CS458: Introduction to Information Security

Notes 3: Symmetric Cryptography

Yousef M. Elmehdwi

Department of Computer Science

Illinois Institute of Technology

yelmehdwi@iit.edu

January 29th, 2024

Slides: Modified from Christof Paar and Jan Pelzl, Computer Security: Principles and Practice, 4th Edition, Stephen R. Tate UNC Greensboro & Nadia Heninger at UCSD

Outline

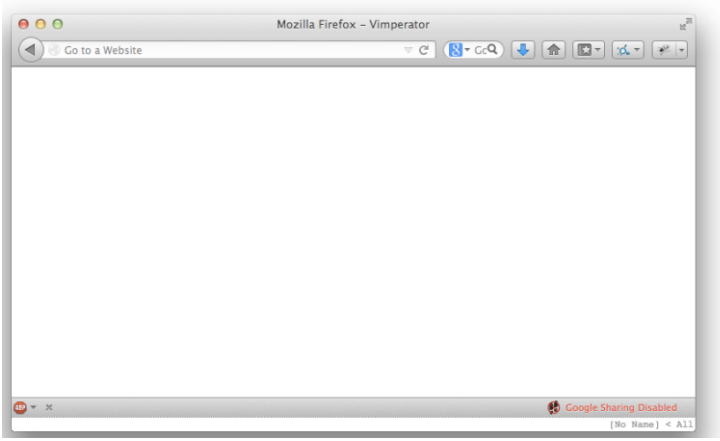
- Threat Model
- Symmetric encryption principles
- Data Encryption Standard (DES)
- Advanced Encryption Standard (AES)
- Cipher block modes of operation
- Key distribution

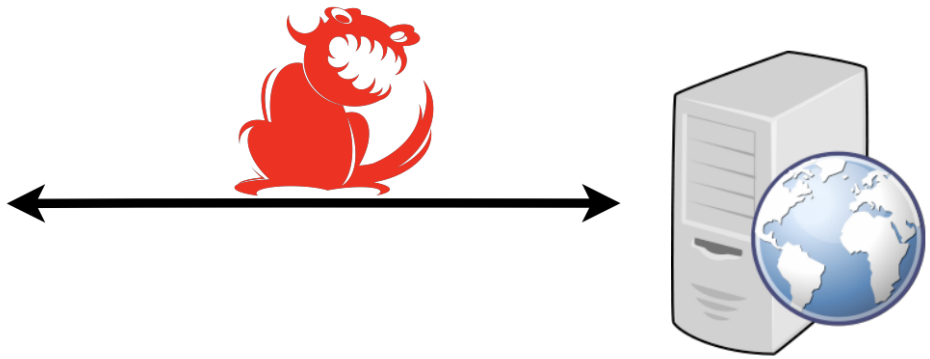
Cryptography

- Is
  - A tremendous tool
  - The basis for many security mechanisms
- Is not
  - The solution to all security problems
  - Reliable unless implemented and used properly
  - Something you should try to invent yourself unless you have sufficient academic knowledge to do so
    - *it is very difficult to ensure security in cryptography*

Real-world crypto: SSL/TLS[1]

- How browsers encrypt communications with web servers



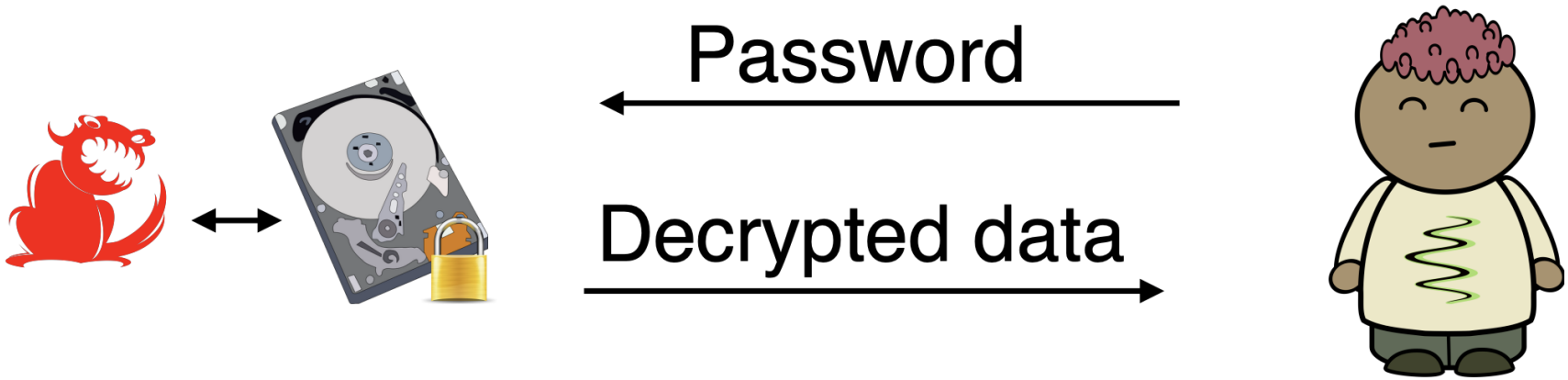


  - Browser and web server run handshake protocol
    - Establishes shared secret key using public-key cryptography (either the RSA or the Diffie-Hellman key exchange)
  - Browser and web server use negotiated key to secure the communication.

[1] *SSL (Secure Socket Layer) and TLS (Transport Layer Security) are popular cryptographic security protocols that are used to imbue web communications with integrity, security, and resilience against unauthorized tampering.*
HTTPS is an implementation of TLS encryption on top of the HTTP protocol, which is used by all websites as well as some other web services.

Real-world crypto: File encryption

- Encrypts disk contents



  - Files are symmetrically encrypted with a secret key
  - The symmetric key is stored encrypted or in tamperproof hardware.
  - The password is used to unlock the key so the data can be decrypted.

Cryptography: Threat Model

- Structured approach to identifying and assessing potential threats and vulnerabilities within a cryptographic system.
- It helps in understanding the security risks involved and designing appropriate countermeasures to mitigate them.
- It often includes assumptions about adversary capabilities and behaviors as part of that assessment.
- These assumptions help define the scope of potential threats and vulnerabilities that need to be addressed in a security system
- Elements in a threat model for cryptography:
  - Adversaries, Motives, Capabilities, Knowledge, Attack Vectors, Countermeasures, etc

Cryptography: Threat Model

- Define the scope of security concerns.
- Identify potential attackers, their motivations, and capabilities.
- Consider attack vectors and vulnerabilities.
- Assess the value of assets to be protected.

Cryptography: Threat Model Elements

Element	Description
Adversaries	Who might attack the system and why (e.g., hackers, competitors, insiders)?
Motives	What are the attackers' goals (e.g., steal data, disrupt operations, gain access)?
Capabilities	What resources and skills do the attackers have (e.g., technical expertise, access to tools)?
Knowledge	What information do the attackers have about the system (e.g., algorithms, key lengths, configurations)?
Attack vectors	How might the attackers exploit vulnerabilities (e.g., brute-force attacks, social engineering)?
Countermeasures	What actions can be taken to mitigate the identified threats (e.g., strong encryption algorithms, secure key management, access controls)?

Threat Model : Adversary Knowledge - Algorithms

- Saltzer and Schroeder Design Principles: Open Design
  - Much older idea than Saltzer and Schroeder...
    - **Kerckhoff's Principle (1883)**: *The security of a cryptosystem depends on the strength of the algorithm and the secrecy of the key.*
- Trying to keep algorithms secret ("security through obscurity") almost never works.
- Bottom Line: Better to use a system that experts have tried (and failed) to break

Threat Model : Adversary Knowledge - Behavior

- Some things an attacker might know:
  - Language of messages (e.g., English)
  - Common phrases (email headers, signatures, ...)
  - Likely keys/pass-phrases (names, birthdays, etc.)

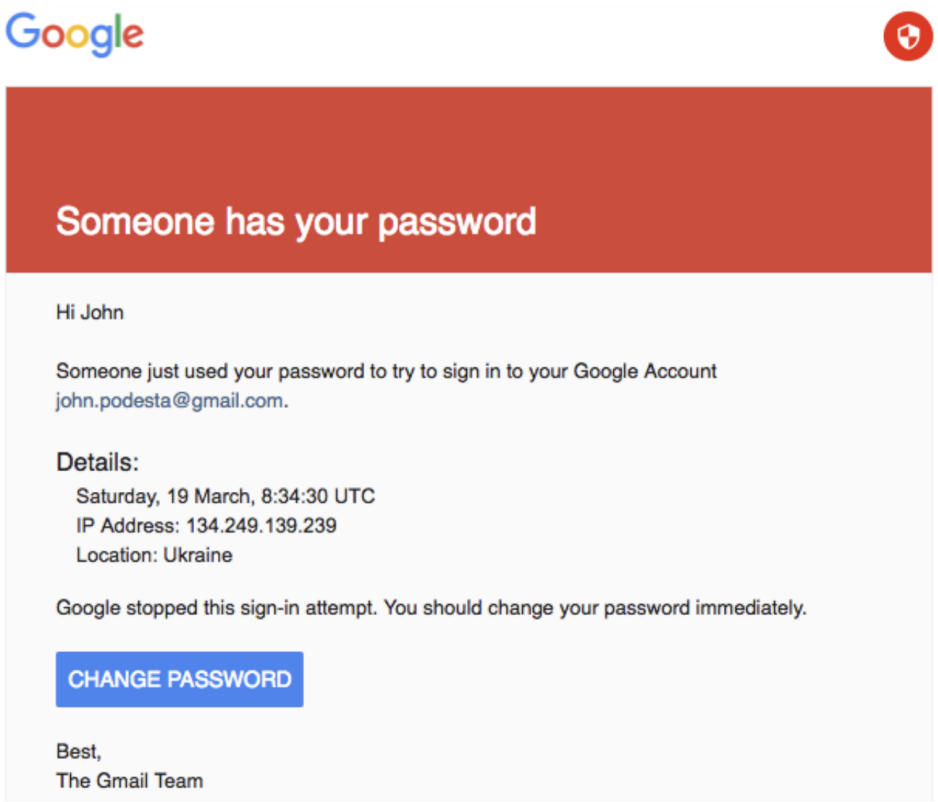
- *Adversaries may have varying levels of information, from no prior knowledge to partial or complete knowledge of the system, its algorithms, and typical user behavior.*

Threat Model : Adversary Access

- For now:
  - Access: Attacker can intercept/modify all communication content

- *How to model crypto use for confidential communication?*

Example of Threat Modeling[1]



A phishing email sent to Hillary Clinton campaign chairman John Podesta may have been so sophisticated that it fooled the campaign's own IT staffers, who at one point advised him it was a legitimate warning to change his password.[2]

[2] https://www.cnn.com/2016/10/28/politics/phishing-email-hack-john-podesta-hillary-clinton-wikileaks

[1] https://www.vox.com/policy-and-politics/2016/10/28/13456368/how-john-podesta-email-got-hacked

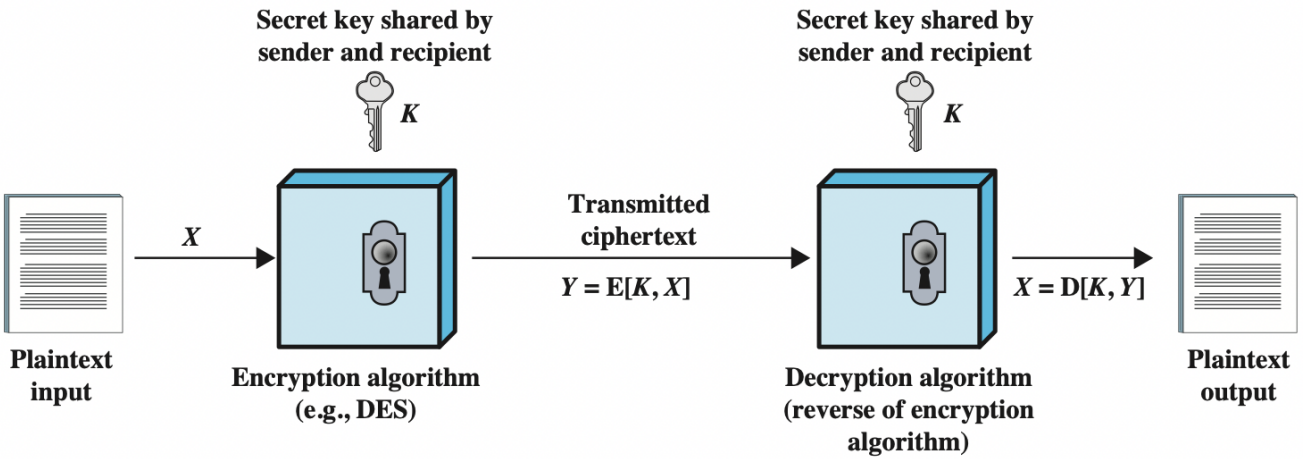
Modern Cryptography

- Symmetric cryptography
- Public key (asymmetric) cryptography.
  - We will cover it next

Symmetric Encryption

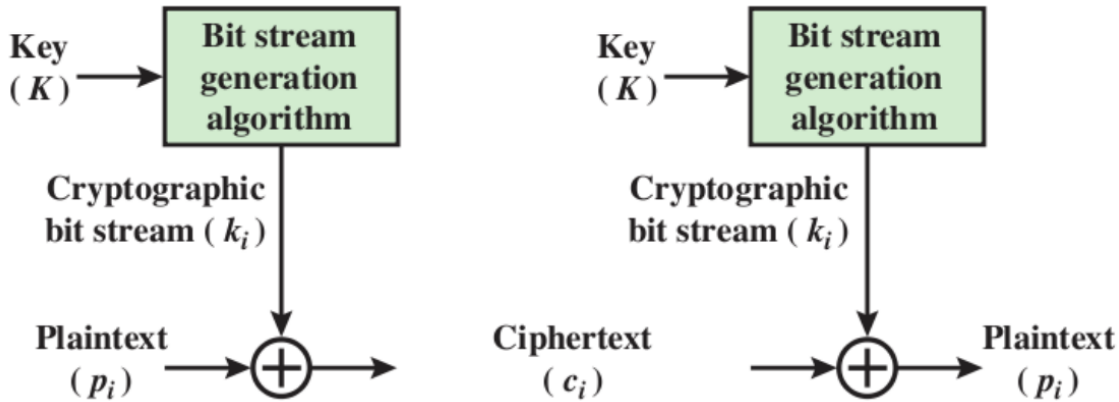
- Also referred to as:
  - Conventional encryption
  - Secret-key or single-key encryption
- Only alternative before public-key encryption in 1970's
  - Still most widely used alternative
- Two requirements for secure use:
  - Need a strong encryption algorithm
  - Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure
- Has five ingredients:
  - Plaintext
  - Encryption algorithm
  - Secret key
  - Ciphertext
  - Decryption algorithm

Simplified Model of Symmetric Encryption



Symmetric Encryption: Stream Ciphers

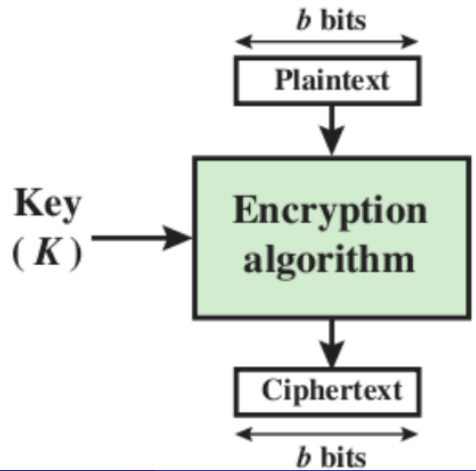
- Encrypts/operates on a digital data stream one bit or one byte at a time.
  - They are characterized by their ability to encrypt pseudorandom sequences (keystream) with corresponding bits of plaintext, typically using a bitwise operation like XOR.
- One time pad is example
- Typical approach for stream cipher:
  - Key (K) used as input to bit-stream generator algorithm
  - Algorithm generates cryptographic bit stream (k_i) used to encrypt plaintext
  - Users share a key; use it to generate keystream[1]



[1] A keystream is a sequence of pseudorandom digits which extend to the length of the plaintext in order to uniquely encrypt each character based on the corresponding digit in the keystream

Symmetric Encryption: Block Cipher

- Most common type of symmetric ciphers used for encryption.
- Encrypt a fixed-size block of plaintext as a whole to produce a block of ciphertext of the same size.
  - *maps a fixed-size input block to a fixed-size output block*
- Properties of a block cipher
  - Must supply a full block of input bits in order to evaluate
    - Partial block? Use padding
    - More than one block?
      - Modes of operation used to apply block ciphers to larger plaintexts
  - Design parameters
    - **Block size**: Bits encrypted in one application
    - **Key size**
      - $\mathcal{K}$ is the keyspace (set of possible keys)
      - Key is typically a `k-bit` binary string
      - So $|\mathcal{K}| = 2^k$
- Typical block sizes are `64` or `128` bits



Block Cipher Primitives: Confusion and Diffusion

- Claude Shannon: There are two primitive operations with which strong encryption algorithms can be built:
  1. Confusion
  2. Diffusion

Confusion

- Confusion
  - *Complexity in the relationship between plaintext, key, and ciphertext.*
  - The property of confusion hides the relationship between the ciphertext and the key.
  - Even if attacker can find some statistical characteristics of ciphertext, still hard to find key.
- Based on ciphertext, without key we cannot deduce/generate the plaintext.
- How: apply complex substitution algorithm

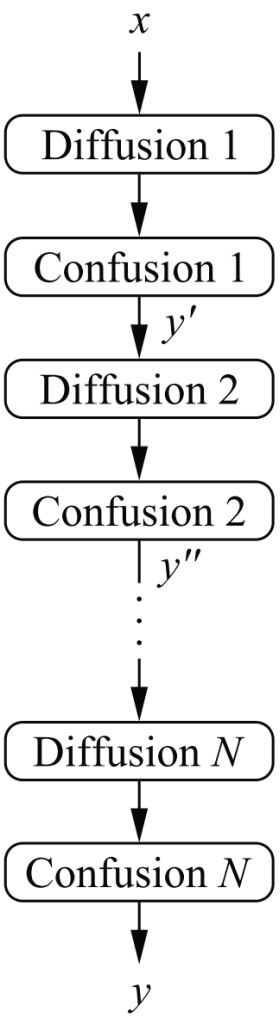
Diffusion

- Diffusion
  - *Spreading the influence of each plaintext bit over many ciphertext bits.*
  - The idea of diffusion is to hide the relationship between the ciphertext and the plaintext
  - Statistical nature of plaintext[1] is reduced in ciphertext (influence of one plaintext character is spread over many ciphertext characters)
  - e.g., a plaintext character affects the value of many ciphertext characters
- How: apply permutation (transposition) to data

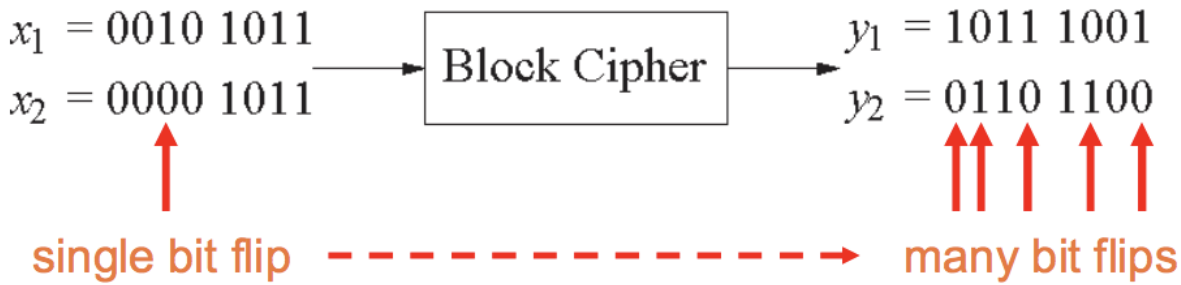
- *Both operations by themselves cannot provide security. The idea is to concatenate confusion and diffusion elements to build so called product ciphers.*

[1] *Refers to the patterns and characteristics observable within the text before encryption such as character frequency and word distribution.*

Product Ciphers



- Most of today's block ciphers are product ciphers as they consist of rounds which are applied repeatedly to the data.
- Can reach excellent diffusion: changing of one bit of plaintext results on average in the change of half the output bits.
- Example:



Composition: Building new ciphers from old

- Let (E', D') and (E'', D'') be ciphers.
- Their composition is the cipher (E, D) with keys of the form $k = (k'', k')$, where

$$E_{(k'',k')}(m) = E''_{k''}(E'_{k'}(m))$$
$$D_{(k'',k')}(c) = D'_{k'}(D''_{k''}(c))$$

Subkey generation

- When ciphers are composed, each component cipher needs a key called a subkey. Together, those subkeys can get rather large and unwieldy.
- For practical reasons, the subkeys are themselves often generated by a deterministic process dependent on a master key, which is the user key of the resulting cryptosystem.

Chaining modes

- More than one block?
- A chaining mode describes how to employ the cipher on a sequence of blocks.
- One obvious way is to repeatedly use the cipher with the same key on each successive block. This is called Electronic Code Book (ECB) mode.
  - i.e., the message is divided into blocks, and each block is encrypted separately.
- We can improve on this by generating a different subkey for each block.
- For example, successive subkeys might depend on the block number or also on previous plaintext and/or ciphertext

Feistel Structure

- Feistel Cipher/Network is not a specific scheme of block cipher. It is a design model from which many different block ciphers are derived.
- A high-level structure/design model, described by Horst Feistel of IBM in 1973, that constructs an invertible function from non-invertible components
  - Components do not need to be invertible
- A cryptographic system based on Feistel cipher structure uses the same algorithm for both encryption and decryption.
- Used in DES, RC5, and many other block ciphers; but not in AES.

Feistel Structure: Encryption

- The encryption process consisting multiple rounds of processing of the plaintext, each round consisting of a substitution step followed by a permutation step.
- The input block to each round is divided into two halves that can be denoted as `L` and `R` for the left half and the right half.
- In each round,
  - the right half of the block, `R`, goes through unchanged.
  - the left half, `L`, goes through an operation that depends on `R` and the encryption key.
  - each round uses a different key, although all these subkeys are related to the original key.
- The number of rounds is specified by the algorithm design.
- Once the last round is completed then the two sub blocks, `R` and `L` are concatenated in this order to form the ciphertext block.

Feistel Structure

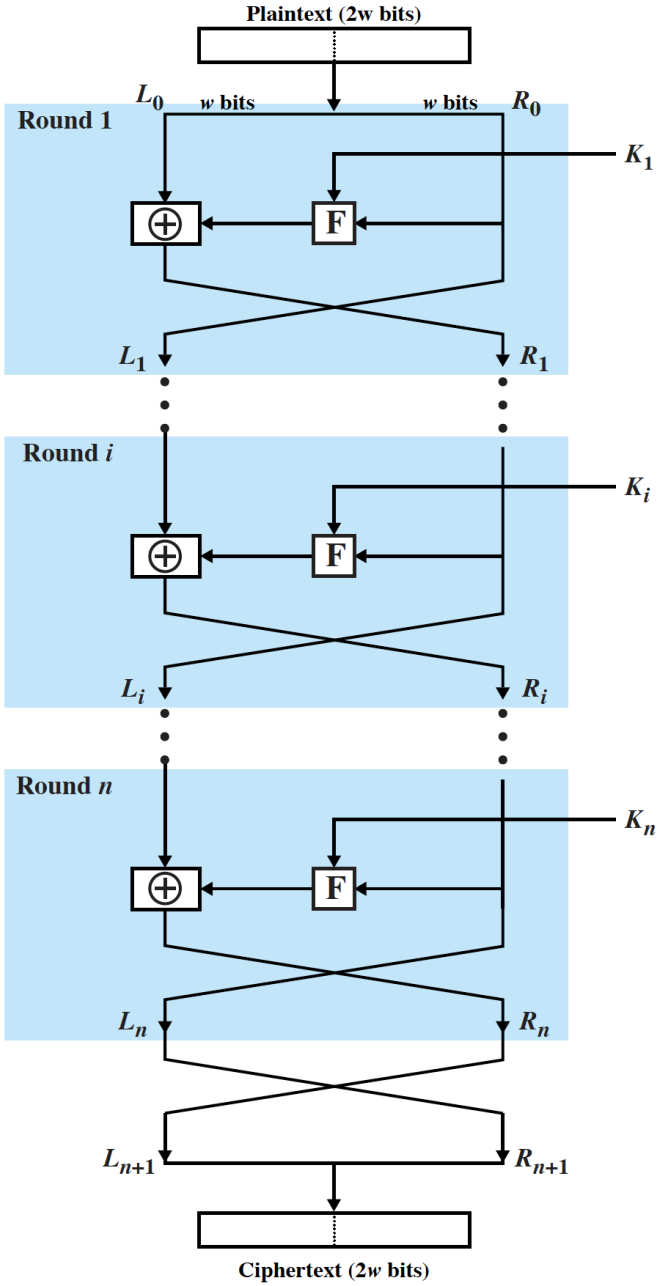
Encryption

- Input:
  - Plaintext block of length $2w$ bits
  - Key K
- Split plaintext block into left and right halves: $m = (L_0, R_0)$
- For each round $i = 1, 2, \ldots, n$, compute
  - $L_i = R_{i-1}$
  - $R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$
    where F is round function and K_i is subkey derived from K
    subkeys K_i are different from K and from each others [a]
- Ciphertext: $c = (R_n, L_n) = (L_{n+1}, R_{n+1})$

[a] K_i is a subkey, which is generally derived in some systematic way from the master key K. This means that each round uses a different key, although all these subkeys are related to the original key. F is the scrambling function.

- The number of rounds is specified by the algorithm design.
- The difficult part of designing a Feistel Cipher is selection of round function F.

Classical Feistel Structure



Feistel Structure: Encryption

- The final swapping of `L` and `R` in last step of the Feistel Cipher is essential.
- If these are not swapped, then the resulting ciphertext could not be decrypted using the same algorithm.
  - *Without final swap, in order for decryption to undo encryption, we'd have to move the exchange of Left and Right registers at the beginning of a decryption round, rather than at end in an encryption round.*
- *It makes encryption and decryption identical processes except for preparing subkeys in reverse order.*
  - This identical nature simplifies the design of hardware and software that need to accommodate both encryption and decryption functions.

Feistel Structure: Decryption

- Almost similar and not exactly same as the encryption process
  - the only difference is that the subkeys used in encryption are used in the reverse order.
- Use the ciphertext as input
- Use subkeys K_i in the reverse order
  - use K_n in the first round
  - use K_{n-1} in the second round
  - ...
  - use K_1 in the last round

Using the Feistel Structure

- Exact implementation depends on various design features:
  - Block size, e.g., `64`, `128` bits: larger values leads to more diffusion
  - Key size, e.g., `128` bits: larger values leads to more confusion, resistance against brute force
  - Number of rounds, e.g., `16` rounds
    - More number of rounds provide more secure system but inefficient slow encryption and decryption processes.
    - Number of rounds in the systems thus depend upon efficiency-security tradeoff
  - Subkey generation algorithm: should be complex. Greater complexity should lead to greater difficulty of cryptanalysis
  - Round function F: should be complex.
- Other factors include fast encryption/decryption in software

Data encryption standard (DES)

- DES is a block cipher that operates on 64-bit blocks of data.
- It uses a 56-bit key for encryption and decryption.
- DES was one of the most widely used encryption systems globally.
- Originally designed by IBM with modifications proposed by the National Security Agency (NSA)[1].
- In 1977, DES was officially adopted by the National Bureau of Standards (now NIST) as FIPS PUB 46[2].
- DES was the first modern, public, and freely available encryption algorithm.
- It is commonly referred to as the Data Encryption Algorithm (DEA).
- DES served as the United States' national standard for encryption (1977 to 2001).

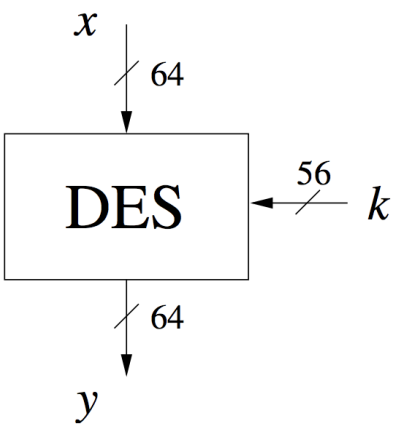
[1] *The NSA is responsible for global monitoring, collection, and processing of information and data for foreign and domestic intelligence and counterintelligence purposes, specializing in a discipline known as signals intelligence (SIGINT). The NSA is also tasked with the protection of U.S. communications networks and information systems.*

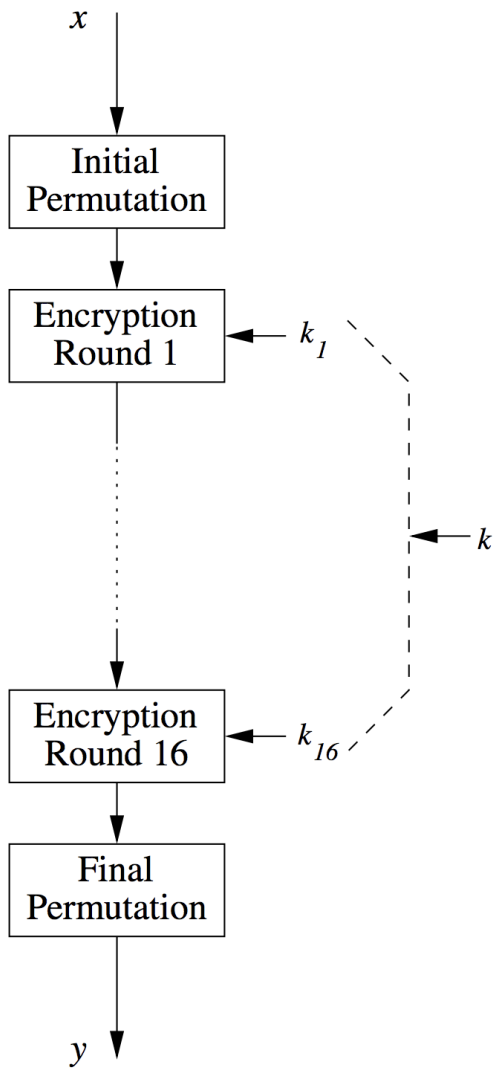
[2] *Stands for "Federal Information Processing Standards Publication." It is a series of publications that define federal standards and guidelines for various aspects of information processing, including computer security, encryption, and data communication.*

Data encryption standard (DES)

- DES's development was marked by controversy:
  - National Security Agency (NSA) secretly involved
  - Design process was secret
  - During development, the key length was reduced from 128 bits to 56 bits.
  - To delve deeper into DES's history and legacy:
    - Reading: `NSA's Role in the Development of DES`, The Legacy of DES
    - Watch: https://www.youtube.com/watch?v=jf0YzUv9wH4

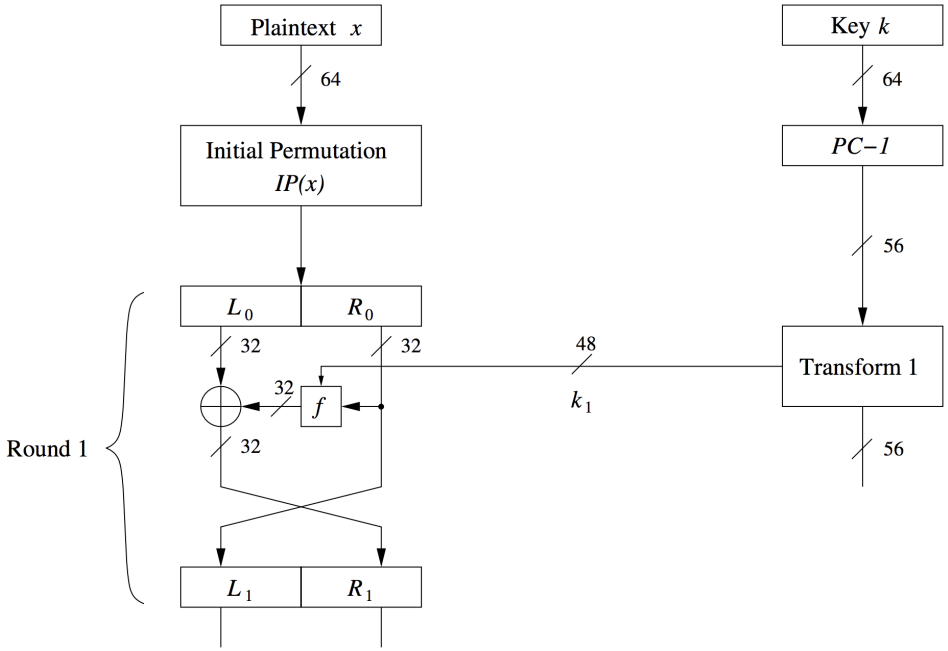
Overview of the DES Algorithm





- Encrypts blocks of size 64 bits.
- Uses a key of size 56 bits.
  - 48 bits of key used each round (subkey)
- Symmetric cipher: uses same key for encryption and decryption
- Uses 16 rounds which all perform the identical operation
- Security depends heavily on "S-boxes", each S-boxes maps 6 bits to 4 bits

DES Feistel network



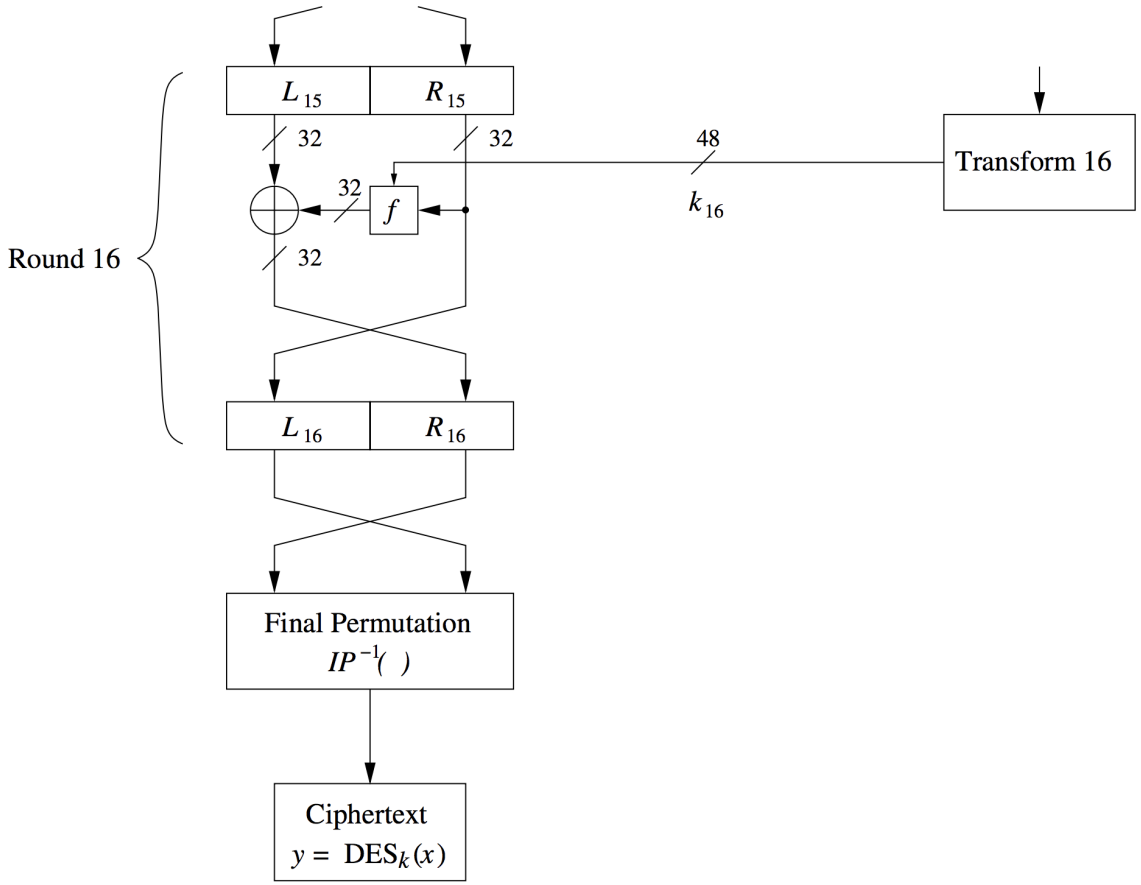
- Bitwise initial permutation, then 16 rounds

1. Plaintext is split into 32-bit halves L_i and R_i
2. R_i is fed into function f, the output of which is then XORed with L_i
3. Left and right half are swapped

- Rounds can be expressed as:
  - $L_i = R_{i-1}$
  - $R_i = L_{i-1} \oplus f(R_{i-1}, k_i)$

DES Feistel network

- L and R swapped again at the end of the cipher, i.e., after round 16 followed by a final permutation

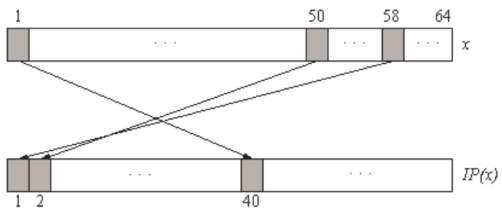


Internal Structure of DES: Initial & Final Permutation

- The initial and final permutations are straight Permutation boxes that are inverses of each other.
  - Bitwise Permutations.
  - Inverse operations.
  - Described by tables IP and IP^{-1}.
- Have no cryptographic significance but were included in order to facilitate loading blocks in and out of mid-1970s 8-bit based hardware
  - *Its primary purpose is to make it easier to load plaintext and ciphertext data into DES chip in byte-sized pieces*

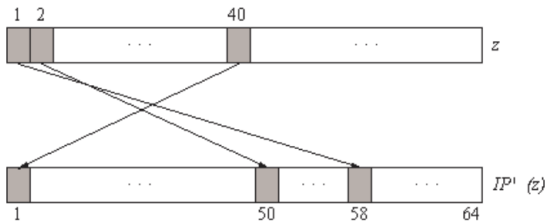
Initial Permutation

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

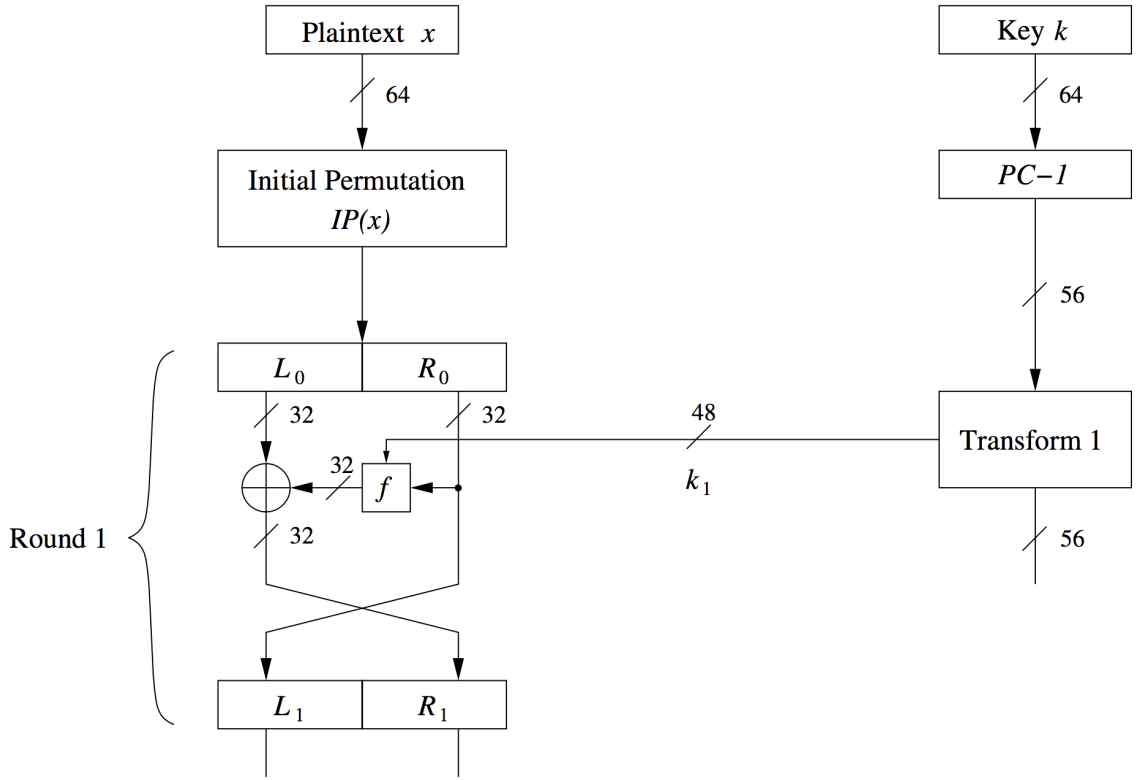


Final Permutation

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



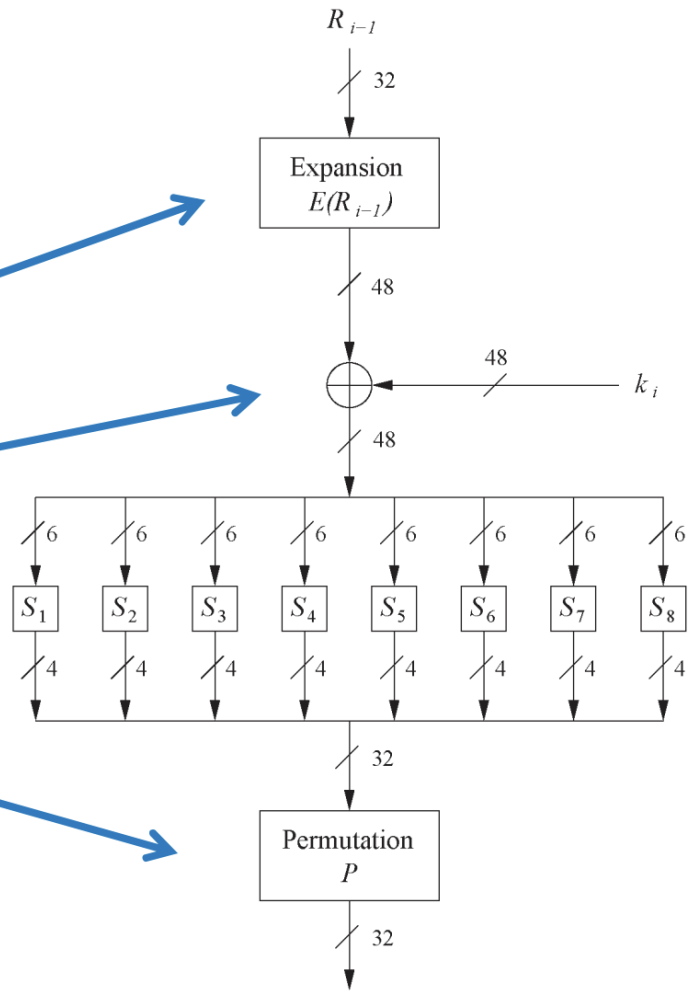
DES Feistel network



Internal Structure of DES: f-Function/Round Function

- The DES function applies a `48-bit` key to the rightmost `32-bit` to produce a `32-bit` output.

- **main operation of DES**
- *f*-Function inputs:
  R_{i-1} and round key k_i
- **4 Steps**:
  1. Expansion *E*
  2. XOR with round key
  3. S-box substitution
  4. Permutation



The f-Function: The Expansion Permutation Box E

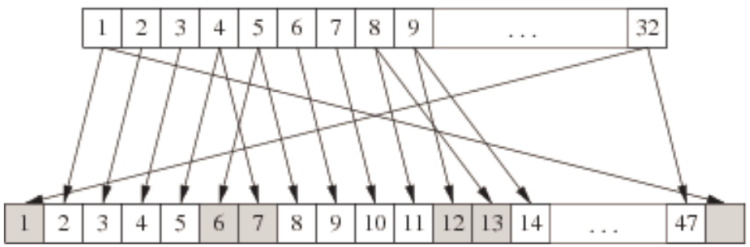
- Right input is `32-bit` and round key is a `48-bit`

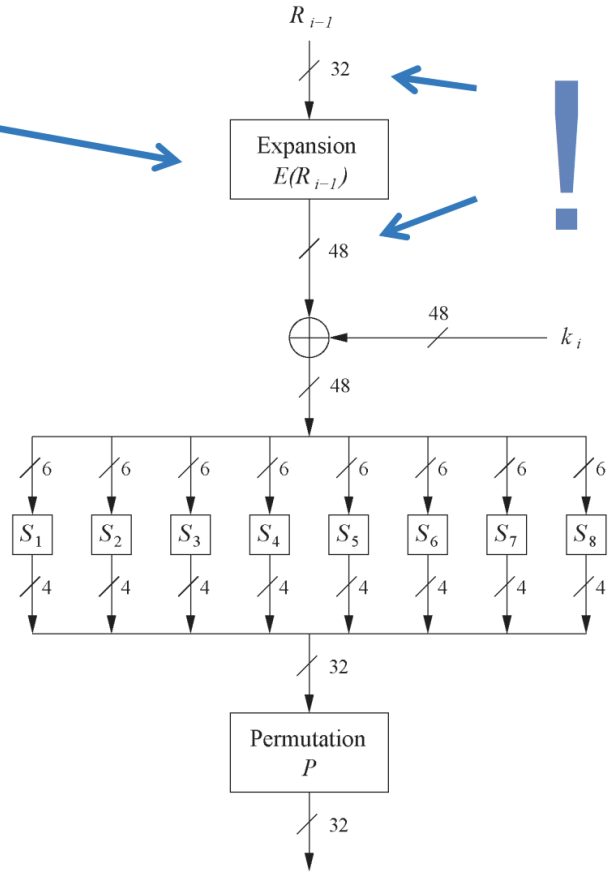
  $\rightarrow$ need to expand right input to `48` bits.

1. Expansion E

- **main purpose: increases diffusion**

E					
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



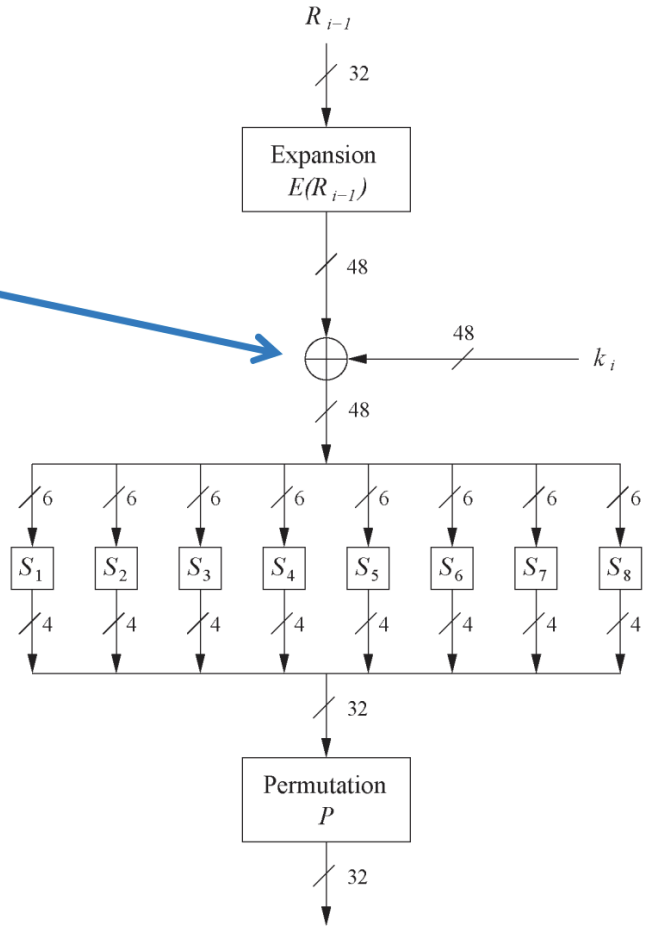


The f-Function: Add Round Key

- XOR (Whitener): XOR operation on the expanded right section and the round key.
- The round key is used only in this operation.

2. XOR Round Key

- **Bitwise XOR of the round key and the output of the expansion function *E***
- **Round keys are derived from the main key in the DES keyschedule (in a few slides)**

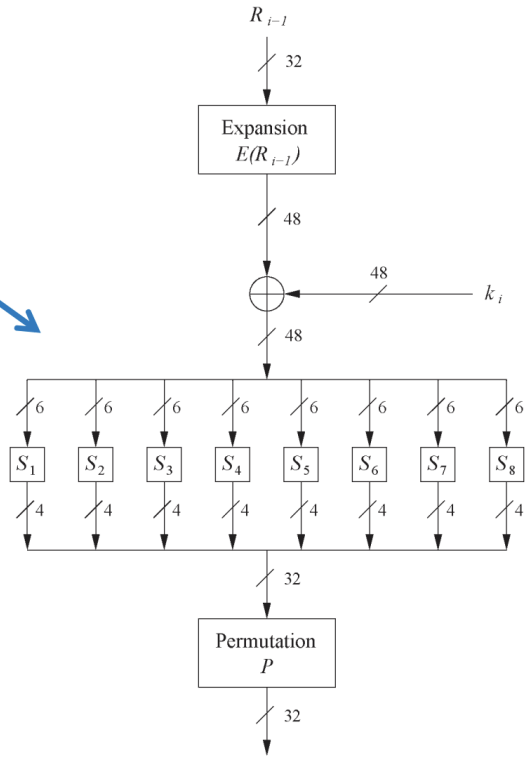


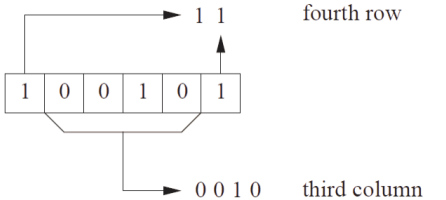
The f-Function: The DES S-Boxes

- The S-boxes carry out the real mixing (confusion).
- DES uses `8 S-boxes`, each with a `6-bit` input and a `4-bit` output

3. S-Box substitution

- Eight substitution tables.
- 6 bits of input, 4 bits of output.
- Non-linear and resistant to differential cryptanalysis.
- Crucial element for DES security!





S_1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	01	10	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

- Each row is a permutation of $\{0,1,\ldots,15\}$.
- Changing one bit in input changes at least two bits in output.

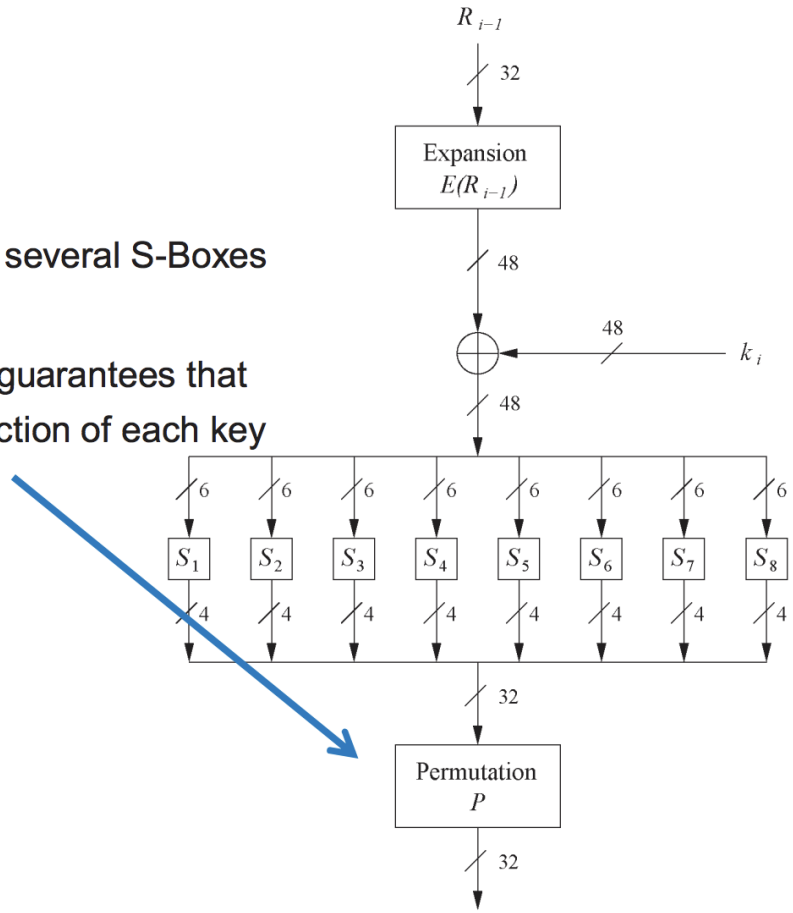
The f-Function: The Permutation P

- The `32-bit` output of S-boxes is then subjected to the Permutation `P`.
- Map each bit to an output position, no bits are used twice and no bits are ignored.

4. Permutation P

- Bitwise permutation.
- Introduces diffusion.
- Output bits of one S-Box effect several S-Boxes in next round
- Diffusion by E, S-Boxes and P guarantees that after Round 5 every bit is a function of each key bit and each plaintext bit.

P							
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



Avalanche effect

- Desirable property of any encryption algorithm.
- Small changes in either the plaintext x or the key k should give big changes (result in a significant change) in the c, and changes increase for each round.
  - A single-bit difference in each S-box results in changes in at least two bits in output
  - The mixing permutation distributes the output bits of any S-box into different S-boxes
  - The above, with sufficient number of rounds, achieves the avalanche effect.

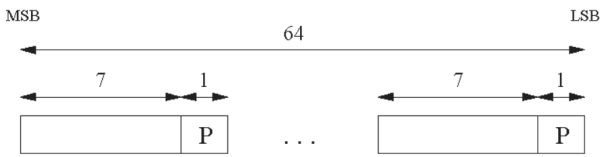
Desired Properties of Block Cipher

- The DES satisfies both the desired properties of block cipher.
- These two properties make cipher very strong.
  - Avalanche effect:
    - A small change in plaintext results in a significant and unpredictable change in the ciphertext output
    - This property ensures that the encryption process scatters the influence of each bit of the plaintext throughout the entire ciphertext, making it difficult for an attacker to discern patterns or relationships between the input and output.
  - Completeness:
    - DES also demonstrates the property of completeness, where each bit of the ciphertext depends on many bits of the plaintext.
    - This characteristic enhances the security of the encryption because altering any part of the plaintext has a widespread and non-linear impact on the resulting ciphertext.

Key Schedule: Compression Permutation/Permuted Choice

- Initially, the `64-bit` key is reduced to a `56-bit` key by ignoring every eighth bit.
  - These bits can be used as parity check to ensure the key is *`error-free`*.

- Derives 16 round keys (or *subkeys*) k_i of 48 bits each from the original 56 bit key.
- The input key size of the DES is 64 bit: **56 bit key** and 8 bit parity:



P = parity bit

- **Parity bits are removed** in a first **permuted choice** *PC-1*:
(note that the bits 8, 16, 24, 32, 40, 48, 56 and 64 are not used at all)

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

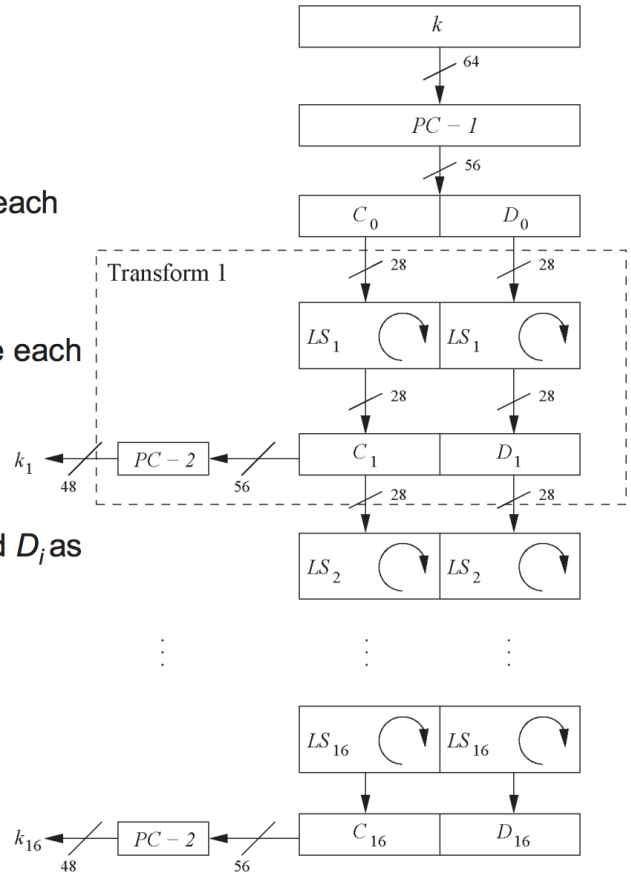
- Though, key length is `64-bit`, DES has an effective key length of `56` bits, since `8` of the `64` bits of the key are not used by the encryption algorithm

Internal Structure of DES: Key Schedule

- Split key into 28-bit halves C_0 and D_0.
- In **rounds i = 1, 2, 9 ,16,** the two halves are each rotated left by **one bit**.
- In **all other rounds** where the two halves are each rotated left by **two bits**.
- *In each round i permuted choice* ***PC-2*** selects a permuted subset of 48 bits of C_i and D_i as round key k_i,

PC – 2							
14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

- **Note:** The total number of rotations:
  4 x 1 + 12 x 2 = 28 $\Rightarrow D_0 = D_{16}$ and $C_0 = C_{16}$!



- *Because of the shifting, a different subset of key bits is used in each subkey. Each bit is used in approximately 14 of the 16 subkeys, although not all bits are used exactly the same number of times.*

DES Weak Keys[1]

- A weak key is a key, which, used with a specific cipher, makes the cipher behave in some undesirable way.
- Weak keys usually represent a very small fraction of the overall keyspace.
  - *Keys that cause the encryption mode of DES to act identically to the decryption mode of DES*
- DES has 4 weak keys (with parity bits)

```
Alternating ones + zeros (0x0101010101010101)
Alternating 'F' + 'E' (0xFEFEFEFEFEFEFEFE)
'0xE0E0E0E0F1F1F1F1'
'0x1F1F1F1F0E0E0E0E'
```

- DES weak keys produce sixteen identical subkeys.
- Using weak keys, the outcome of the Permuted Choice 1 (PC-1) leads to round keys being either all zeros, all ones or alternating zero-one patterns.
- Since all the subkeys are identical, and DES is a Feistel network, the encryption function is self-inverting; that is, despite encrypting once giving a secure-looking cipher text, encrypting twice produces the original plaintext.

[1] Credit: https://en.wikipedia.org/wiki/Weak_key

DES Weak Keys[1]

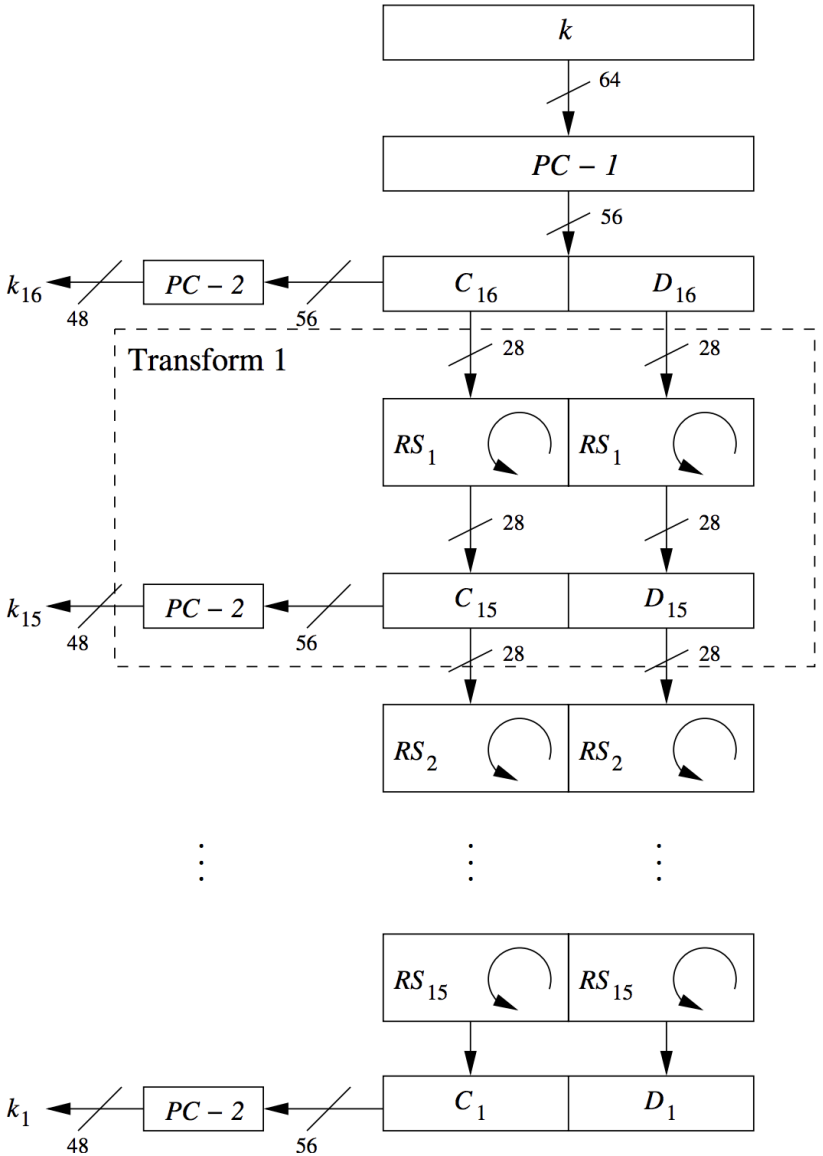
- DES also has semi-weak keys, which only produce two different subkeys, each used eight times in the algorithm.
  - Due to the way in which DES generates subkeys; instead of generating 16 different subkeys, these keys generate only two different subkeys.
  - Each of these subkeys is used eight times in the algorithm.
- Weak keys should be avoided at key generation.

```
01FE 01FE 01FE 01FE and FE01 FE01 FE01 FE01
1FE0 1FE0 0EF1 0EF1 and E01F E01F F10E F10E
01E0 01E0 01F1 01F1 and E001 E001 F101 F101
1FFE 1FFE 0EFE 0EFE and FE1F FE1F FE0E FE0E
011F 011F 010E 010E and 1F01 1F01 0E01 0E01
E0FE E0FE F1FE F1FE and FEE0 FEE0 FEF1 FEF1
```

- The third type of keys which repeat after a period of four. They produce four subkeys, each used four times in the algorithm. These are called possibly weak keys.
- There are 48 keys of this type.

[1] Credit: https://en.wikipedia.org/wiki/Weak_key

Decryption


k
64
PC – 1
56
k_{16}
48
PC – 2
56
C_{16}
D_{16}
Transform 1
28
28
RS_1
RS_1
28
28
k_{15}
48
PC – 2
56
C_{15}
D_{15}
28
28
RS_2
RS_2
RS_{15}
RS_{15}
k_1
48
PC – 2
56
C_1
D_1

Decryption

- In Feistel ciphers only the key schedule has to be modified for decryption.
- Generate the same 16 round keys in reverse order (right shift) **reversed key schedule**

DES Last Word (Almost)

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

Security of DES

- After proposal of DES two major criticisms arose:
  1. Key space is too small (2^{56} keys)
  2. S-box design criteria have been kept secret: Are there any hidden analytical attacks (backdoors), only known to the NSA?
- **Analytical Attacks**: DES is highly resistant to both differential and linear cryptanalysis, which have been published years later than the DES. This means IBM and NSA had been aware of these attacks for 15 years! So far there is no known analytical attack which breaks DES in realistic scenarios
- **Exhaustive key search**: For a given pair of plaintext-ciphertext (x, c) test all 2^{56} keys until the condition $DES_k^{-1}(c) = x$ is fulfilled.
  - $\Rightarrow$ Relatively easy given today's computer technology!
- The weakest point of DES remains the size of the key (56 bits)

Brute Force Attack

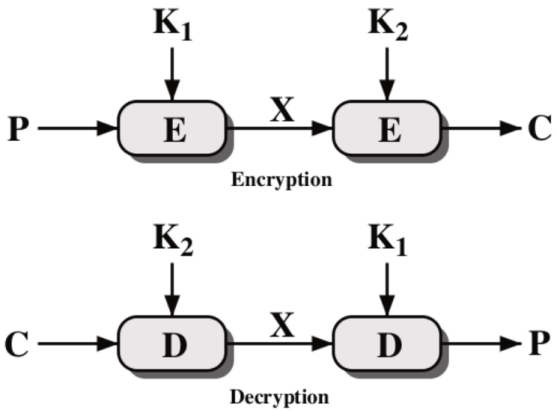
- Exhaustive search remains the most effective attack.
- What do you need?
- How many steps should it take?
  - $c = E_k(p)$
  - `56 bit` keys
  - 2^{56} possibilities
- How can you do better?

Multiple Encryption with DES

- DES is vulnerable to brute force attack
- Alternative: encrypt multiple times with different keys
- Options
  1. Double DES: not much better than single DES
  2. Triple DES (3DES) with 2 keys: brute force 2^{112}
  3. Triple DES with 3 keys: brute force 2^{168}

Double Encryption

- A plaintext P is first encrypted with a key K_1, and the resulting ciphertext is encrypted again using a second key K_2



- Assuming a key length of $k\text{-}bits$, an exhaustive key search would require $2^k \times 2^k = 2^{2k}$ encryptions or decryptions
- For DES:
  - 2×56-bit keys, meaning 112-bit key length
  - Requires 2^{112} operations for brute force
  - Can we find a better attack?

Meet-in-the-Middle Attack

- Can we search for K_1 and K_2 separately?

$$\underbrace{2^{56}}_{\text{search for } K_1} + \underbrace{2^{56}}_{\text{search for } K_2} = 2 \times 2^{56} = 2^{57}$$

- Meet-in-the-Middle Attack

Meet-in-the-Middle Attack

- Double DES Encryption: $C = E_{K_2}(E_{K_1}(P))$
- Given a pair (P, C), we have $X = E_{K_1}(P) = D_{K_2}(C)$
- Attacker knows two plaintext-ciphertext pairs (P_a, C_a) and (P_b, C_b)
  1. Encrypt P_a using all 2^{56} values of K_1 to get multiple values of X
  2. Store results in table and sort by X
  3. Decrypt C_a using all 2^{56} values of K_2
  4. As each decryption result produced, check against table
  5. If match, check current K_1,K_2 on C_b. If P_b obtained, then accept the keys
- With two known plaintext-ciphertext pairs, probability of successful attack is almost 1
- Encrypt/decrypt operations required: 2^{57} (twice as many as single DES)
  - A Meet-in-the-Middle attack requires $2^k + 2^k = 2^{k+1}$ operations!

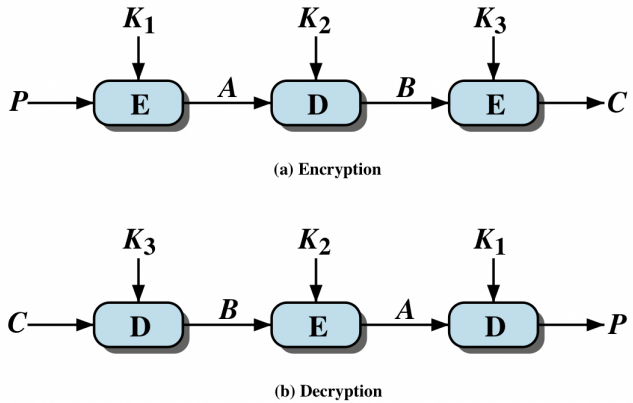
Meet-in-the-Middle Attack

- Computational Complexity
  - number of encryptions and decryptions = $2^k + 2^k = 2^{k+1}$
  - number of storage locations = 2^k
- Need to know multiple plaintext-ciphertext pairs in advance to be successful
- *Double encryption is not much more secure than single encryption!*

Triple Encryption: 3DES

- 3DES uses three keys and three executions of the DES algorithm.
- The function follows an encrypt-decrypt-encrypt (EDE) sequence:

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



- Decryption is simply the same operation with the keys reversed:

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

- 2 keys, 112 bits key length (K_3 *is replaced by* K_1)
- 3 keys, 168 bits key length
- Why E-D-E? To be compatible with single DES
- Still we can perform a meet-in-the middle attack, and it reduces the effective key length of triple encryption from 3K to 2K
  - The attacker must run 2^{112} tests in the case of 3DES
- **Triple encryption effectively doubles the key length**

- Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES)

- The algorithm for AES was chosen by the US National Institute of Standards and Technology (NIST) in a multi-year selection process
- NIST called for proposals for new standard in 1997
  - Goal: replace DES for both government and private-sector encryption.
  - The algorithm must implement symmetric key cryptography as a block cipher and (at a minimum) support block sizes of `128-bit` and key sizes of `128`, `192`, and `256-bit`.
- Candidate algorithms from around the world
- Rijndael chosen, standard called AES created in 2001
- AES:
  - Block size: 128 bits (others possible)
  - Key size: 128, 192, 256 bits
  - Rounds: 10, 12, 14 (depending on key)
  - Operations: XOR with round key, substitutions using S-Boxes, mixing using Galois Field arithmetic[1]
- AES is the most widely used symmetric cipher today
  - Widely used in file encryption, network communications

[1] *This is a special mathematical construct where addition, subtraction, multiplication, and division are redefined, and where there are a limited number of integers in the field. Also known as finite field, refers to a field in which there exists finitely many elements. Representing data as a vector in a Galois Field allows mathematical operations to scramble data easily and effectively.*

Chronology of the AES Selection

- The need for a new block cipher announced by NIST in January, 1997
- 15 candidates algorithms accepted in August, 1998
- 5 finalists announced in August, 1999:
  - Mars - IBM Corporation
  - RC6 - RSA Laboratories
  - Rijndael - J. Daemen & V. Rijmen
  - Serpent - Eli Biham et al.
  - Twofish - B. Schneier et al.
- In October 2000, Rijndael was chosen as the AES
  - issued as FIPS PUB 197 standard in Nov-2001
    http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
- AES was formally approved as a US federal standard in November 2001
- Iterated block cipher (like DES)
- Not a Feistel cipher (unlike DES)
- Designed to be efficient in both hardware and software across a variety of platforms
- No known weaknesses

AES: Overview

- Block size: `128` bits (others in Rijndael)
- Key length: `128`, `192` or `256` bits (independent of block size)
- The number of rounds depends on the chosen key length:

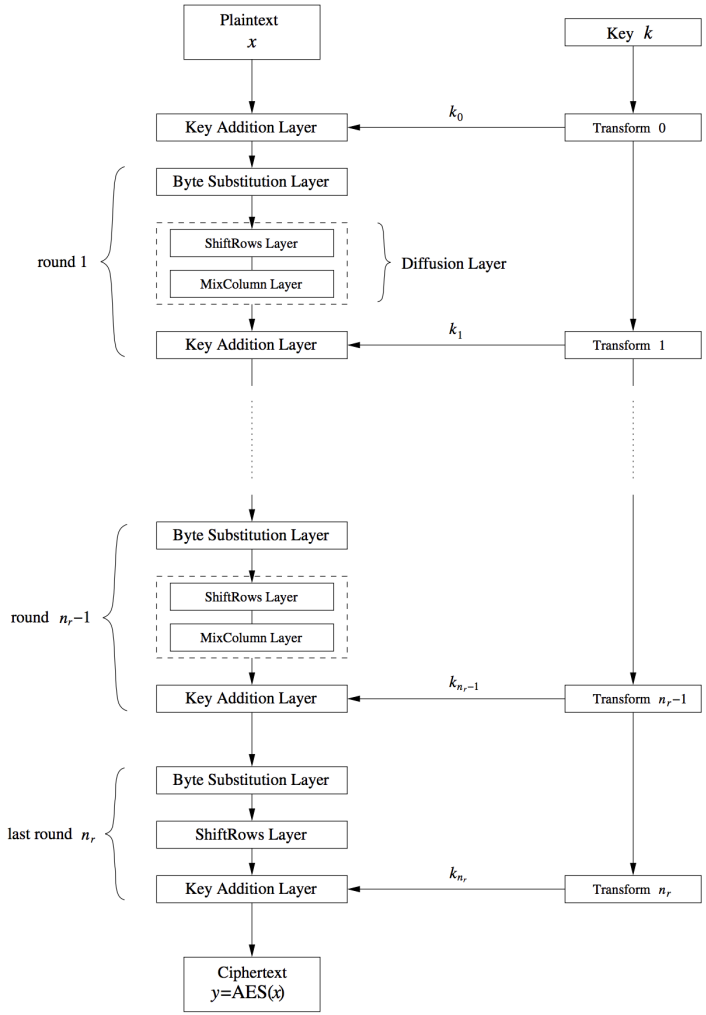
Key length (bits)	Number of rounds
128	10
192	12
256	14

  - needs `N+1` round keys for `N` rounds
- Essentially a Substitution-Permutation Network
- Each round uses `4` functions (`3` "layers")
  - ByteSub (nonlinear layer)
  - ShiftRow (linear mixing layer)
  - MixColumn (nonlinear layer)
  - AddRoundKey (key addition layer)

Overview of the AES Algorithm

- Iterated cipher with 10/12/14 rounds
- Each round consists of "Layers"

```
 1: State = X
 2:     AddRoundKey(State, k_0)
 3:     for r = 1 to n_r − 1
 4:       SubBytes(State, S-box)
 5:       ShiftRows(State)
 6:       MixColumns(State)
 7:       AddRoundKey(State, K_r)
 8:     endfor
 9: SubBytes(State, S-box)
10: ShiftRows(State)
11: AddRoundKey(State, K_{n_r})
12: Y = State
```



- State: 4 by 4 array of bytes
  - 128 bits = 16 bytes

Internal structure of AES

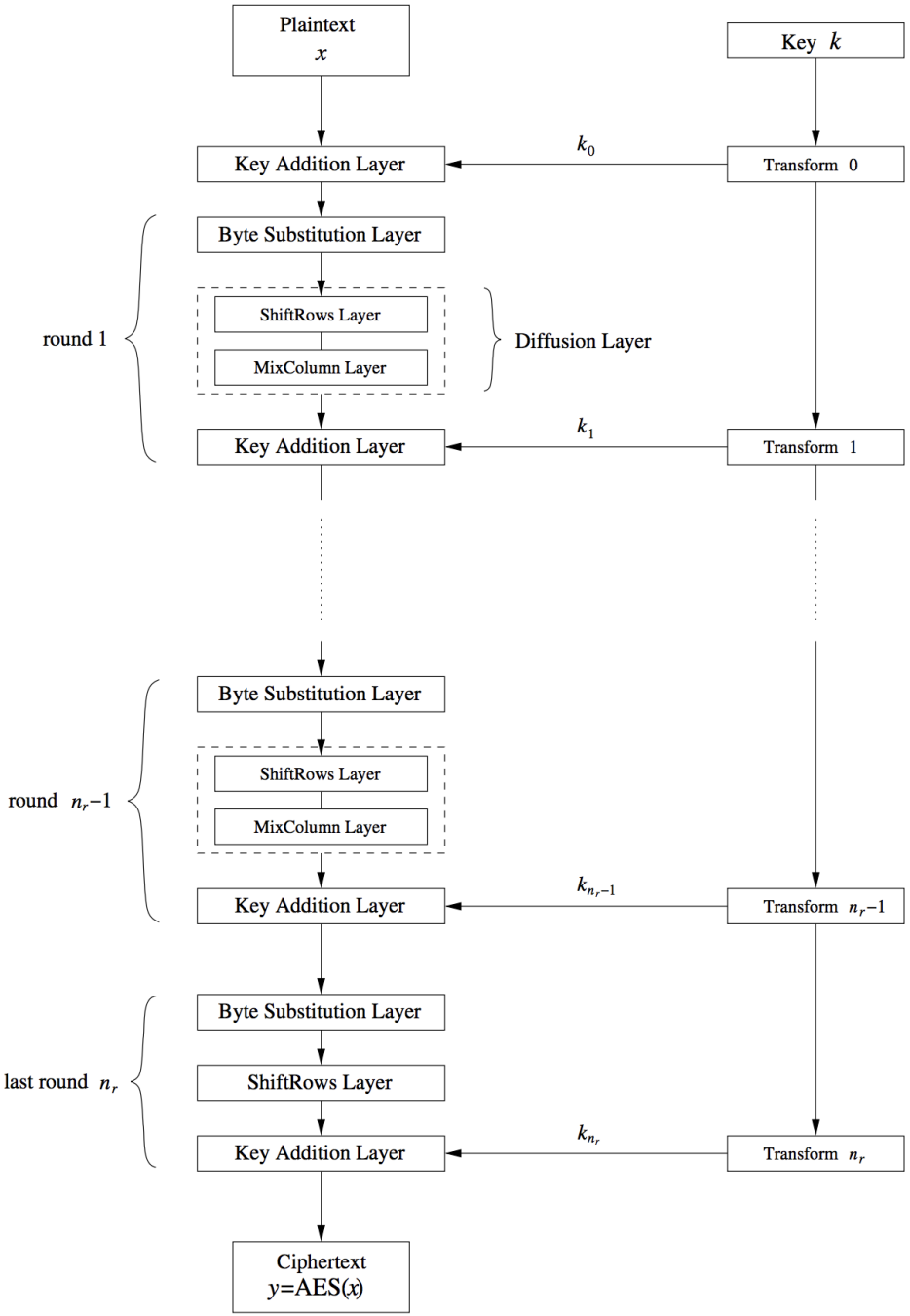
- Byte Substitution layer
- Diffusion layer
- Key Addition layer
- Key schedule

Internal structure of AES

- AES is a byte-oriented cipher
- AES treats the `128` bits of a plaintext block as `16` bytes.
- The `state A` (i.e., the `128-bit` data path) can be arranged in a `4×4` matrix:

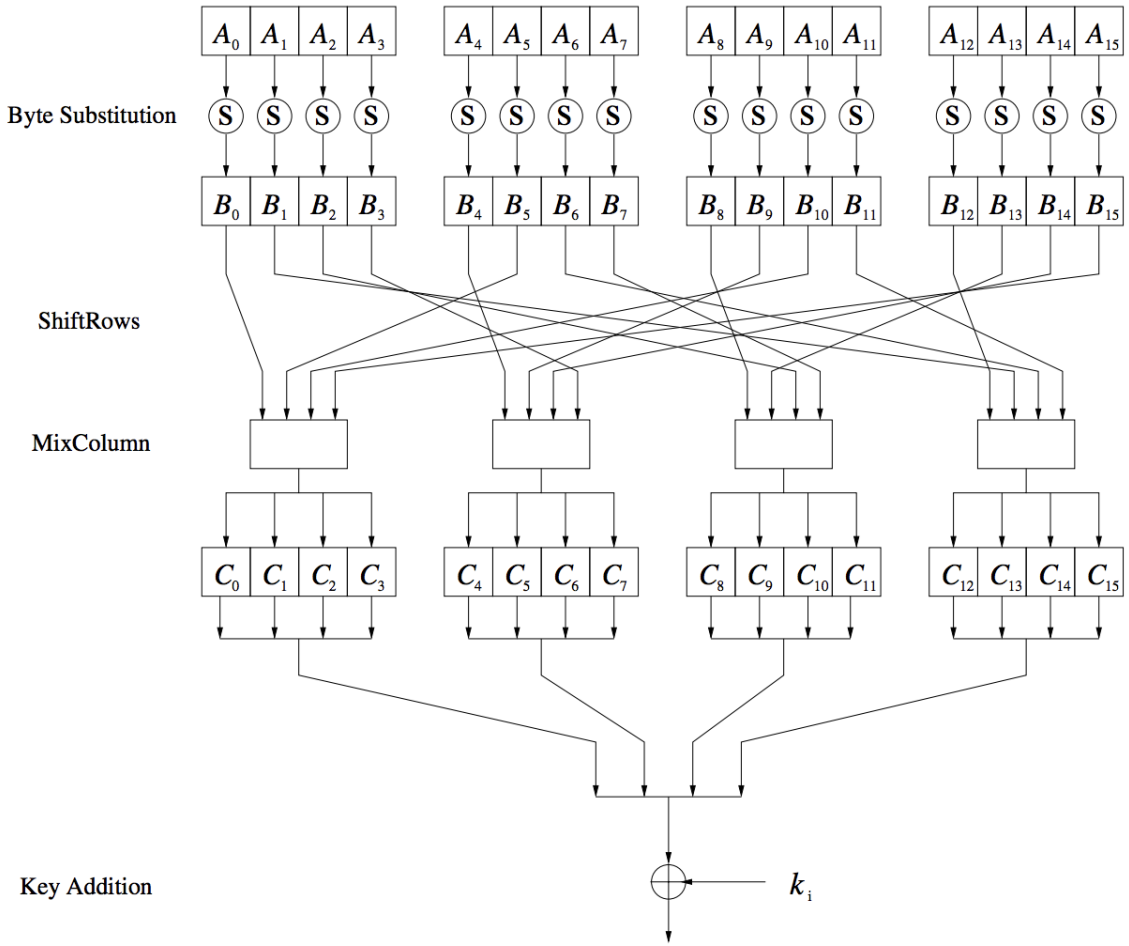
A_0	A_4	A_8	A_{12}
A_1	A_5	A_9	A_{13}
A_2	A_6	A_{10}	A_{14}
A_3	A_7	A_{11}	A_{15}

- with $A_0, \ldots, A_{15}$ denoting the *16-byte* input of AES


Plaintext
x
Key k
k_0
Key Addition Layer
Transform 0
Byte Substitution Layer
round 1
ShiftRows Layer
MixColumn Layer
Diffusion Layer
k_1
Key Addition Layer
Transform 1
Byte Substitution Layer
round n_r-1
ShiftRows Layer
MixColumn Layer
k_{n_r-1}
Key Addition Layer
Transform n_r-1
Byte Substitution Layer
last round n_r
ShiftRows Layer
k_{n_r}
Key Addition Layer
Transform n_r
Ciphertext
y=AES(x)

Internal structure of AES

- Round function for rounds $1, 2, \ldots, n_{r-1}$:



- Note: In the last round, the MixColumn transformation is omitted

Byte Substitution Layer

- Byte substitution using non-linear S-Box (independently on each byte).
- S-box is constructed using a defined transformation of the values in Galois Field[2] GF(2^8)
- The Byte Substitution layer consists of 16 S-Boxes with the following properties:
  - The S-Boxes are
    - **identical**
    - the only **nonlinear** elements of AES,
      i.e., $ByteSub(A_i)+ByteSub(A_j) \neq ByteSub(A_i+A_j)$, for $i,j=0,..,15$
    - **bijective**, i.e., there exists a one-to-one mapping of input and output bytes
      $\Rightarrow$ S-Box can be uniquely reversed
  - S-box is represented as a 16×16 array, rows and columns indexed by hexadecimal bits

[2] *Galois field is a field containing a finite number of elements uses the characteristic 2 finite field with 256 elements*

AES S-Box: Substitution values in hexadecimal notation for input byte (xy)

- The 16 input bytes are substituted by looking up a fixed table (S-box) given in design.
- The result is in a matrix of four rows and four columns.
- To read this Table, the byte input is broken into two 4-bit halves.
- The first half determines the row, and the second half determines the column.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	3	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

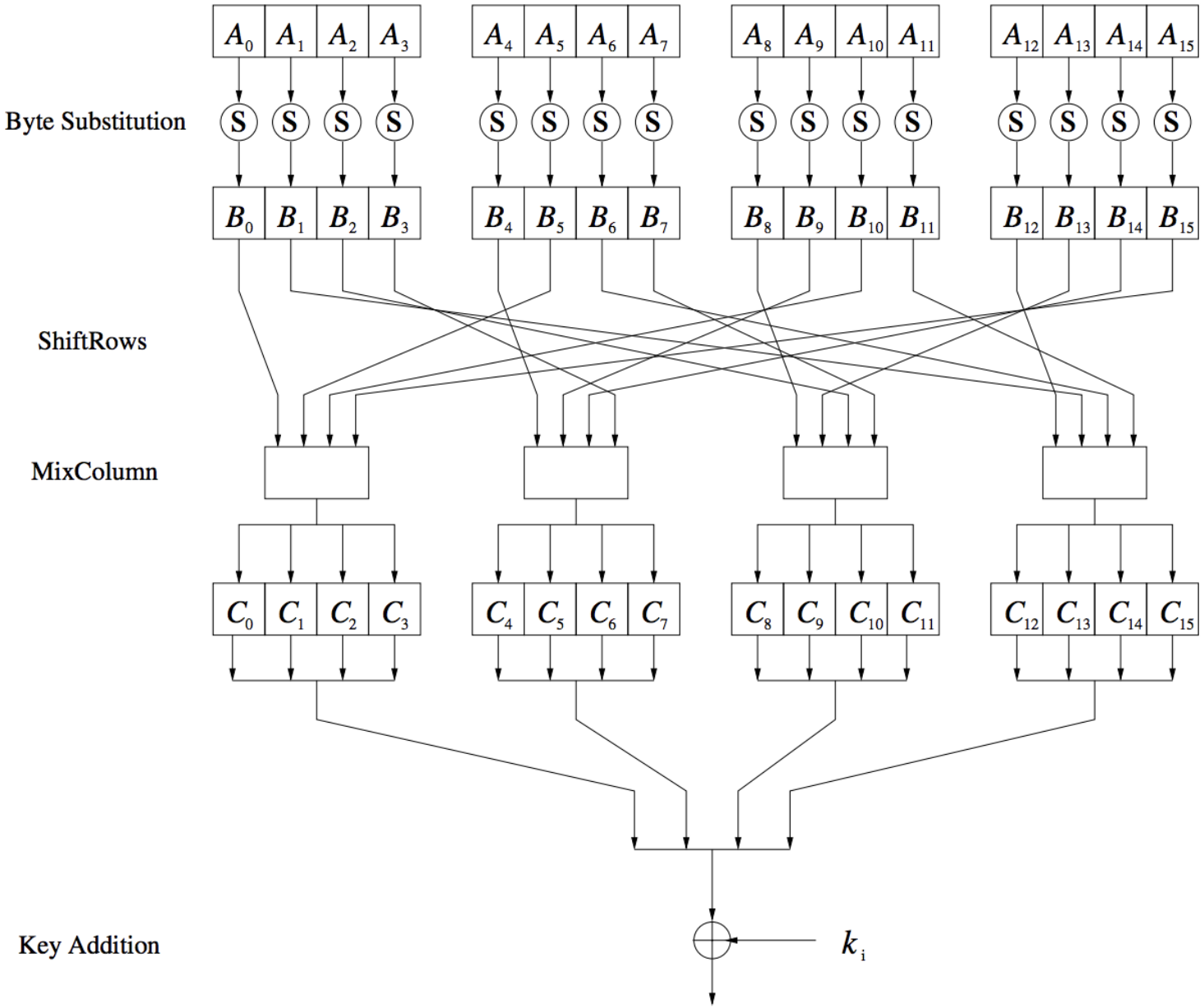
Example: hexa 53 is replaced with hexa ED

What's the point of this step?

- The byte substitution step, where each of the data points is changed according to a predetermined table, also performs an essential role.
- It alters the data in a non-linear way, in order to apply confusion[1] to the information.

[1] *Confusion is a process that helps to hide the relationship between the encrypted data and the original message.*

Internal structure of AES



Diffusion Layer

- The Diffusion layer
  - provides diffusion over all input state bits
  - consists of two sublayers:
    - ShiftRows Sublayer: Permutation of the data on a byte level
    - MixColumn Sublayer: Matrix operation which combines ("mixes") blocks of four bytes
  - performs a linear operation on state matrices A, B,
    i.e., *DIFF(A)+DIFF(B)=DIFF(A+B)*

ShiftRows Sublayer

- Rows of the state matrix are shifted cyclically:

Input matrix

B_0	B_4	B_8	B_{12}
B_1	B_5	B_9	B_{13}
B_2	B_6	B_{10}	B_{14}
B_3	B_7	B_{11}	B_{15}

  - Each of the four rows of the matrix is shifted to the left. Any entries that fall off are re-inserted on the right side of row:

Output matrix

B_0	B_4	B_8	B_{12}	no shift
B_5	B_9	B_{13}	B_1	← one position left shift
B_{10}	B_{14}	B_2	B_6	← two positions left shift
B_{15}	B_3	B_7	B_{11}	← three positions left shift

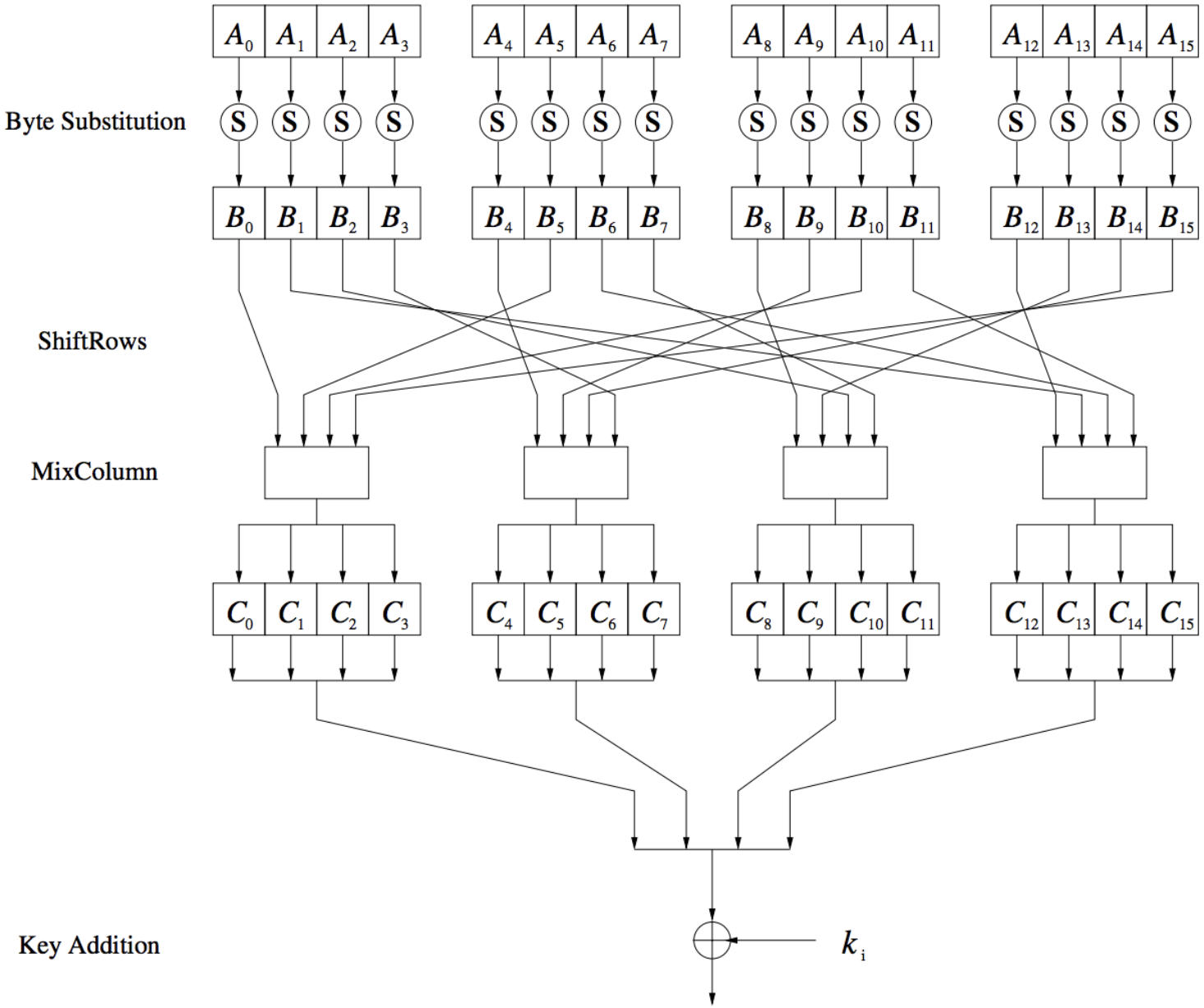
- The result is a new matrix consisting of the same `16` bytes but shifted with respect to each other.

What's the point of this step?

- Performs diffusion
- By shifting the rows, the data is moved from its original position, further helping to obscure it.

- Mix columns acts in a similar way, altering the data vertically rather than horizontally.

Internal structure of AES



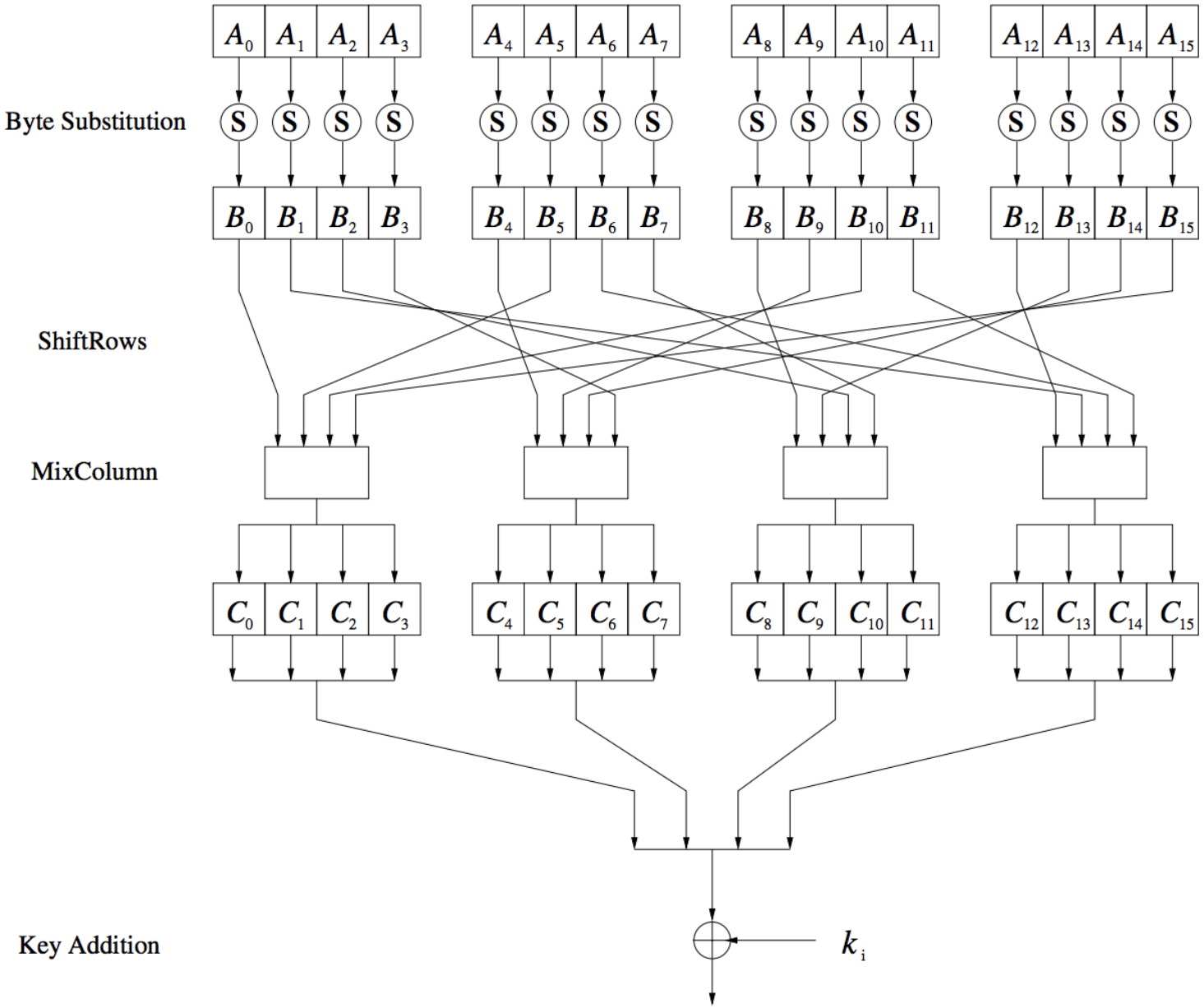
MixColumn Sublayer

- Linear transformation which mixes each column of the state matrix
- Since every input byte influences four output bytes, the MixColumn operation is the major diffusion element in AES
- Each *4-byte* column is considered as a vector and multiplied by a fixed 4×4 matrix, e.g.,

$$\begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} B_0 \\ B_5 \\ B_{10} \\ B_{15} \end{pmatrix}$$

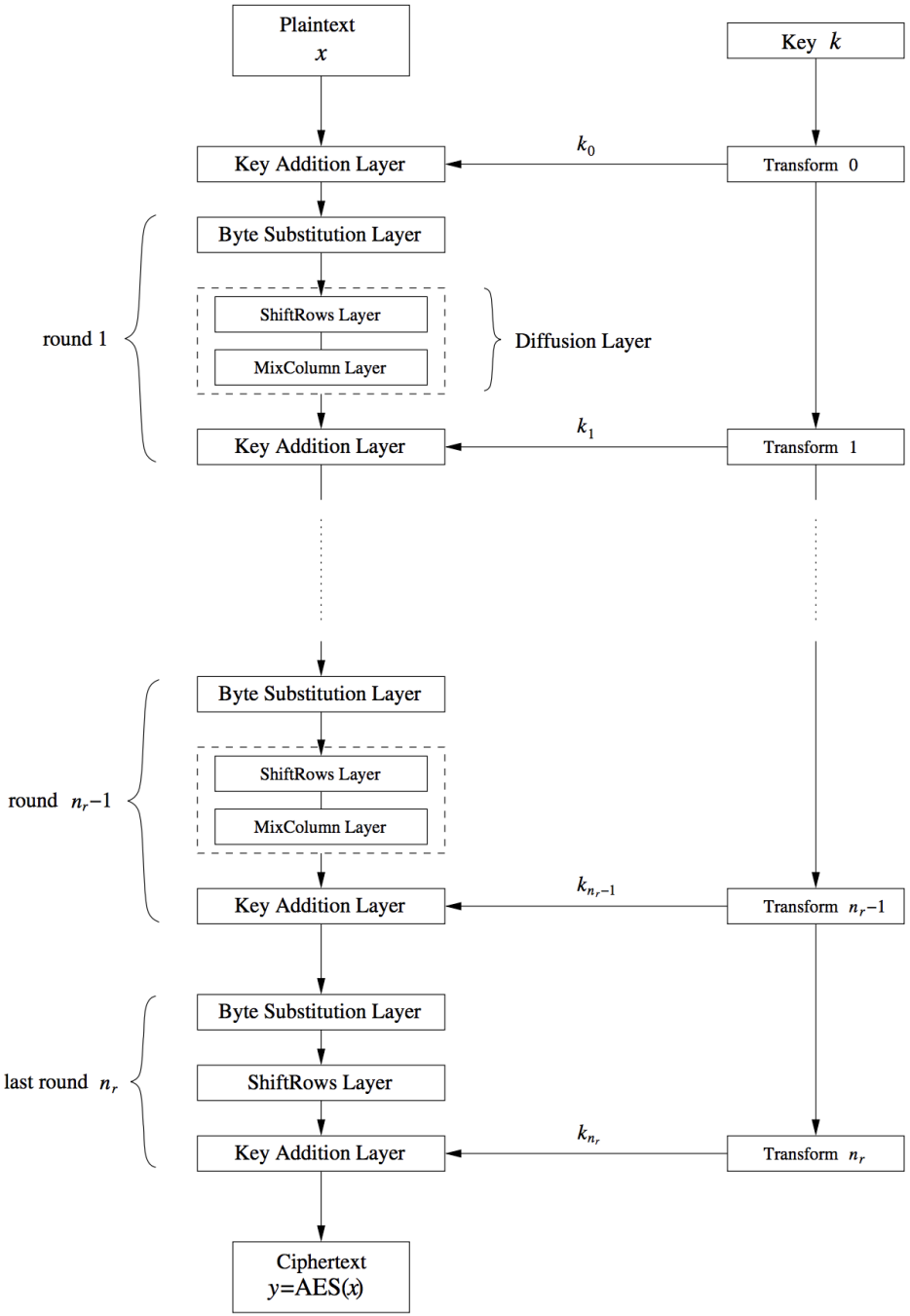
- where *01*, *02* and *03* are given in hexadecimal notation
- All arithmetic is done in the Galois field $GF(2^8)$
  - In AES the finite field contains 256 elements and is denoted as $GF(2^8)$. This field was chosen because each of the field elements can be represented by one byte
- The combination of the ShiftRows and MixColumn layer makes it possible that after only three rounds every byte of the state matrix depends on all 16 plaintext bytes

Internal structure of AES



Key Addition Layer

- The AddRoundKey operation is the only phase of AES encryption that directly operates on the AES round key.
- Inputs:
  - *16-byte* state matrix C
  - *16-byte* subkey k_i
- Output: $C \oplus k_i$
- If this is the last round, then the output is the ciphertext.
- Otherwise, the resulting 128 bits are interpreted as 16 bytes and we begin another similar round.
- The subkeys are generated in the key schedule


Plaintext
x
Key k
k_0
Key Addition Layer
Transform 0
Byte Substitution Layer
round 1
ShiftRows Layer
MixColumn Layer
Diffusion Layer
k_1
Key Addition Layer
Transform 1
Byte Substitution Layer
round n_r-1
ShiftRows Layer
MixColumn Layer
k_{n_r-1}
Key Addition Layer
Transform n_r-1
Byte Substitution Layer
last round n_r
ShiftRows Layer
k_{n_r}
Key Addition Layer
Transform n_r
Ciphertext
y=AES(x)

Key Schedule

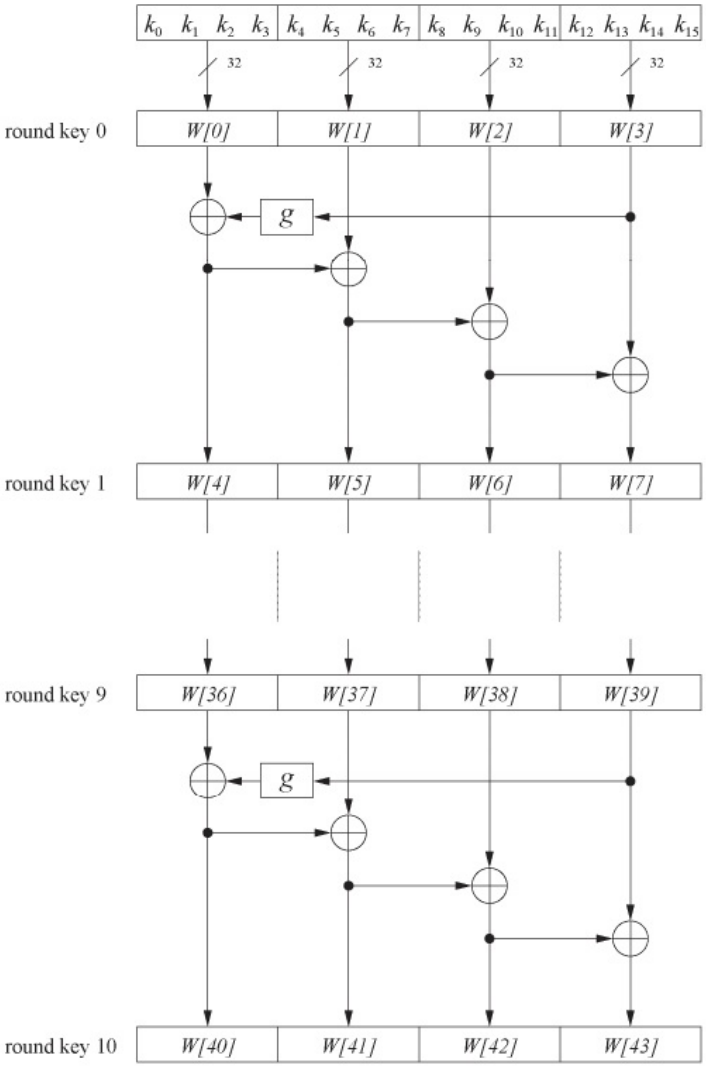
- Subkeys are derived recursively from the original *128/192/256-bit* input key
- Each round has 1 subkey, plus 1 subkey at the beginning of AES

Key length (bits)	Number of subkeys
128	11
192	13
256	15

- Note that an XOR addition of a subkey is used both at the input and output of AES. This process is sometimes referred to as Key whitening.
  - Key whitening is a technique intended to increase the security of an iterated block cipher. It consists of steps that combine the data with portions of the key.
- The number of subkeys: $\Rightarrow$ *# subkeys = # rounds + 1*
- There are different key schedules for the different key sizes

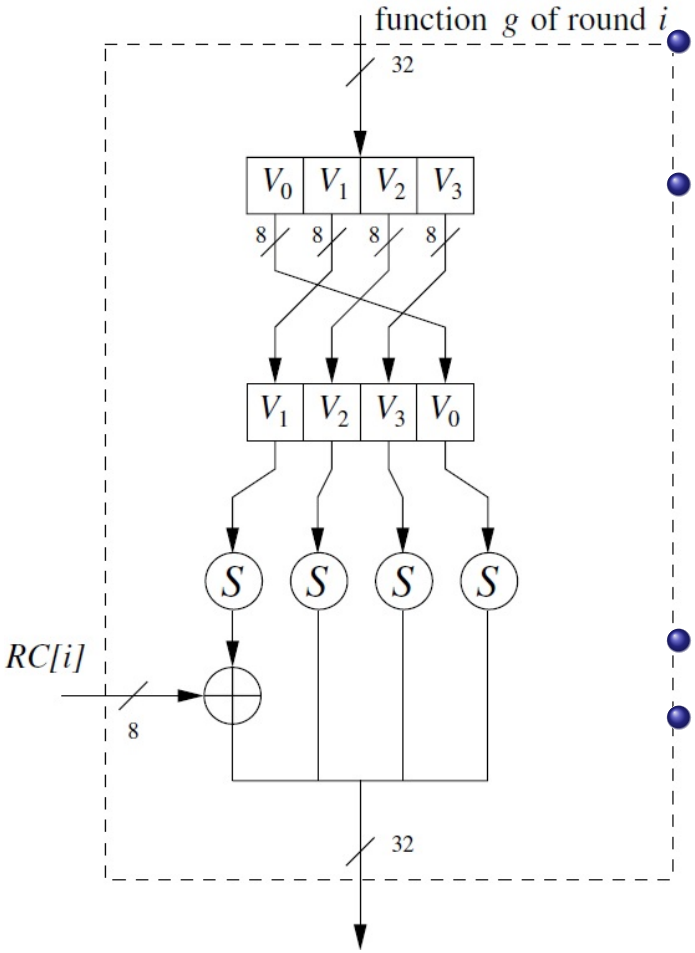
Key Schedule

Example: Key schedule for 128-bit key AES



- Word-oriented: 1 word = 32 bits
- 11 subkeys are stored in *W[0]…W[3], W[4]…W[7], … , W[40]…W[43]*
- First subkey *W[0]…W[3]* is the original AES key

Key Schedule

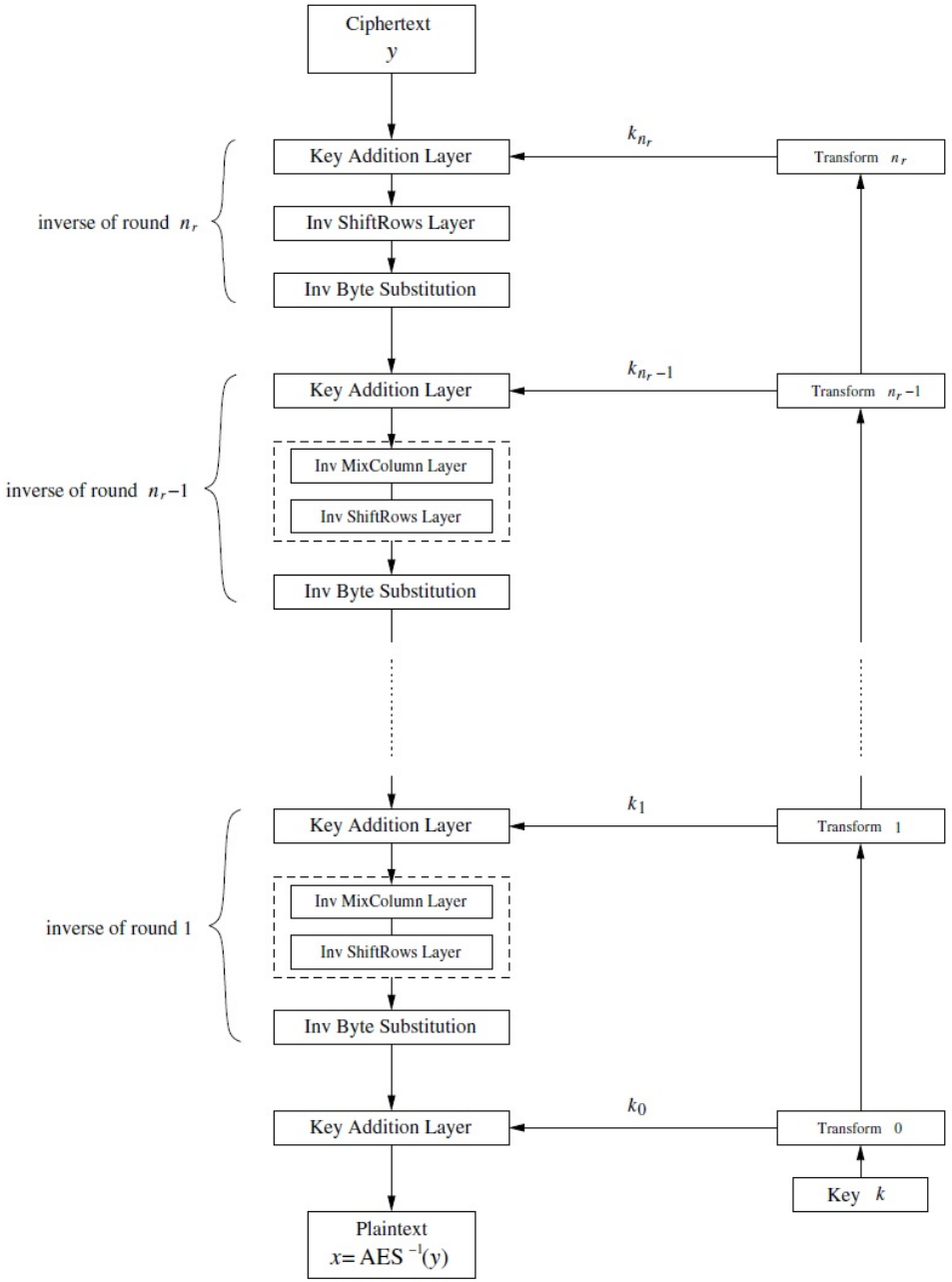


- Function g rotates its four input bytes and performs a bytewise S-Box substitution
- The *round coefficient RC* is only added to the leftmost byte and varies from round to round:

$$RC[1] = x^0 = (00000001)_2$$
$$RC[2] = x^1 = (00000010)_2$$
$$RC[3] = x^2 = (00000100)_2$$
$$\ldots$$
$$RC[10] = x^9 = (00110110)_2$$

- x^i represents an element in a Galois field $\mathtt{GF}(2^2)$
- The function g has two properties:
  - it adds nonlinearity to the key schedule
  - it removes symmetry in AES

AES Decryption



AES Decryption

- AES is not based on a Feistel network

  $\Rightarrow$ All layers must be inverted for decryption
- The process of decryption of an AES ciphertext is similar to the encryption process in the reverse order
- To decrypt, process must be invertible
  - Inverse of AddRoundKey is easy, since $\oplus$ is its own inverse
  - MixColumn is invertible.
    - The matrix used in MixColumn is invertible, so InvMixColumn uses the inverse matrix
  - Inverse of ShiftRow is easy (cyclic shift the other direction)
  - ByteSub is invertible (inverse is also implemented as a lookup table)
- Since sub-processes in each round are in reverse manner, unlike for a Feistel Cipher, the encryption and decryption algorithms need to be separately implemented.

Security

- Brute-force attack: Due to the key length of 128, 192 or 256 bits, a brute-force attack is not possible
- Analytical attacks: There is no analytical attack known that is better than brute-force
- Side-channel attacks:
  - Any attack based on information gained from the implementation of a computer system, rather than weaknesses in the implemented algorithm itself (e.g. cryptanalysis and software bugs). Timing information, power consumption, electromagnetic leaks or even sound can provide an extra source of information, which can be exploited.[1]
  - Several side-channel attacks have been published
  - Note that side-channel attacks do not attack the underlying algorithm but the implementation of it

[1] Credit:https://en.wikipedia.org/wiki/Side-channel_attack

AES Questions

- *128 vs 192 vs 256-bit AES*
  - `128-bit` AES is enough for most practical purposes
  - Either `192` or `256-bit` AES. for Highly sensitive data
  - *Cost: 192 vs 256-bit: The extra four rounds of 256-bit encryption making it about 40 percent less efficient.*
- *Why are there so many rounds?*
  - When AES was being designed, shortcut attacks were found for up to six rounds of its process.
  - Because of this, an extra four rounds were added for the minimum of `128-bit` AES as a security margin.
- The `10`, `12` and `14` rounds of AES have been settled on because they provide a good compromise between pure defensive strength, usability, and performance.

Modes of Operation

- There are several ways of encrypting long plaintexts, e.g., an e-mail or a computer file, with a block cipher (modes of operation)
- Block cipher
  - Operates on fixed length b-bit input to produce b-bit ciphertext.
  - Usually, the size of a message is larger than the block size.
- *What about encrypting plaintext longer than b bits?*
  - Break plaintext into b-bit blocks (padding if necessary) and apply cipher on each block
  - Security issues arise: different modes of operation have been developed.

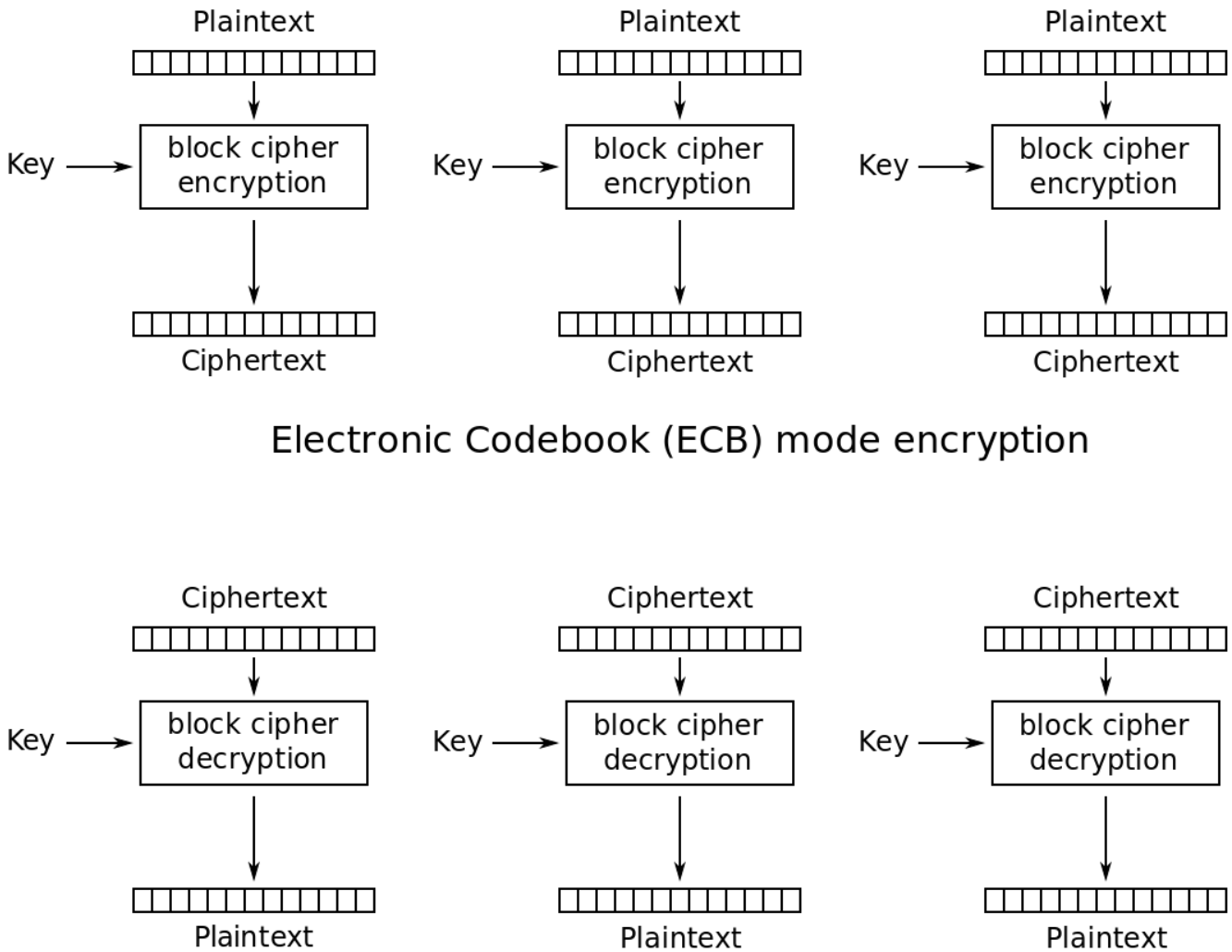
Modes of Operation

- There are several ways of encrypting long plaintexts, e.g., an e-mail or a computer file, with a block cipher ("modes of operation")
- Many modes - we will discuss 3 most popular.
- Electronic Codebook (ECB) mode
  - Encrypt each block independently.
  - Most obvious approach, but a bad idea.
- Cipher Block Chaining (CBC) mode
  - Chain the blocks together.
  - More secure than ECB, virtually no extra work.
- Counter (CTR) mode
  - Block ciphers acts like a stream cipher.
  - Popular for random access.

Electronic Codebook (ECB) mode

- Most obvious way to use a block cipher.
  - Encrypt all plaintext blocks.
  - Concatenate all resulting ciphertext blocks.
  - Output ciphertext

ECB Mode[1]



[1] Image Source Wikipedia:Block cipher mode of operation

ECB Weakness

- Problem: identical plaintext blocks produce identical ciphertext blocks
- Suppose $p_i = p_j$
  - Then $c_i = c_j$ and Eve knows $p_i = p_j$
  - This gives Eve some information, even if she does not know p_i or p_j
  - Eve might know p_i
- Q: Is this a serious issue?
  - Because ECB encrypts identical plaintext blocks into identical ciphertext blocks, it does not hide data patterns well.
  - In some senses, it doesn't provide serious message confidentiality, and it is not recommended for use in cryptographic protocols at all.

Alice hates ECB mode

Alice's unencrypted image, Alice ECB encrypted

- Q: Why does it happen? Same plaintext yields same ciphertext!
- Solution: Insert information about block's position into the plaintext block, then encrypt

ECB: advantages/disadvantages

- Advantages
  - Bit errors caused by noisy channels only affect the corresponding block but not succeeding blocks
  - Block cipher operating can be parallelized
    - advantage for high-speed implementations
- Disadvantages
  - ECB encrypts entirely deterministic:.
    - identical plaintexts result in identical ciphertexts
      - an attacker recognizes if the same message has been sent twice
    - plaintext blocks are encrypted independently of previous blocks
      - an attacker may reorder ciphertext blocks which results in valid plaintext

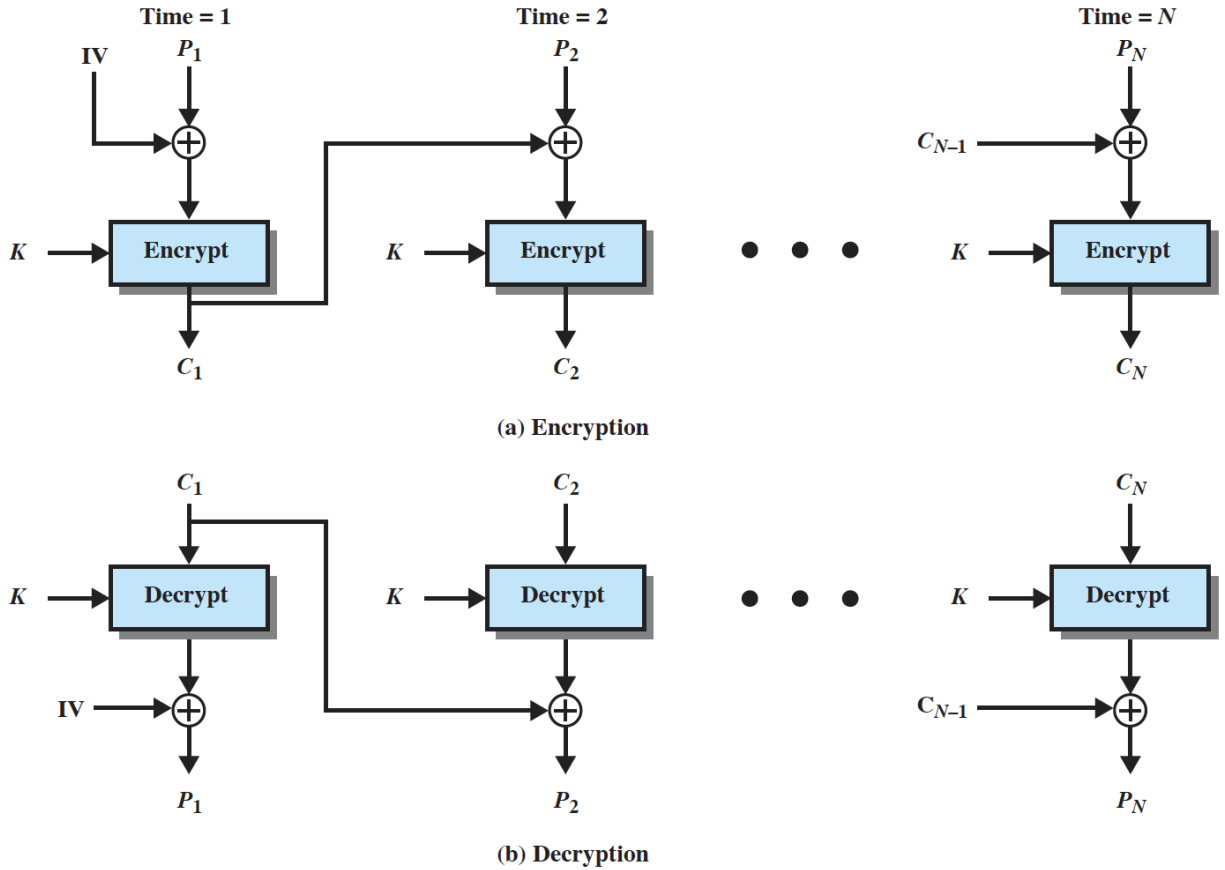
Cipher Block Chaining (CBC) mode

- We want to solve two problems
  1. Make encryption probabilistic
  2. Combine encryption of all blocks
- An encryption scheme is "deterministic" if a particular plaintext is mapped to a fixed ciphertext if the key is unchanged
- A "probabilistic" encryption scheme uses randomness to achieve a non-deterministic generation of p_i

Cipher Block Chaining (CBC) mode

- Blocks are "chained" together in a special way that introduces dependency between them.
  - The current plaintext block is added to the previous ciphertext block, and then the result is encrypted with the key.
  - Decryption is the reverse process, which involves decrypting the current ciphertext and then adding the previous ciphertext block to the result.
- A random initialization vector, or IV, is required to initialize CBC mode.
  - Nothing to chain the first block with.
  - IV is random, but not secret

CBC Mode



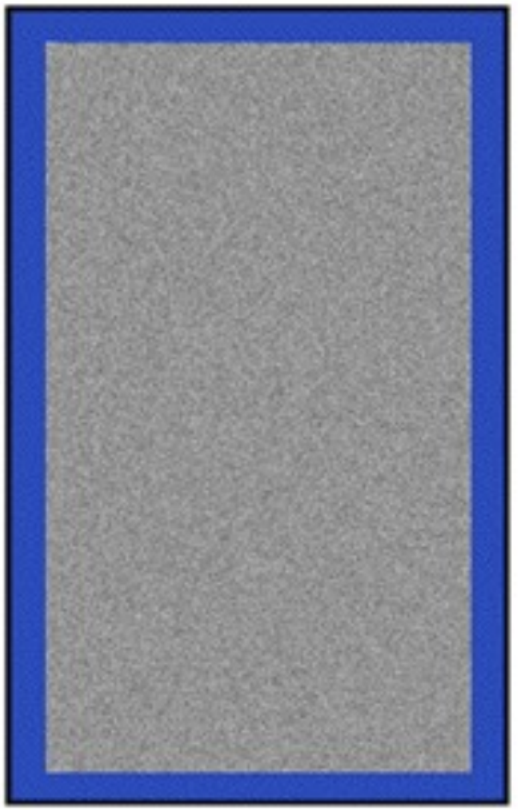
- **Encryption**
  - $C_1 = E_K(P_1 \oplus IV)$
  - $C_i = E_K(P_i \oplus C_{i-1})$, for $i \geq 2$
- **Decryption**
  - $P_1 = D_K(C_1) \oplus IV$
  - $P_i = D_K(C_i) \oplus C_{i-1}$, for $i \geq 2$

CBC Mode

- Identical plaintext blocks yield different ciphertext blocks - this is very good!
- But what about errors in transmission?
  - If C_j is garbled to, say, G then
    - $P_j \neq D_K(G) \oplus C_{j-1}$
    - $P_{j+1} \neq D_K(C_{j+1}) \oplus G$
  - But
    - $P_{j+2} = D_K(C_{j+2}) \oplus C_{j+1}$
    - $P_{j+3} = D_K(C_{j+3}) \oplus C_{j+2}$
    - $\ldots$
  - Automatically recovers from errors!
  - *One damaged block propagates to two blocks.*
- Cut and paste is still possible, but more complex (and will cause garbles)

Alice likes CBC mode

Alice's unencrypted image, Alice CBC encrypted



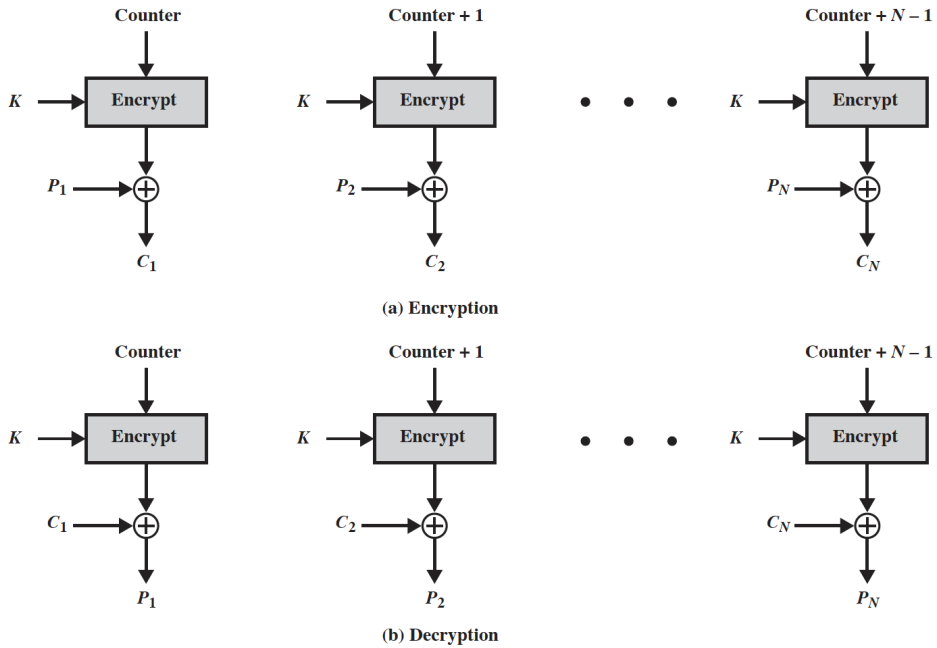
- Q: Why does it happen? Same plaintext yields different ciphertext!

IV: Initial Vector

- Does not have to be secret
- Should be a non-secret nonce (value used only once)

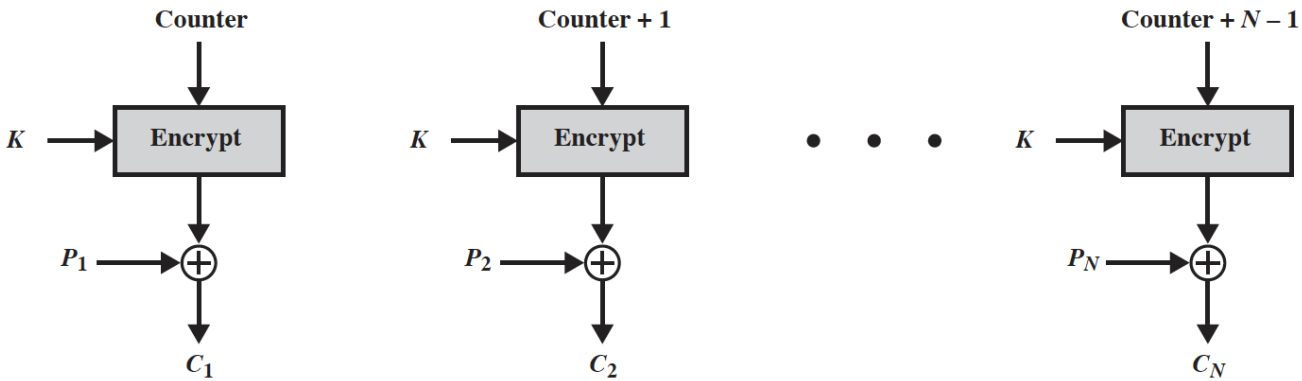
Counter (CTR) Mode

- Use block cipher like a stream cipher.
  - i.e., use the block cipher as a key stream generator
- The key stream is computed in a block wise fashion
- The input to the block cipher is a counter which assumes a different value every time the block cipher computes a new key stream block

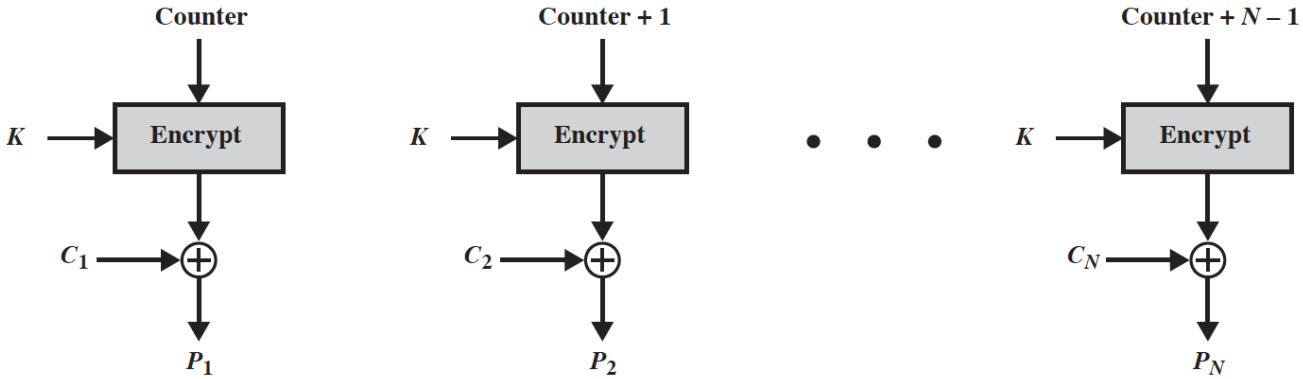


- CTR is popular for random access.
- *No error propagation in case of loss or damage.*

Counter (CTR) Mode



(a) Encryption



(b) Decryption

- **Encryption**
  - $C_i = E_k(CTR_i) \oplus P_i$
- **Decryption**
  - $P_i = E_k(CTR_i) \oplus C_i$

Key Distribution

- The means of delivering a key to two parties that wish to exchange data without allowing others to see the key
- Two parties (`A` and `B`) can achieve this by:
  1. A key could be selected by `A` and physically delivered to `B`
  2. A third party could select the key and physically deliver it to `A` and `B`
  3. If `A` and `B` have previously and recently used a key, one party could transmit the new key to the other, encrypted using the old key
  4. If `A` and `B` each have an encrypted connection to a third party `C`, `C` could deliver a key on the encrypted links to `A` and `B`

Summary

- Symmetric encryption principles
  - Cryptography
  - Cryptanalysis
  - Feistel cipher structure
- Data encryption standard
  - Data encryption standard
  - Triple DES
- Advanced encryption standard
  - Overview of the algorithm
  - Algorithm details
- Cipher block modes of operation
  - Electronic codebook mode
  - Cipher block chaining mode
  - Counter mode
- Key distribution

Reading

- Computer Security: Principles and Practice
  - Chapter 2 and chapter 20
- Understanding Cryptography: A Textbook for Students and Practitioners
  - Chapter 3: The Data Encryption Standard (DES) and Alternatives.