or sometimes byte) at a time as a stream
- Simple substitution - adds bits of message to random key bits
- The most famous: Vernam cipher
  (Invented by Vernam, AT&T in 1917)

Stream Ciphers

- **Usage**
  - Stream ciphers are used in applications where plaintext comes in quantities of unknowable length - for example, a secure wireless connection
- **Drawbacks**
  - Need as many key bits as message, difficult in practice
- **Strengths**
  - Is unconditionally secure provided key is truly *random*
- **Stream Ciphers Examples**:
  - **A5/1** (based on shift registers, used in GSM mobile phone system),
  - **RC4** (based on a changing lookup table, used in many places),
  - …FISH, Helix, ISAAC, Panama, Pike, SEAL, SOBER, WAKE

RC4

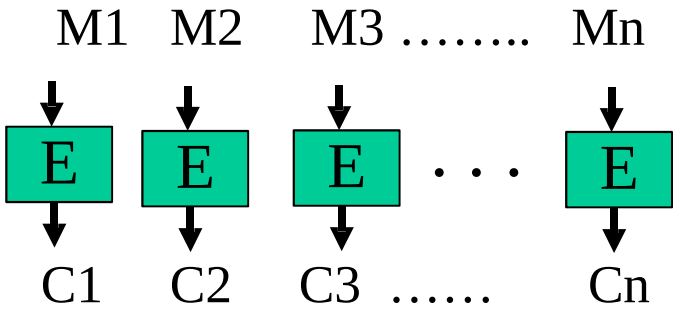
- A self-modifying lookup table
- Table always contains a permutation of the byte values 0,1,…,255
- Initialize the permutation using key
- At each step, RC4 does the following
  - Swaps elements in current lookup table
  - Selects a keystream byte from table
- Each step of RC4 produces a **byte**
  - Efficient in software
- Each step of **A5/1** produces only a bit
  - Efficient in hardware

Stream Ciphers

- Stream ciphers were popular in the past
  - Efficient in hardware
  - Speed was needed to keep up with voice, etc.
  - Today, processors are fast, so software-based crypto is usually more than fast enough
- Future of stream ciphers?
  - Shamir declared “the death of stream ciphers”
  - May be greatly exaggerated…
  - …because people are trying to do cryptography on very small devices with minimum computing power , etc

Block Ciphers

- Output should look random
- No correlation between plaintext and ciphertext
  - bit spreading



- The plaintext message is broken into groups of bits (blocks)
  - Each of which is then encrypted
  - Typical block size is 64 bits or multiple of it (e.g. 128 bits, 256 bits, 512bits)

(Iterated) Block Cipher

- Plaintext and ciphertext consist of fixed-sized blocks
- Ciphertext obtained from plaintext by iterating a **round function**
- Input to round function consists of ***key*** and ***output*** of previous round
- Usually implemented in software

Symmetric Encryption and XOR

The XOR operator results in a 1 when the value of either the *first bit* or the *second bit* is a 1

The XOR operator results in a 0 when *neither* or *both* of the bits is 1

Plain Text	**1**	**1**	**0**	**1**	**0**	**0**	**1**	**1**
Key (Apply)	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**
XOR (Cipher Text)	**1**	**0**	**0**	**0**	**0**	**1**	**1**	**0**
Key (Re-Apply)	**0**	**1**	**0**	**1**	**0**	**1**	**0**	**1**
XOR (Plain Text)	**1**	**1**	**0**	**1**	**0**	**0**	**1**	**1**

Feistel Cipher: Encryption

- **Feistel cipher** is a type of block cipher
  - ***Not*** a specific block cipher
- Split plaintext block into left and right halves:

$$\mathbf{P = (L_0, R_0)}$$

- For each round i = 1, 2, ..., n, compute

$$\mathbf{L_i = R_{i-1}}$$

$$\mathbf{R_i = L_{i-1} \oplus F(R_{i-1}, K_i)}$$

  where F is **round function** and K_i is **subkey**
- C iphertext: $\mathbf{C = (L_n, R_n)}$

Feistel Cipher: Decryption

- Start with ciphertext $C = (L_n, R_n)$
- For each round $i = n, n-1, \ldots, 1$, compute

$$\mathbf{R_{i-1} = L_i}$$

$$\mathbf{L_{i-1} = R_i \oplus F(R_{i-1}, K_i)}$$

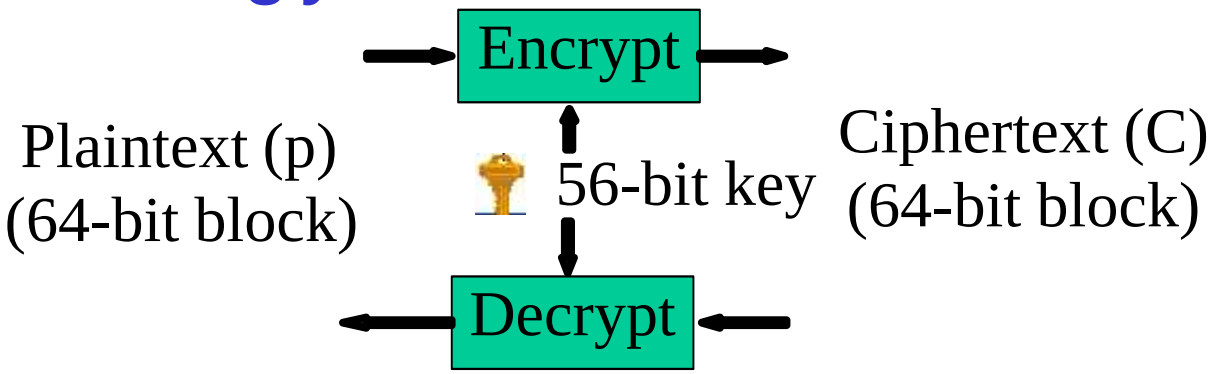
  where F is round function and K_i is subkey

- Plaintext: $\mathbf{P = (L_0, R_0)}$
- Decryption works for any function F
  - But only secure for certain functions F

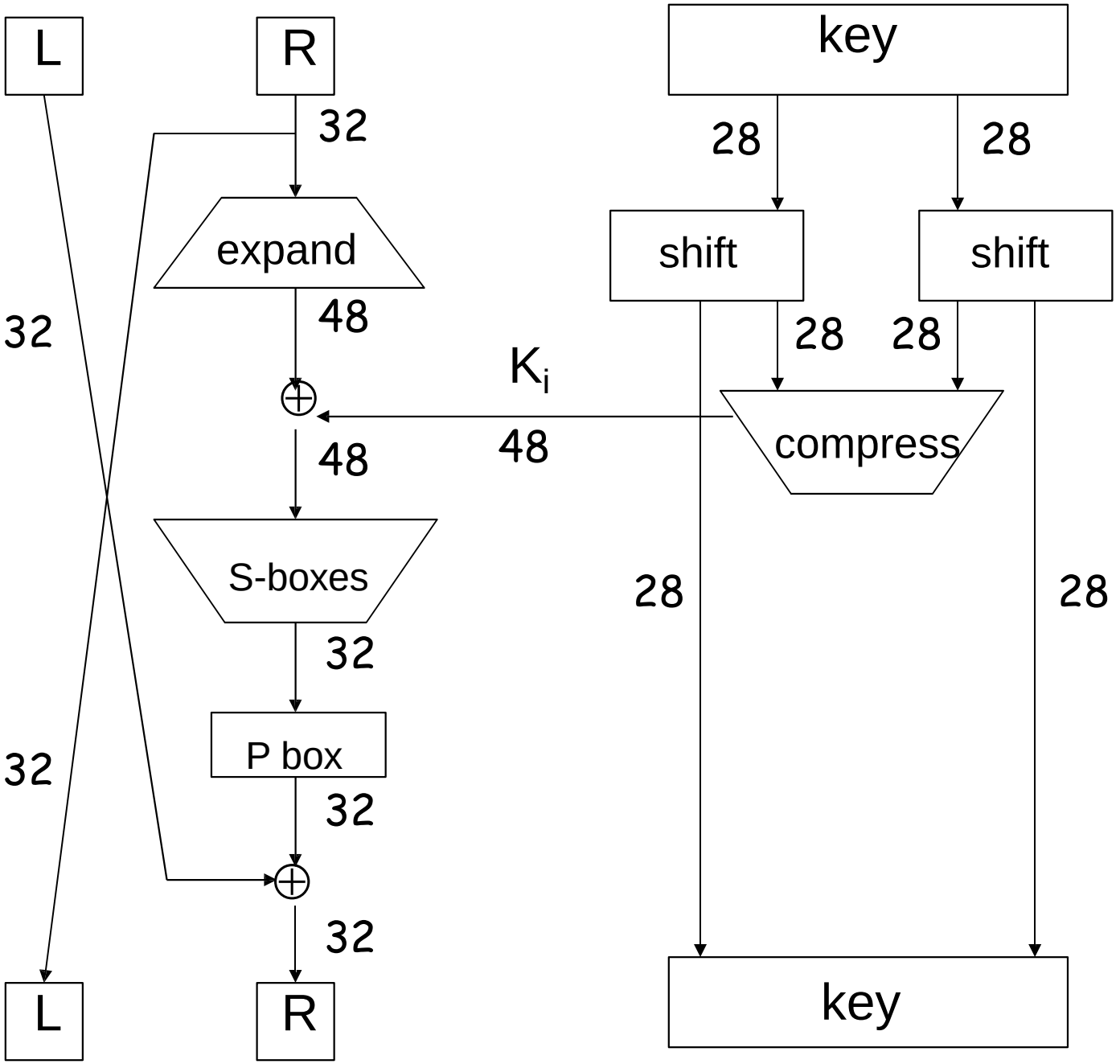
Data Encryption Standard

- **DES** developed in 1970's
- Based on IBM's Lucifer cipher
- DES was U.S. government standard
- DES was controversial
  - NSA secretly involved
  - Design process was secret
  - Key length reduced from 128 to 56 bits
  - Subtle changes to Lucifer algorithm

DES Numerology



- DES is a Feistel cipher with…
  - 64 bit block length: 64 bit input (plaintext), 64 bit output (ciphertext).
  - 56 bit key length
  - 16 rounds for Encryption & Decryption
  - 48 bits of key used each round (subkey)
- Round function is simple (for block cipher)
- Security depends heavily on "S-boxes"
  - Each S-box maps 6 bits to 4 bits



One Round of DES

DES Expansion Permutation

- Input 32 bits

```
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
```

- Output 48 bits

```
31  0  1  2  3  4  3  4  5  6  7  8
 7  8  9 10 11 12 11 12 13 14 15 16
15 16 17 18 19 20 19 20 21 22 23 24
23 24 25 26 27 28 27 28 29 30 31  0
```

DES S-box

- 8 “substitution boxes” or S-boxes
- Each S-box maps 6 bits to 4 bits
- Here is S-box number 1

input bits (0,5)
↓ input bits (1,2,3,4)

	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
00	1110	0100	1101	0001	0010	1111	1011	1000	0011	1010	0110	1100	0101	1001	0000	0111
01	0000	1111	0111	0100	1110	0010	1101	0001	1010	0110	1100	1011	1001	0101	0011	1000
10	0100	0001	1110	1000	1101	0110	0010	1011	1111	1100	1001	0111	0011	1010	0101	0000
11	1111	1100	1000	0010	0100	1001	0001	0111	0101	1011	0011	1110	1010	0000	0110	1101

DES P-box

- Input 32 bits

```
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
```

- Output 32 bits

```
15  6 19 20 28 11 27 16  0 14 22 25  4 17 30  9
 1  7 23 13 31 26  2  8 18 12 29  5 21 10  3 24
```

DES Subkey

- 56 bit DES key, numbered 0,1,2,…,55
- Left half key bits, LK

```
49 42 35 28 21 14  7
 0 50 43 36 29 22 15
 8  1 51 44 37 30 23
16  9  2 52 45 38 31
```

- Right half key bits, RK

```
55 48 41 34 27 20 13
 6 54 47 40 33 26 19
12  5 53 46 39 32 25
18 11  4 24 17 10  3
```

DES Subkey

- For rounds `i=1,2,...,16`
  - Let LK = (LK circular shift left by r_i)
  - Let RK = (RK circular shift left by r_i)
  - Left half of subkey K_i is of LK bits

```
13 16 10 23  0  4  2 27 14  5 20  9
22 18 11  3 25  7 15  6 26 19 12  1
```

  - Right half of subkey K_i is RK bits

```
12 23  2  8 18 26  1 11 22 16  4 19
15 20 10 27  5 24 17 13 21  7  0  3
```

DES Subkey

- For rounds 1, 2, 9 and 16 the shift r_i is 1, and in all other rounds r_i is 2
- Bits 8,17,21,24 of LK omitted each round
- Bits 6,9,14,25 of RK omitted each round
- **Compression permutation** yields 48 bit subkey K_i from 56 bits of LK and RK
- **Key schedule** generates subkey

DES Last Word (Almost)

- An initial permutation before round 1
- Halves are swapped after last round
- A final permutation (inverse of initial perm) applied to (R_{16}, L_{16})
- None of this serves any security purpose

Security of DES

- Security depends heavily on S-boxes
  - Everything else in DES is linear
- 35+ years of intense analysis has revealed no back door
- Attacks, essentially exhaustive key search
- **Inescapable conclusions**
  - Designers of DES knew what they were doing
  - Designers of DES were way ahead of their time (wrt certain cryptanalytic techniques)

Block Cipher Notation

- P = plaintext block
- C = ciphertext block
- Encrypt P with key K to get ciphertext C
  - C = E(P, K)
- Decrypt C with key K to get plaintext P
  - P = D(C, K)
- Note: P = D(E(P, K), K) and C = E(D(C, K), K)
  - But $P \neq D(E(P, K_1), K_2)$ and $C \neq E(D(C, K_1), K_2)$ when $K_1 \neq K_2$

Symmetric Encryption Algorithms

	DES	Triple DES	AES
Plaintext block size (bits)	64	64	128
Ciphertext block size (bits)	64	64	128
Key size (bits)	56	112 or 168	128, 192, or 256

DES = Data Encryption Standard
AES = Advanced Encryption Standard