

CSE 4351: Computer Security



Instructor: Dr. MD Armanuzzaman

Symmetric Encryption II

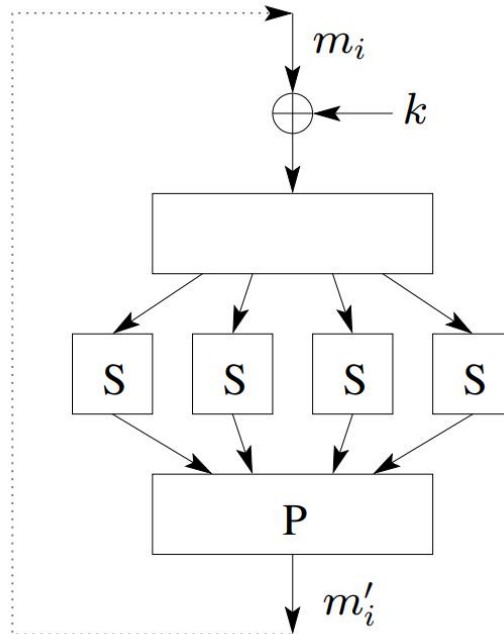


Design Principles of Block Ciphers

- Confusion-diffusion paradigm
 - a. split a block into small chunks
 - b. define a substitution on each chunk separately (confusion)
 - c. mix outputs from different chunks by rearranging bits (diffusion)
 - d. repeat to strengthen the result

Design Principles of Block Ciphers

- For this type of algorithm to be reversible, each operation needs to be invertible



Design Principles of Block Ciphers

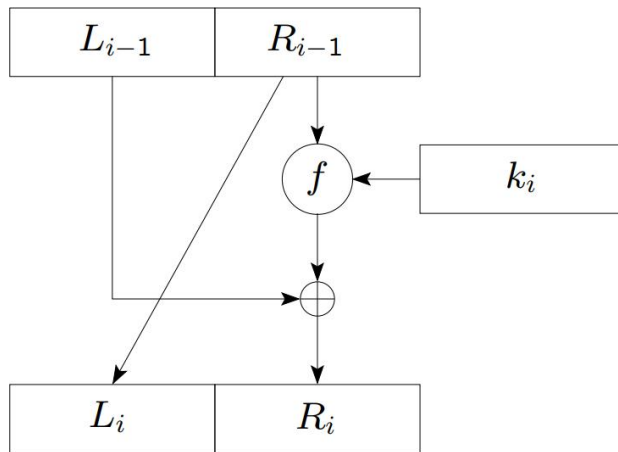
- Let's denote one iteration or round by function g
 - The initial state s_0 is the message m itself
 - In round i :
 - g 's input is round key k_i and state s_{i-1}
 - g 's output is state s_i
 - The ciphertext c is the final state s_{Nr} , where Nr is the number of rounds
 - Decryption algorithm applies g^{-1} iteratively
 - the order of round keys is reversed
 - set $s_{Nr} = c$, compute $s_{i-1} = g^{-1}(k_i, s_i)$

Design Principles of Block Ciphers

- Another way to realize confusion-diffusion paradigm is through **Feistel** network
 - in Feistel network each state is divided into halves of the same length: L_i and R_i
 - in one round:
 - $L_i = R_{i-1}$
 - $R_i = L_{i-1} \oplus f(k_i, R_{i-1})$

Design Principles of Block Ciphers

- Are there any advantages over the previous design?
 - operations no longer need to be reversible, as the inverse of the algorithm is not used!
 - reverse one round's computation as $R_{i-1} = L_i$ and $L_{i-1} = R_i \oplus f(k_i, R_{i-1})$



Why it matters: SPN vs Feistel Invertibility

Can the round operation avoid being invertible on its own?

SPN: the S-box MUST be invertible

Good S-box (bijective)

In	Out
00	01
01	10
10	00
11	11

Every output maps
back to exactly
one input.

Bad S-box (lazy / collision)

In	Out
00	01
01	01
10	00
11	11

00 and 01 both map
to 01 — collision!

Ciphertext "01" could have come from
plaintext 00 OR 01 — decryption is stuck.

Feistel: f can be non-invertible

Let $f(k_i, R) = 00$ always (a useless, non-invertible f)

Encrypt one round: $L_0=10, R_0=01$, key k_1

$$L_1 = R_0 = 01$$

$$R_1 = L_0 \oplus f(k_1, R_0) = 10 \oplus 00 = 10$$

→ Ciphertext: $L_1R_1 = 0110$

Decrypt (only $L_1=01, R_1=10, k_1$ known):

$$R_0 = L_1 = 01$$

$$\begin{aligned} L_0 &= R_1 \oplus f(k_1, R_0) = 10 \oplus f(k_1, 01) \\ &= 10 \oplus 00 = 10 \end{aligned}$$

→ Recovered $L_0=10, R_0=01$ — correct!

f was never "undone" — it was just
recomputed forward and XORed away.

Key takeaway: XOR is self-cancelling ($X \oplus Y \oplus Y = X$) — that's the real source of invertibility, not f itself.

Design Principles of Block Ciphers

- In both types of networks, the substitution and permutation algorithms must be carefully designed
 - choosing random substitution/permutation strategies leads to significantly weaker ciphers
 - each bit difference in S-box input creates at least 2-bit difference in its output
 - mixing permutation ensures that difference in one S-box propagates to at least 2 S-boxes in next round

Block Ciphers

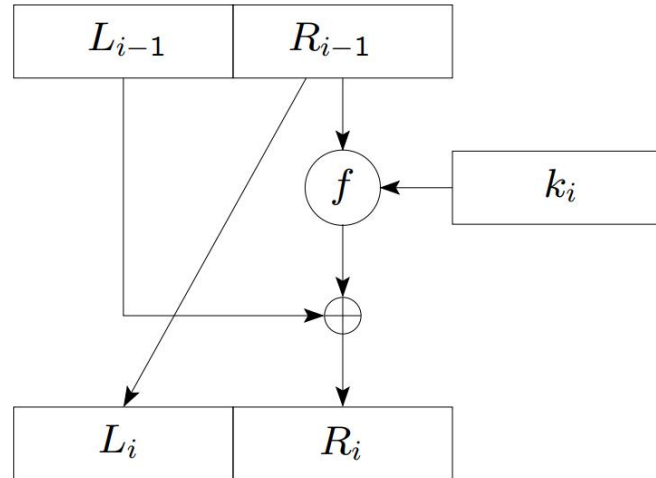
- Larger key size means greater security
 - for n-bit keys, brute force search takes $2^{(n/2)}$ time on average
 - More rounds often provide better protection
 - the number of rounds must be large enough for proper mixing
 - Larger block size offers increased security
 - security of a cipher also depends on the block length

Data Encryption Standard (DES)

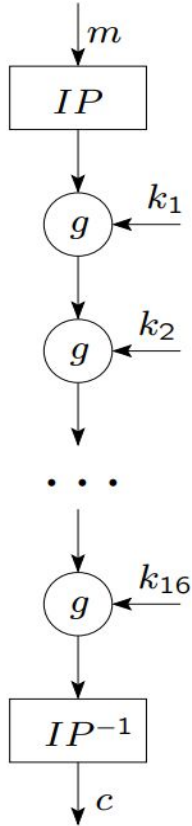
- In 1973 National Institute of Standards and Technology (NIST) published a solicitation for cryptosystems
- DES was developed by IBM and adopted as a standard in 1977
- It was expected to be used as a standard for 10–15 years
- Was replaced only in 2001 with AES (Advanced Encryption Standard)
- DES characteristics:
 - key size is 56 bits
 - block size is 64 bits
 - number of rounds is 16

Data Encryption Standard (DES)

- DES uses Feistel network
 - Feistel network is used in many block ciphers such as DES, RC5, etc.
 - not used in AES
 - in DES, each L_i and R_i is 32 bits long; k_i is 48 bits long

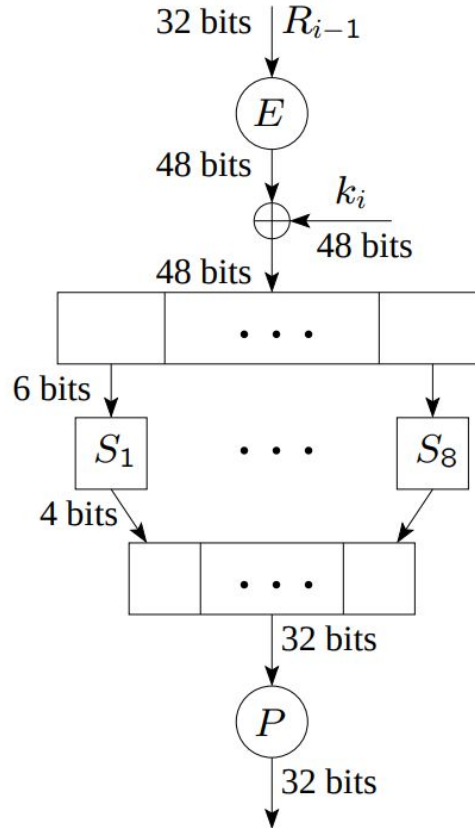


Data Encryption Standard (DES)



- DES has a fixed initial permutation IP prior to 16 rounds of encryption
 - The inverse permutation IP^{-1} is applied at the end

DES f function



- The f function $f(k_i, R_{i-1})$
 - first expands R_{i-1} from 32 to 48 bits (k_i is 48 bits long)
 - XORs expanded R_{i-1} with k_i
 - applies substitution to the result using S-boxes
 - and finally permutes the value

DES

- There are 8 S-boxes
 - S-boxes are the only non-linear elements in DES design
 - they are crucial for the security of the cipher
- Example S1

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

input to each S-box is 6 bits $b_1b_2b_3b_4b_5b_6$

- row = b_1b_6 , column = $b_2b_3b_4b_5$
- output is 4 bits

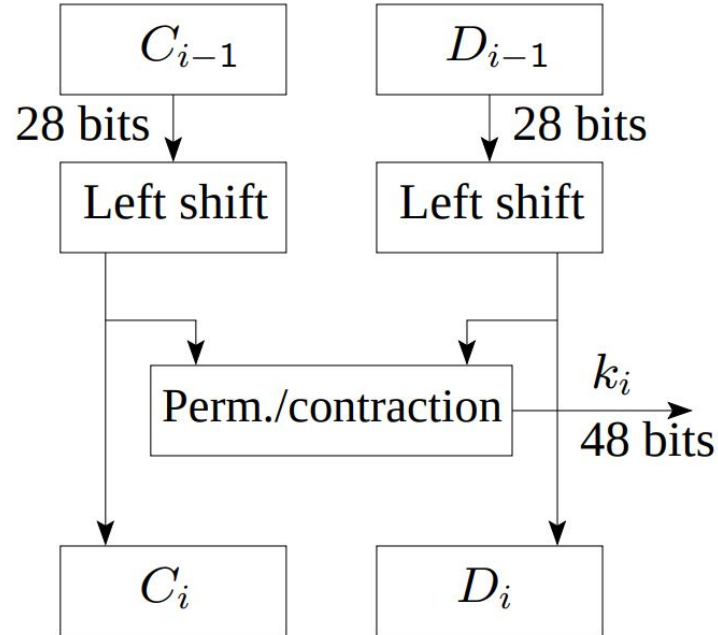
DES

More about S-boxes..

- a modified version of IBM's proposal was accepted as the standard
- some of the design choices of S-boxes weren't public, which triggered criticism
- in late 1980s – early 1990s differential cryptanalysis techniques were discovered
- it was then revealed that DES S-boxes were designed to prevent such attacks
- such cryptanalysis techniques were known almost 20 years before they were discovered by others

DES Key Schedule

- Key computation consists of:
 - circular shift
 - permutation
 - contraction



DES Weak Keys

- The master key k is used to generate 16 round keys
- Some keys result in the same round key to be generated in more than one round
 - this reduces complexity of the cipher
- Solution: check for weak keys at key generation
- DES has 4 weak keys:
 - 00000000 00000000
 - 00000000 FFFFFFFF
 - FFFFFFFF 00000000
 - FFFFFFFF FFFFFFFF

Attacks on DES

- Brute force attack: try all possible 2^{56} keys
 - time-consuming, but no storage requirements
- Differential cryptanalysis: traces the difference of two messages through each round of the algorithm
 - was discovered in early 90s
 - not effective against DES
- Linear cryptanalysis: tries to find linear approximations to describe DES transformations
 - was discovered in 1993
 - has no practical implication

Brute Force Search Attacks on DES

- It was conjectured in 1970s that a cracker machine could be built for \$20 million
- In 1990s RSA Laboratories called several DES challenges
 - Challenge II-2 was solved in 1998 by Electronic Frontier Foundation
 - a DES Cracker machine was built for less than \$250,000 and found the key was in 56 hours
 - Challenge III was solved in 1999 by the DES Cracker in cooperation with a worldwide network of 100,000 computers
 - the key was found in 22 hours 15 minutes
 - <http://www.distributed.net/des>

Increasing Security of DES

- DES uses a 56-bit key and this raised concerns
- One proposed solution is double DES
 - apply DES twice by using two different keys k_1 and k_2
 - encryption $c = E_{k_2}(E_{k_1}(m))$
 - decryption $m = D_{k_1}(D_{k_2}(c))$
- The resulting key is $2 \cdot 56 = 112$ bits, so it should be more secure, right?
 - an attack called meet-in-the-middle discovers keys k_1 and k_2 with 2^{56} computation and storage
 - better, but not substantially than regular DES

✚ 2^{112}

Triple DES

- Triple DES with two keys k_1 and k_2 :
 - encryption $c = E_{k_1}(D_{k_2}(E_{k_1}(m)))$
 - decryption $m = D_{k_1}(E_{k_2}(D_{k_1}(c)))$
 - key space is $2 \cdot 56 = 112$ bits
- Triple DES with three keys k_1 , k_2 , and k_3 :
 - encryption $c = E_{k_3}(D_{k_2}(E_{k_1}(m)))$
 - decryption $m = D_{k_1}(E_{k_2}(D_{k_3}(c)))$
 - key space is $3 \cdot 56 = 168$ bits
- There is no known practical attack against either version
- Can be made backward compatible by setting $k_1 = k_2$ or $k_3 = k_2$

Summary of Attacks on DES

- DES – best attack: brute force search
 - 2^{55} work on average
 - no other requirements
- Double DES
 - best attack: meet-in-the-middle
 - requires 2 plaintext-ciphertext pairs
 - requires 2^{56} space and about 2^{56} work
- Triple DES
 - best practical attack: brute force search

Symmetric Encryption

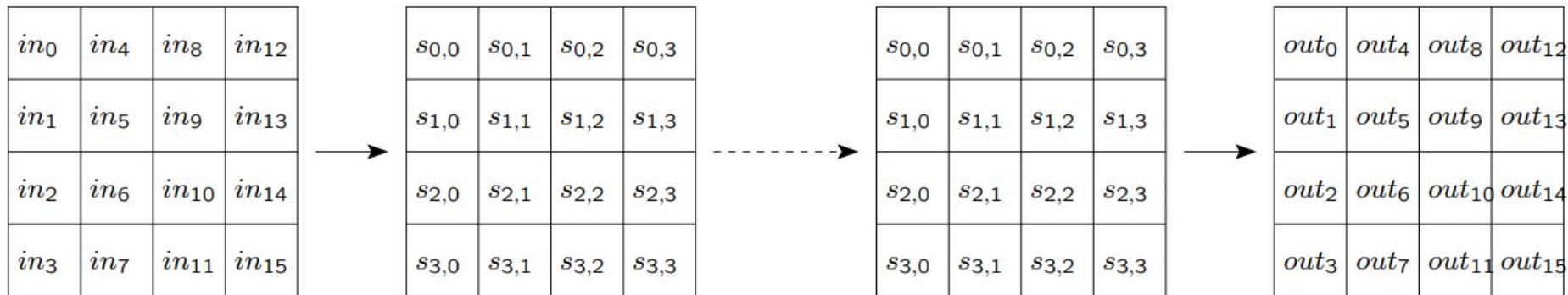
- So far we've covered:
 - what secure symmetric encryption is
 - high-level design of block ciphers
 - DES
- Next, we'll talk about:
 - AES
 - block cipher encryption modes

Advanced Encryption Standard (AES)

- In 1997 NIST made a formal call for an **unclassified publicly disclosed encryption algorithm available worldwide and royalty-free**
 - the goal was to replace DES with a new standard called AES
 - the algorithm must be a symmetric block cipher
 - the algorithm must support (at a minimum) 128-bit blocks and key sizes of 128, 192, and 256 bits
- The **evaluation criteria** were:
 - security
 - speed and memory requirements
 - algorithm and implementation characteristics

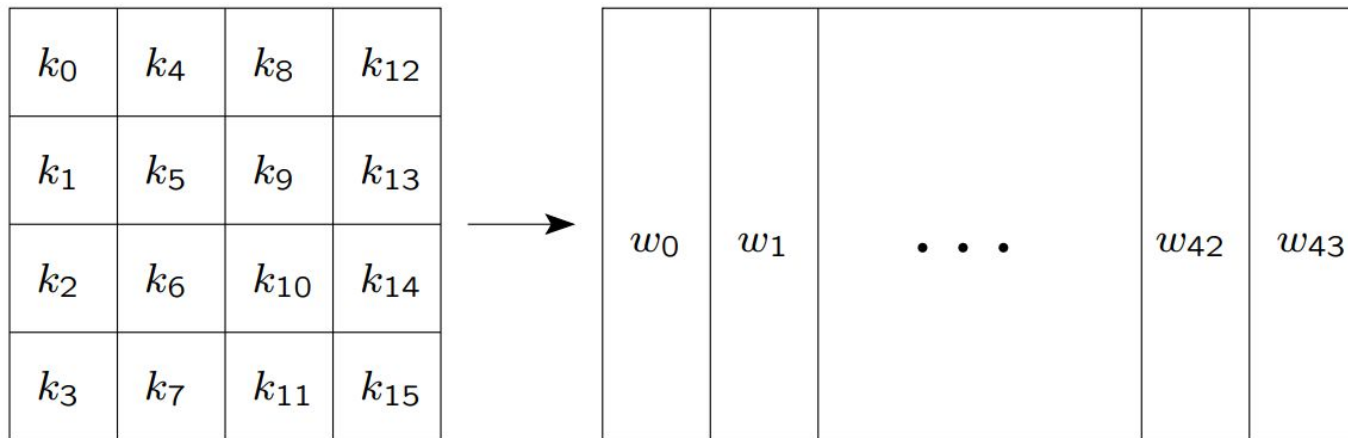
AES

- During encryption:
 - the block is copied into the state matrix
 - the state is modified at each round of encryption and decryption
 - the final state is copied to the ciphertext



AES

- The key schedule in AES:
 - the key is treated as a 4×4 matrix as well
 - the key is then expanded into an array of words
 - each word is 4 bytes and there are 44 words (for 128-bit key)
 - four distinct words serve as a round key for each round



AES

- Rijndael doesn't have a Feistel structure
 - 2 out of 5 AES candidates (including Rijndael) don't use Feistel structure
 - they process the entire block in parallel during each round
- **The operations** are (3 substitution and 1 permutation operations):
 - SUBBYTES: byte-by-byte substitution using an S-box
 - SHIFTRROWS: a simple permutation
 - MIXCOLUMNS: a substitution using *mod 28* arithmetics
 - ADDROUNDKEY: a simple XOR of the current state with a portion of the expanded key

AES

- At a high-level, **encryption** proceeds as follows:
 - set initial state $s_0 = m$
 - perform operation **ADDROUNDKEY** (XORs k_i and s_i)
 - for each of the first $Nr - 1$ rounds:
 - perform a substitution operation **SUBBYTES** on s_i and an S-box
 - perform a permutation **SHIFTRows** on s_i
 - perform an operation **MIXCOLUMNS** on s_i
 - perform **ADDROUNDKEY**
 - the last round is the same except no **MIXCOLUMNS** is used
 - set the ciphertext $c = s_{Nr}$

AES

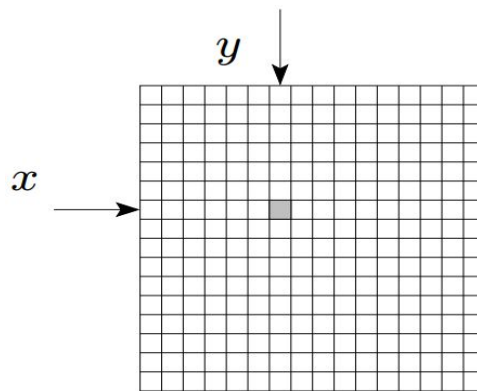
- More about Rijndael design...
 - ADDROUNDKEY is the only operation that uses key
 - that's why it is applied at the beginning and at the end
- all operations are reversible
- the decryption algorithm uses the expanded key in the reverse order
- the decryption algorithm, however, is not identical to the encryption algorithm

AES

- The SUBBYTES operation
 - maps a state byte $s_{i,j}$ to a new byte $s'_{i,j}$ using S-box
 - the S-box is a 16×16 matrix with a byte in each position
 - the S-box contains a permutation of all possible 256 8-bit values
 - the values are computed using a formula
 - it was designed to resist known cryptanalytic attacks (i.e., to have low correlation between input bits and output bits)

AES

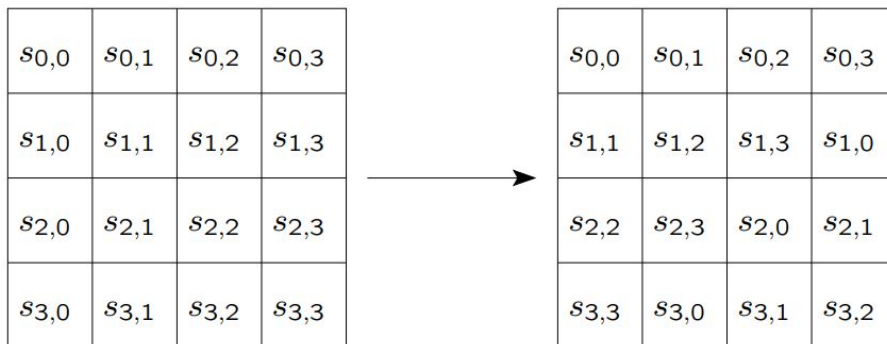
- The SUBBYTES operation
 - to compute the new $s'_{i,j}$:
 - set x to the 4 leftmost bits of $s_{i,j}$ and y to its 4 rightmost bits
 - use x as the row and y as the column to locate a cell in the S-box
 - use that cell value as $s'_{i,j}$



- the same procedure is performed on each byte of the state

AES

- The SHIFTRROWS operation
 - performs circular left shift on state rows
 - 2nd row is shifted by 1 byte
 - 3rd row is shifted by 2 bytes
 - 4th row is shifted by 3 bytes



- important because other operations operate on a single cell

AES

- The MIXCOLUMNS operation
 - multiplies the state by a fixed matrix

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

- was designed to ensure good mixing among the bytes of each column
- the coefficients 01, 02, and 03 are for implementation purposes (multiplication involves at most a shift and an XOR)

AES

- Decryption:
 - inverse S-box is used in SUBBYTES
 - inverse shifts are performed in SHIFTROWS
 - inverse multiplication matrix is used in MIXCOLUMNS
- Key expansion:
 - was designed to resist known attacks and be efficient
 - knowledge of a part of the key or round key doesn't enable calculation of other key bits
 - round-dependent values are used in key expansion

AES

- Summary of Rijndael design
 - simple design but resistant to known attacks
 - very efficient on a variety of platforms including 8-bit and 64-bit platforms
 - highly parallelizable
 - had the highest throughput in hardware among all AES candidates
 - well suited for restricted-space environments (very low RAM and ROM requirements)
 - optimized for encryption (decryption is slower)

AES Hardware Implementation

- It's been long known that hardware implementations of AES are extremely fast
 - the speed of encryption is compared with the speed of disk read
- Hardware implementations however remained inaccessible to the average user
- Recently Intel introduced new AES instruction set (AES-NI) in its commodity processors
 - other processor manufacturers support it now as well
 - hardware acceleration can be easily used on many platforms

Secure Encryption

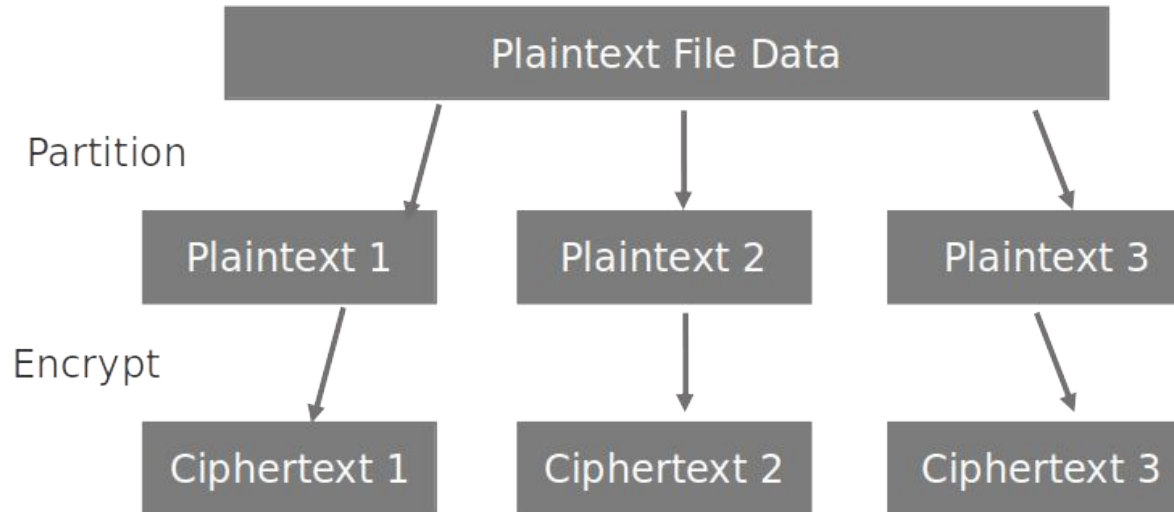
- For symmetric encryption to be secure, the key must be chosen completely at random
 - cryptography failures are often due to incorrect implementations
- Using a strong block cipher is not enough for secure encryption!
 - if you need to send more than 1 block (i.e., 16 bytes) over the key lifetime, applying plain block cipher to the message as will fail even weak definitions of secure encryption

$\text{Enck}(b1), \text{Enck}(b2), \dots$

- no deterministic encryption can be secure if multiple blocks are sent

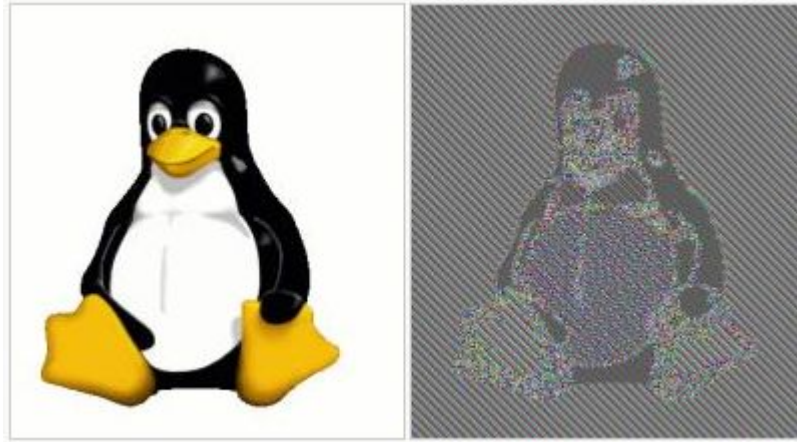
Block Cipher Limitation

- Block length is fixed (n-bit)
- Need to Partition into n-bit blocks to encrypt large messages



Block Cipher Limitation

- Does not hide data patterns, unsuitable for long messages



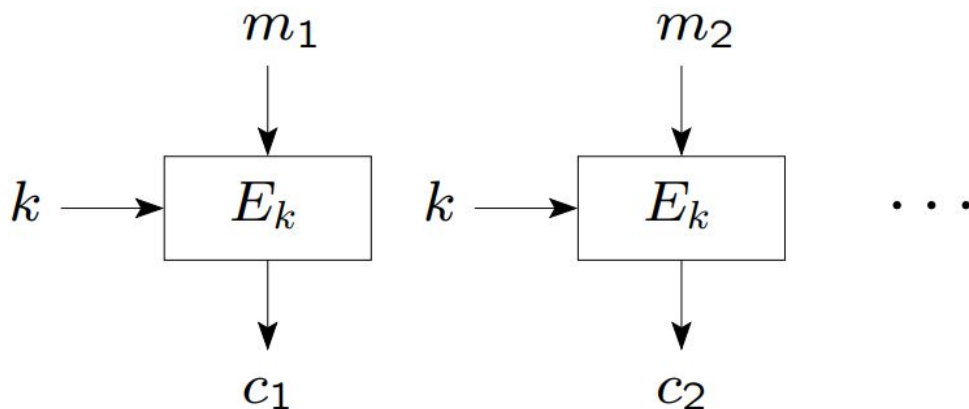
- Susceptible to replay attacks
 - Example: a wired transfer transaction can be replayed by resending the original message)

Encryption Modes

- Encryption modes indicate how messages longer than one block are encrypted and decrypted
- 4 modes of operation were standardized in 1980 for Digital Encryption Standard (DES)
 - can be used with any block cipher
 - electronic codebook mode (ECB), cipher feedback mode (CFB), cipher block chaining mode (CBC), and output feedback mode (OFB)
- 5 modes were specified with the current standard Advanced Encryption Standard (AES) in 2001
 - the 4 above and counter mode

Encryption Modes

- **Electronic Codebook (ECB)** mode
 - divide the message m into blocks $m_1 m_2 \dots m_\ell$ of size n each
 - encipher each block separately: for $i = 1, \dots, \ell$, $c_i = E_k(m_i)$, where E denotes block cipher encryption
 - the resulting ciphertext is $c = c_1 c_2 \dots c_\ell$



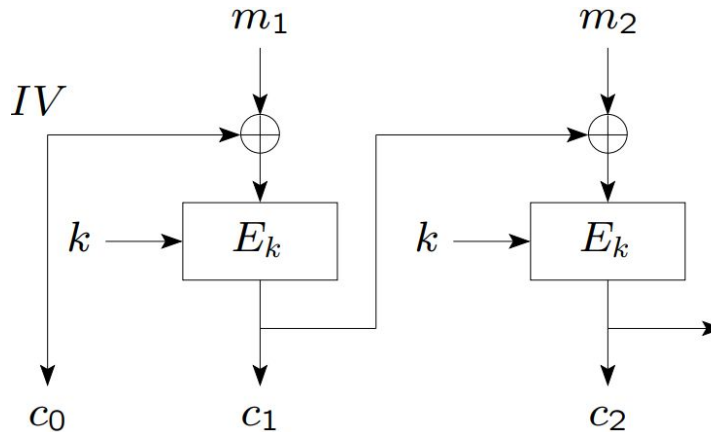
Encryption Modes

- Properties of ECB mode:
 - identical plaintext blocks result in identical ciphertexts (under the same key)
 - each block can be encrypted and decrypted independently
 - this mode doesn't result in secure encryption
- ECB mode is a plain invocation of the block cipher
 - it allows the block cipher to be used in other, more complex cryptographic constructions

Encryption Modes

- Cipher Block Chaining (CBC) mode

- set $c_0 = IV \xleftarrow{R} \{0, 1\}^n$ (initialization vector)
- encryption: for $i = 1, \dots, \ell$, $c_i = E_k(m_i \oplus c_{i-1})$
- decryption: for $i = 1, \dots, \ell$, $m_i = c_{i-1} \oplus D_k(c_i)$, where D is block cipher decryption



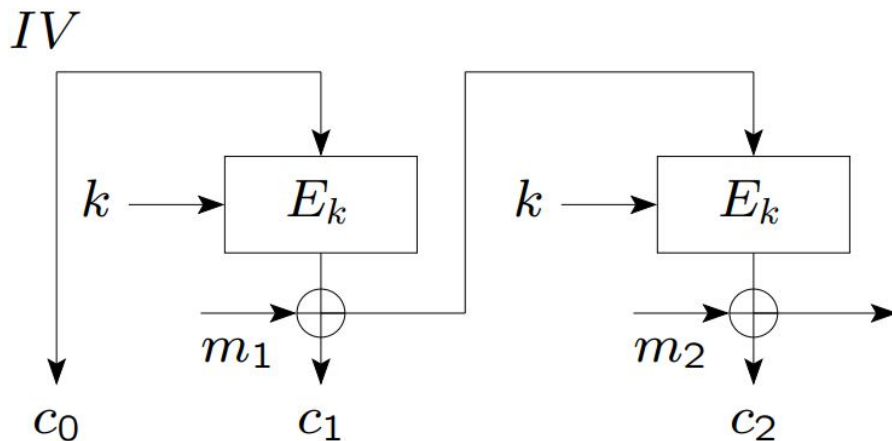
Encryption Modes

- Properties of CBC mode:
 - this mode is CPA-secure (has a formal proof) if the block cipher can be assumed to produce pseudo random output
 - a ciphertext block depends on all preceding plaintext blocks
 - sequential encryption, cannot use parallel hardware
 - *IV* must be random and communicated intact
 - if the *IV* is not random, security quickly degrades
 - if someone can fool the receiver into using a different *IV*, security issues arise

Encryption Modes

- Cipher Feedback (CFB) mode
 - the message is XORed with the encryption of the feedback from the previous block
 - generate random IV and set initial input $I_1 = IV$
 - encryption: $c_i = E_k(I_i) \oplus m_i; I_{i+1} = c_i$
 - decryption: $m_i = c_i \oplus E_k(I_i)$

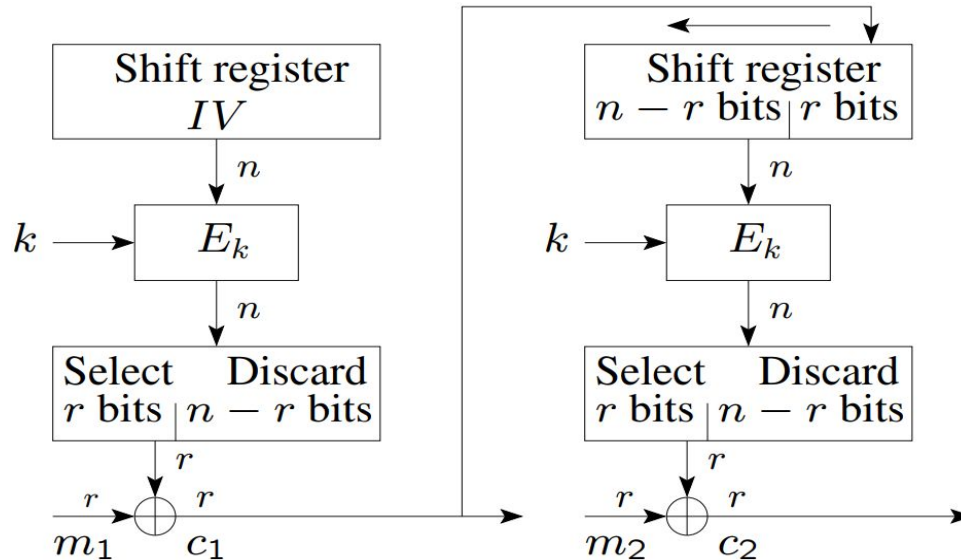
Encryption Modes



- This mode allows the block cipher to be used as a **stream cipher**
 - if our application requires that plaintext units shorter than the block are transmitted without delay, we can use this mode
 - the message is transmitted in r -bit units (r is often 8 or 1)

Encryption Modes

- **Cipher Feedback (CFB) mode:**
 - input: key k , r -bit plaintext blocks $m_1, \dots$
 - output: n -bit IV , r -bit ciphertext blocks $c_1, \dots$

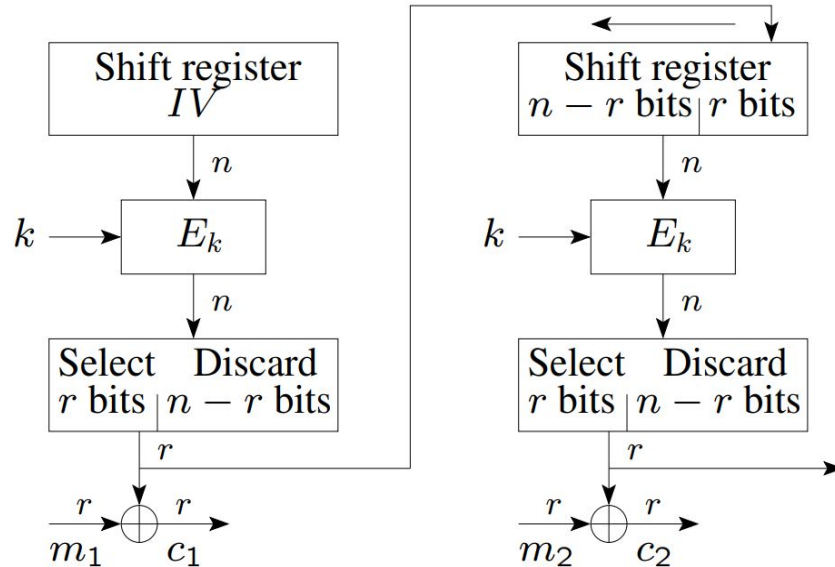


Encryption Modes

- Properties of CFB mode:
 - the mode is CPA-secure (under the same assumption that the block cipher is strong)
 - similar to CBC, a ciphertext block depends on all previous plaintext blocks
 - throughput is decreased when the mode is used on small units
 - one encryption operation is applied per r bits, not per n bits

Encryption Modes

- **Output Feedback (OFB) mode:**
 - similar to CFB, but the feedback is from encryption output and is independent of the message

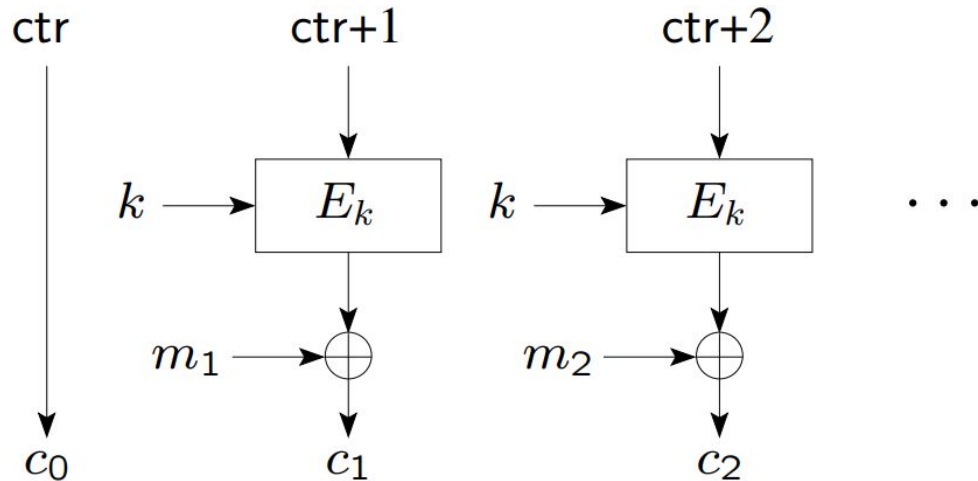


Encryption Modes

- **Output Feedback (OFB) mode:**
 - n -bit feedback is recommended
 - using fewer bits for the feedback reduces the size of the cycle
- **Properties of OFB:**
 - the mode is CPA-secure
 - the key stream is plaintext-independent
 - similar to CFB, throughput is decreased for $r < n$, but the key stream can be precomputed

Encryption Modes

- Counter (CRT) mode:
 - a counter is encrypted and XORed with a plaintext block
 - no feedback into the encryption function
 - initially set $ctr = IV \xleftarrow{R} \{0, 1\}^n$



Encryption Modes

- Counter (CRT) mode:
 - encryption: for $i = 1, \dots, \ell$, $c_i = E_k(ctr + i) \oplus m_i$
 - decryption: for $i = 1, \dots, \ell$, $m_i = E_k(ctr + i) \oplus c_i$
- Properties:
 - there is no need to pad the last block to full block size
 - if the last plaintext block is incomplete, we just truncate the last cipher block and transmit it

Encryption Modes

- Advantages of counter mode
 - Hardware and software efficiency: multiple blocks can be encrypted or decrypted in parallel
 - Preprocessing: encryption can be done in advance; the rest is only XOR
 - Random access: i th block of plaintext or ciphertext can be processed independently of others
 - Security: at least as secure as other modes (i.e., CPA-secure)
 - Simplicity: doesn't require decryption or decryption key scheduling
- But what happens if the counter is reused?

Summary

- **AES** is the current block cipher standard
 - it offers strong security and fast performance
- Five **encryption modes** are specified as part of the standard
 - ECB mode is not for secure encryption
 - any other encryption mode achieves sufficient security
 - use one of these modes for encryption even if the message is a single block
- **Strong randomness** is required for cryptographic purposes
 - key generation, IV generation, etc.