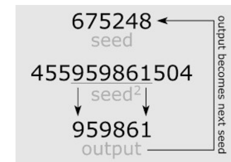
Lecture 4:
Stream and Block Ciphers

Stephen Huang

Pseudorandomness

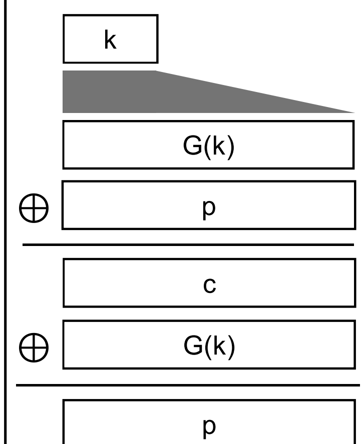
- A pseudorandom sequence of numbers is one that appears to be statistically random despite having been produced by a completely deterministic and repeatable process.



Content

1. Stream Cipher examples:
   - RC4
   - Salsa20
   - ChaCha20
2. Block Ciphers
   - What is a block cipher?
3. Data Encryption Standard (DES)
4. Advanced Encryption Standard (AES)

Stream Ciphers



- encrypt plaintext by XORing it to the pseudorandom sequence bit-by-bit
- decrypt ciphertext by XORing it to the pseudorandom sequence bit-by-bit

same as encryption

RC4 Cipher

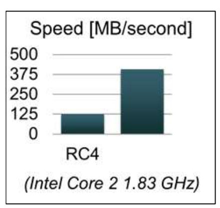
- RC4 Cipher (Rivest Cipher 4): Old WiFi and Web Security
- Designed in 1987 by Ron Rivest for RSA Security, a security company
  - Originally, it was kept a trade secret, but someone leaked it in 1994
- Advantages
  - Variable key length (from 8 to 2048 bits)
  - Very simple, based on byte-oriented operations: Only eight to sixteen machine operations are required per output byte

RC4 Cipher

- Applications
  - WiFi security: WEP (1997) and WPA (2003)
  - Very practical attack found in 2001, WEP and WPA deprecated in 2004.
- Web security (HTTPS):
  - Secure Socket Layer (SSL) 1995
  - Transport Layer Security (TLS) 1999
  - Practical attack found in 2013, RC4 in SSL/TLS deprecated in 2015
- RC4 had a good run, but it has been retired.

Salsa20 & ChaCha20 Ciphers

- Designed by Daniel Bernstein in 2005 (Salsa20) and 2008 (ChaCha20)
  - Not patented; several public domain implementations
  - ChaCha20 variant: more secure, more efficient
- Key length: 128 or 256 bits
- Advantages
  - Fast software implementation (simple 32-bit operations)
  - Can seek to any position in the output sequence
  - 64-bit nonce is part of the algorithm (to prevent key-reuse issues)



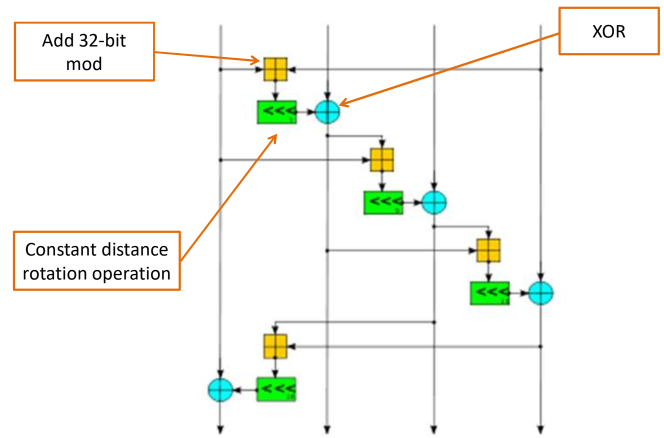
Salsa20 & ChaCha20 Ciphers

- Security: no significantly stronger attacks than brute force (yet)
- Adoption
  - Google implemented it in OpenSSL as a replacement for RC4
  - Linux (and some other operating systems) use it for random number generation

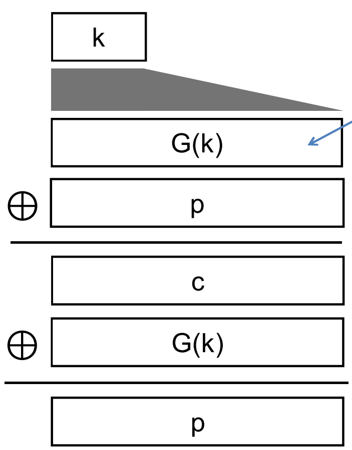
Salsa20 Cipher Algorithm

- Generates its output in blocks of 16 x 32 bits
- Internal state: 16 x 32 bits
  - initialized using the key (128, 192, 256 bits), the nonce (64 bits), and seek position (64 bits)
- Operations for updating the state:
  XOR, 32-bit addition mod 2^{32}, and rotating 32-bit values
- Salsa20 performs 20 rounds of add-rotate-XOR, each of which updates all values in the state
  - Salsa20/8 and Salsa20/12 perform only 8 and 12 rounds
- Finally, the state is added to the original state to obtain the output

Add-Rotate-XOR



Stream Ciphers

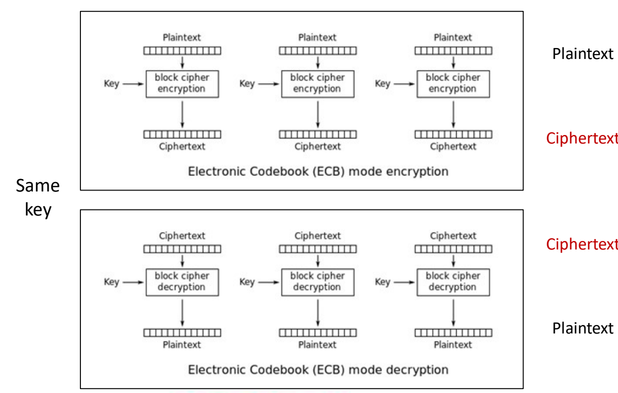


The generator does not need to be invertible.
Each sequence can be used only once.

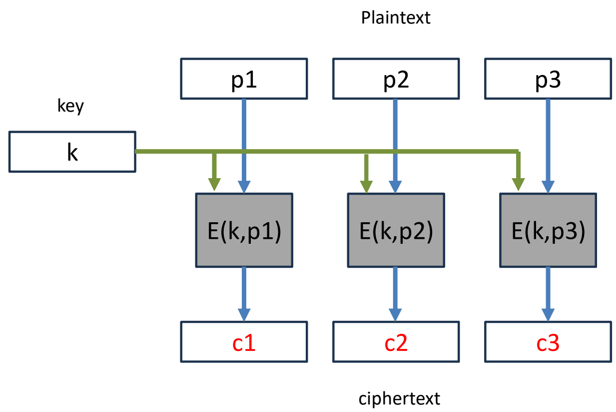
- encrypt plaintext by XORing it to the pseudorandom sequence bit-by-bit
- decrypt ciphertext by XORing it to the pseudorandom sequence bit-by-bit

2. Block Ciphers

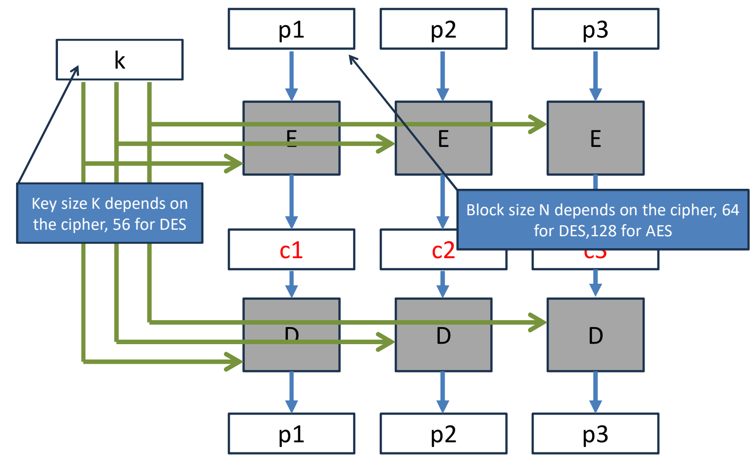
- Encrypt plaintext in fixed-size blocks.
- Encryption/decryption are different operations.



Block Ciphers

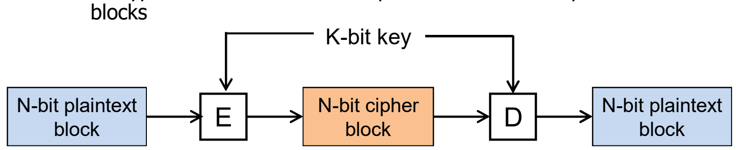


Block Ciphers



Design Considerations

- Key size: number of possible keys with K-bit key = 2^K (must prevent brute force attacks)
- Block size
  - too short → does not hide patterns in the plaintext
    - N = 8 bits (1 character in ASCII), same as a classic substitution cipher
  - too long → impractical, wasteful
- Encryption must be invertible
  - different input blocks must be transformed by the encryption to different output blocks
  - encryption can be viewed as a permutation over all possible N-bit blocks

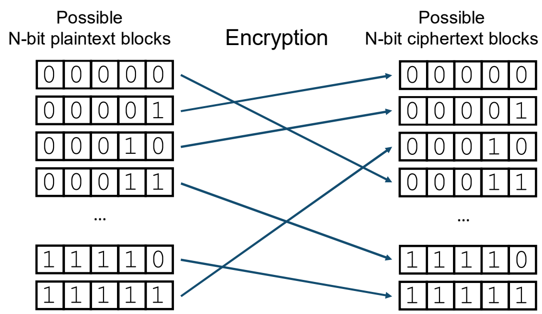


Initialization Vector

- Sometimes, an Initialization Vector (IV) is used to ensure distinct ciphertexts are produced even when the same plaintext is encrypted multiple times independently with the same key.

Secure Block Cipher

- An N-bit block cipher can be viewed as a permutation over all possible N-bit blocks.



mapping depends on the
cipher algorithm and random secret key

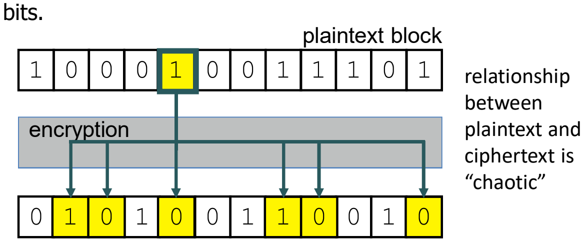
Secure Block Cipher

- An N-bit block cipher can be viewed as a permutation over all possible N-bit blocks
  - number of possible permutations with N-bit blocks $= 2^N!$
- An N-bit block cipher is secure if it is indistinguishable from a random permutation of N-bit blocks (for a computationally bounded attacker)
- We need more practical goals to design a practical cipher.

Secure Block Ciphers in Practice

1. Diffusion

- Goal: dissipate the plaintext's statistical structure over the ciphertext's long-range statistics.
- Each plaintext bit should affect the value of many ciphertext bits.
- Ideally, changing one of the bits in the plaintext changes half of the bits in the ciphertext block.

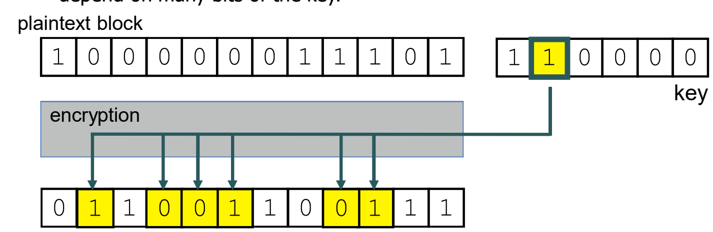


ciphertext block

Secure Block Ciphers in Practice

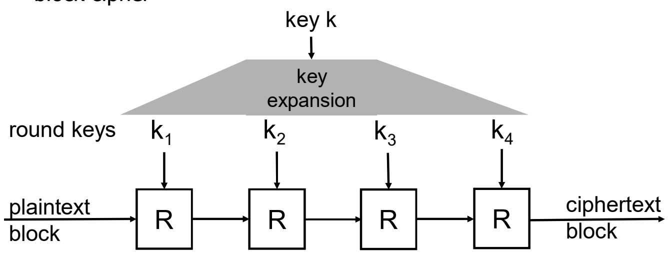
2. Confusion

- Goal: make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as possible.
- Each bit of the ciphertext should depend on many bits of the key.
- Ideally, changing one of the bits in the key changes half of the bits in the plaintext block



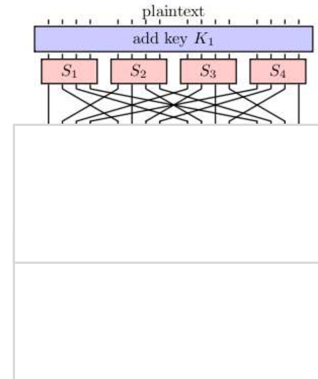
Iterated Block Ciphers

- It is difficult to design a single invertible transformation that satisfies both the diffusion and confusion properties
- R: round function (4 rounds in the example below)
  - relatively "weak" transformation, which introduces some diffusion and confusion
  - by combining a large number of rounds, we can build a strong block cipher



Substitution-Permutation Ciphers

- A very common subtype of iterated block ciphers
- Each round R consists of two steps
- Substitution S
  - substitutes a small block of bits with another small block
  - ideally, changing one input bit changes half of the output bits
- Permutation P
  - permutation of all the bits

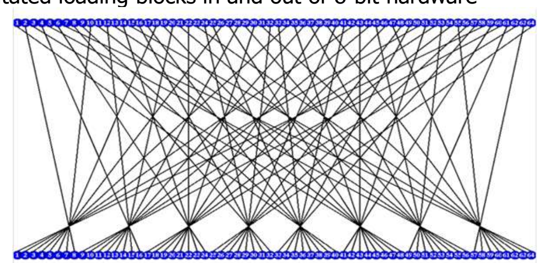


3. Data Encryption Standard

- DES: Data Encryption Standard.
- In the early 1970s, Horst Feistel developed the Lucifer cipher at IBM with his colleagues
  - multiple variants with key and block sizes from 48 to 128 bits
- In 1973, the National Bureau of Standards (now named NIST) solicited proposals for a government-wide standard encryption
- In 1974, IBM submitted a cipher based on Lucifer
- In 1976, DES was approved as a federal standard by the NBS
  - block size: 64 bits
  - key size: 56 bits
  - iterated substitution-permutation cipher with 16 rounds

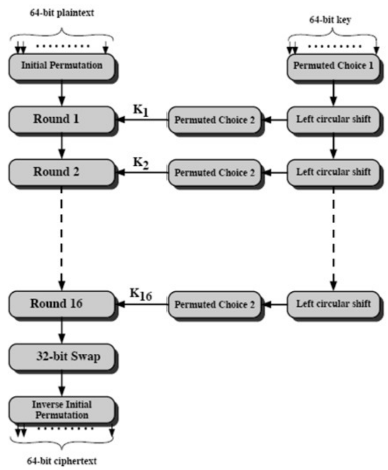
DES Structure

- Key
  - 56-bit random,
  - plus 8-bit parity check
- Initial Permutation
  - no cryptographic significance
  - facilitated loading blocks in and out of 8-bit hardware



DES Structure

Key permutation
- discards the parity bits
- no cryptographic significance

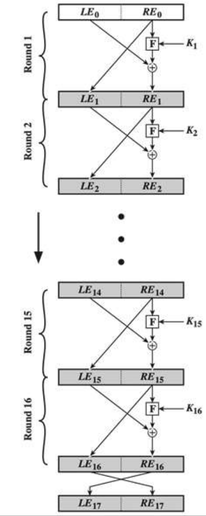


Feistel Network

- Encryption round
  - *input*: block from the previous round (or the plaintext)
  - divide input into two halves L_i and R_i
  - derive round key K_i from the secret key (different for each round)
  - *output*:

$$L_{i+1} = R_i$$

$$R_{i+1} = L_i \oplus F(K_i, R_i)$$



Feistel Network

- Decryption round
  - We can invert the encryption without inverting F:

$R_i = L_{i+1}$

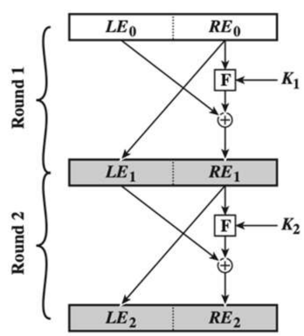
$R_{i+1} \oplus F(K_i, L_{i+1})$

$= R_{i+1} \oplus F(K_i, R_i)$

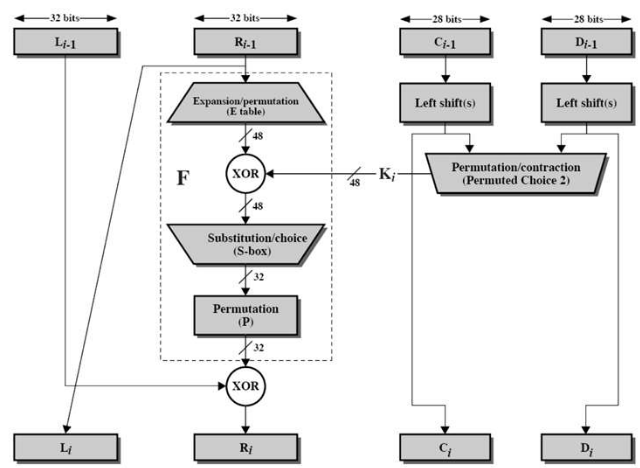
$= L_i \oplus F(K_i, R_i) \oplus F(K_i, R_i)$

$= L_i$

  - Use the same implementation with round keys in reverse order.

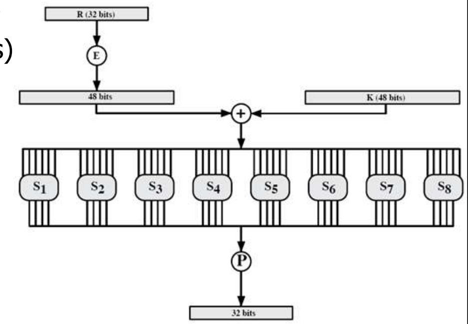


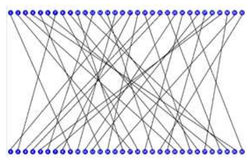
Single Round of DES



DES F-Function

- Expansion:
  duplicates half of the bits
- Substitution (S-boxes)
  maps 6-bit block into
  4-bit block-based
  lookup table
- Permutation (P-box):
  fixed permutation





DES S-Boxes

- Each S-box S_i is different
  - tables are specified by the standard
- S-boxes (and P-box) were carefully designed
  - randomly chosen boxes would result in an insecure cipher

S_5		Middle 4 bits of input															
		0000	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
Outer bits	**00**	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

Security of DES

- Cryptanalysis
  - best-known attack: linear cryptanalysis, which requires 2^{43} known plaintexts and ciphertexts, and finds a key in 2^{39} steps
- Vulnerable to brute-force attacks: key length = 56 bits
  - in 1977, Diffie and Hellman proposed a parallel machine with 1 million encryption devices (~$20 million), which would have found a DES key in 10 hours.
  - in 1997, RSA Security sponsored a contest for breaking DES: DESCHALL Project utilized thousands of Internet-connected computers run by volunteers to find DES key in 3 months.
  - in 1998, the Electronic Frontier Foundation built a machine for less than $250,000, which found a DES key in 56 hours.
  - in 2008, SciEngines designed RIVYERA, which can find a DES key in less than a day and costs around $10,000.
- Since 1999, DES is permitted by NIST only in legacy systems

DES Analysis

- The DES satisfies both the desired properties of a block cipher. These two properties make the cipher very strong.
  - Avalanche effect – A small change in plaintext results in a very great change in the ciphertext.
  - Completeness – Each bit of ciphertext depends on many bits of plaintext.
- Cryptanalysis found weaknesses in DES when the key selected are weak. These keys shall be avoided.
- DES has proved to be a very well-designed block cipher. There have been no significant cryptanalytic attacks on DES other than exhaustive key searches.

4. AES

- Advanced Encryption Standard (AES)

	DES	AES
Developed	1977	2000
Cipher Type	Symmetric block cipher	Symmetric block cipher
Block size	64 bits	128 bits
Key length	56 bits	128/192/256 bits
Security	Rendered insecure	Considered secure

AES

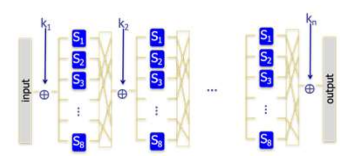
- In 1997, NIST announced a request for a proposal to replace DES
- Based on initial feedback, NIST announced a call for ciphers
  - requirements: 128-bit block size and 128, 192, and 256-bit key size
- 15 submissions were received in 9 months
  - ciphers were evaluated based on both their strength against cryptanalytic attacks as well as performance
- In 1999, the list was narrowed to five “AES finalists.”
- In 2000, NIST announced the winning cipher: Rijndael
  - developed by Belgian cryptographers Joan Daemen and Vincent Rijmen
- Standard: FIPS PUB 197: Advanced Encryption Standard (2001)

AES Applications

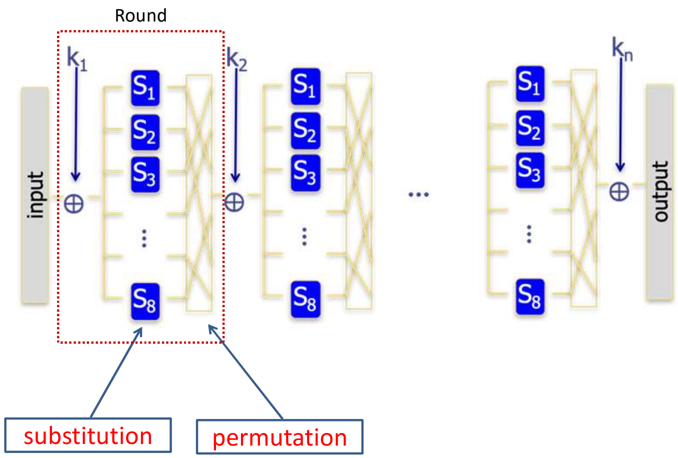
- WiFi security
  - WPA2 / WPA3: current standards
- Web security (HTTPS)
  - SSL/TLS: supported since 2008, one of the most widely used ciphers today
- Other protocols
  - IPSec, SSH
- Disk encryption
  - FileVault (Mac OS X), BitLocker (Windows)
- Compressed archives
  - 7z, WinZIP

AES Structure

- Substitution-permutation cipher
  - but not a Feistel network
- Each round must be invertible for decryption
- Key expansion and schedule: generates a different “round key” for each round
- The number of rounds depends on the key size
  - 10 for a 128-bit key, 12 for a 192-bit key, 14 rounds for a 256-bit key

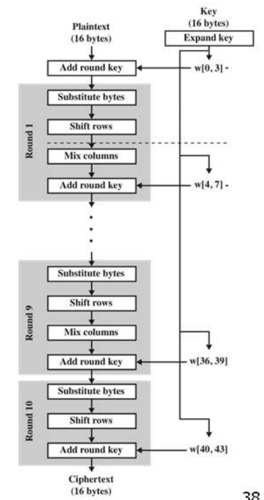


AES Structure

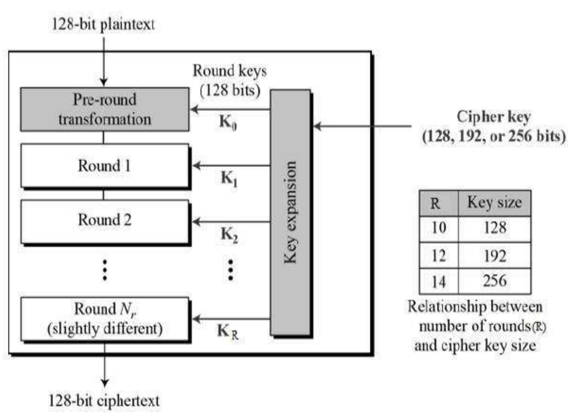


AES Round

- Input:
  - 128-bit "state" from previous round (or the plaintext) represented as a 4 x 4 byte matrix
  - 128-bit round key (from key schedule)
- Output: 128-bit state
- Each round consists of multiple steps:
  - `AddRoundKey`: XOR round key to the state
  - substitution and permutation:
    - `SubBytes`
    - `ShiftRows`
    - `MixColumns`



AES Structure



- AES Encryption Process

- Stalling, p. 175

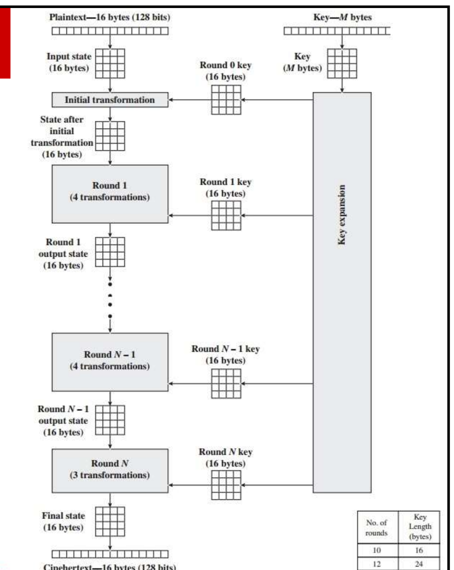


Figure 5.1 AES Encryption Process

- AES Encryption and Decryption
- Stalling, p. 178

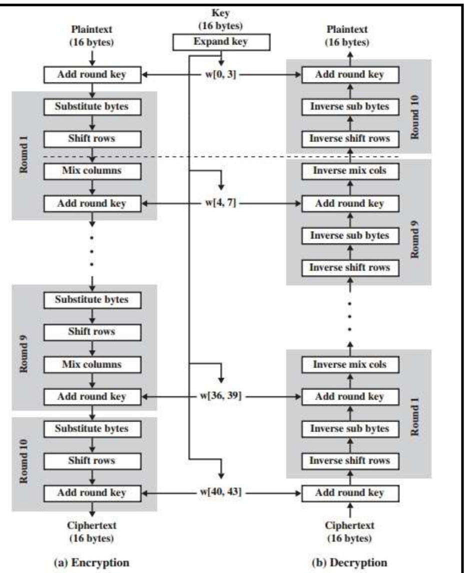
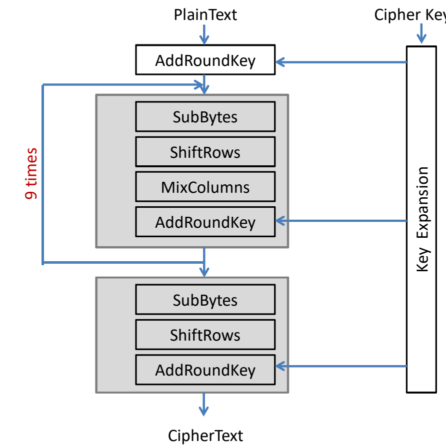


Figure 5.3 AES Encryption and Decryption

Encryption steps

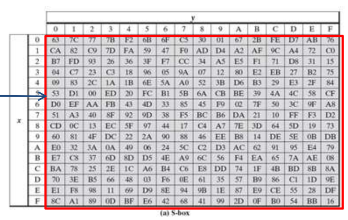
- Repeat the following four steps for 10 rounds, except there is no need to Mix Columns in the last round.
  - Substitute Bytes
  - Shift Rows
  - Mix Columns
  - Add Round Key
- AES Transformation Functions



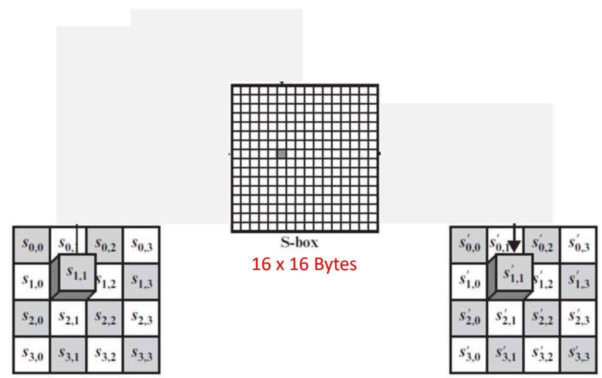
Step 1: SubBytes

- 128-bit (16 bytes) key (10 rounds in the example)
- Each byte is replaced using an 8-bit substitution box (S-box, 16 x 16 matrix)
  - defined using mathematical operations: multiplicative inverse over a finite field + affine transformation
- Designed to be resistant to cryptanalysis
  - minimize correlation to linear functions
  - minimize difference propagation

All 256 8-bit values



Substitute Byte Transformation



Example

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	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	13	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

(a) S-box

EA	04	65	85
83	45	5D	96
5C	33	98	B0
F0	2D	AD	C5

→

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

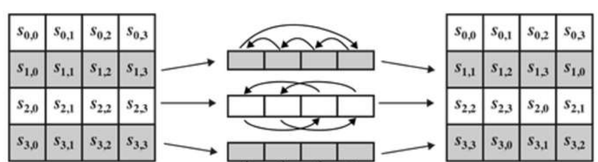
Step 2: (Forward) Shift Rows

- Input: Output of the Substitute Bytes
- Row i shift left by i positions circularly

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

`ShiftRows` Step

- Cyclically shifts the second, third, and fourth rows to the left
  - second row is shifted one byte
  - third row is shifted two bytes
  - forth row is shifted three bytes
- Ensures that the 4 bytes of each column are spread out to four different columns → provides diffusion
  - without this step, each input byte would affect only a single column



input output

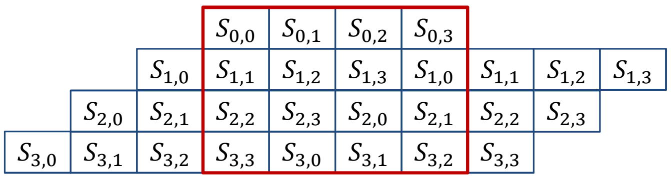
Shift Rows

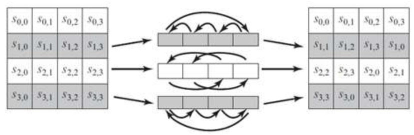
- Input: Output of the Substitute Bytes
- Row i shift left by i positions circularly

			$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
		$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$	
	$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$		
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$			

Shift Rows

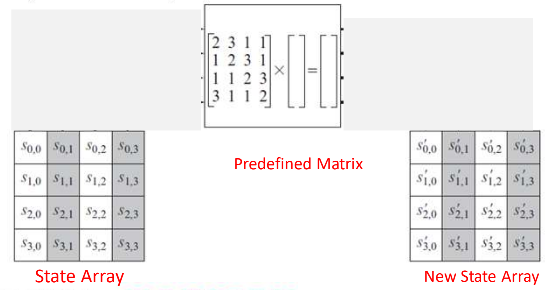
- Input: Output of the Substitute Bytes
- Row i shift left by i positions circularly
- Distribute a column





Step 3: Mix Column

- Each column is multiplied by a fixed matrix
  - invertible linear transformation
- Good mixing among the bytes of each column → provides diffusion
  - combined with `ShiftRows`, ensures that each output bit depends on every input bit after a few rounds



Mix Column

$$s'_{0,j} = (2 \cdot s_{0,j}) \oplus (3 \cdot s_{1,j}) \oplus s_{2,j} \oplus s_{3,j}$$
$$s'_{1,j} = s_{0,j} \oplus (2 \cdot s_{1,j}) \oplus (3 \cdot s_{2,j}) \oplus s_{3,j}$$
$$s'_{2,j} = s_{0,j} \oplus s_{1,j} \oplus (2 \cdot s_{2,j}) \oplus (3 \cdot s_{3,j})$$
$$s'_{3,j} = (3 \cdot s_{0,j}) \oplus s_{1,j} \oplus s_{2,j} \oplus (2 \cdot s_{3,j})$$

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

→

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

STEP 4: Add Round Key

- The 128 bits of State are bitwise XORed with the 128 bits of the round key.
- The add round key transformation is as simple as possible and affects every bit of State.

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

State

⊕

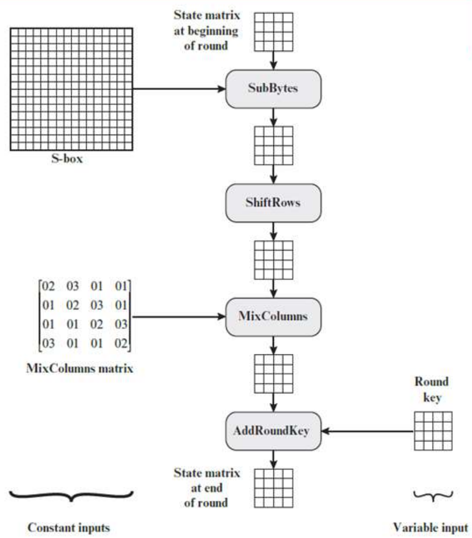
AC	19	28	57
77	FA	D1	5C
66	DC	29	00
F3	21	41	6A

Round Key

=

EB	59	8B	1B
40	2E	A1	C3
F2	38	13	42
1E	84	E7	D6

New State



Encryption Round

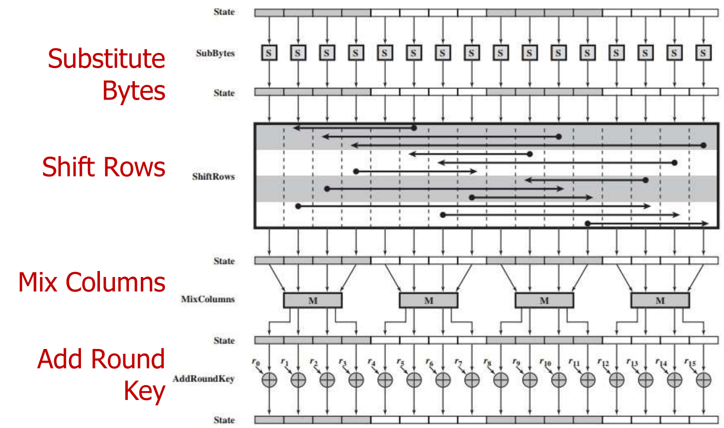
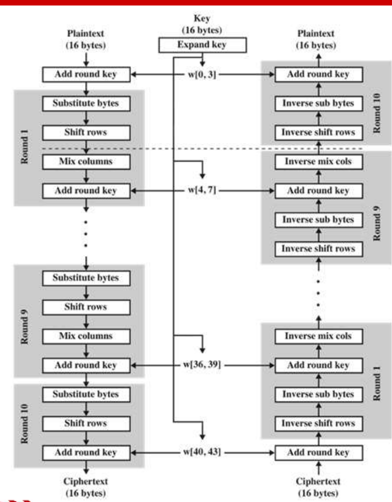


Figure 5.4 AES Encryption Round

AES Decryption

- Each step is invertible
  - `InvertMixColumns`: multiply by matrix inverse
  - `InvertShiftRows`: shift rows cyclically to the right
  - `InvertSubBytes`: invert affine transformation and multiplicative inverse
  - `InvertAddRoundKey`: XOR round key to state
- For decryption, round keys are used in reverse order



Performance and Security

- Operations on bytes and 32-bit words
  - Most operations can be precomputed (e.g., 256-byte substitution table for SubBytes)
- Hardware support: AES instruction set for CPUs
  - Introduced for x86 by Intel in 2008, supported by newer Intel and AMD CPUs
  - Other architectures also provide support (e.g., ARM, IBM Power, SPARC)
  - Instructions for computing a round of encryption/decryption, key generation, etc.
  - Supported by many software (e.g., Java, Linux cryptography API, OpenSSL)
- Best attack against arbitrary keys
  - In 2015, it was shown that 128-bit AES keys can be recovered in 2126 steps (only four times faster than brute-force search over the entire key space)
- There are no publicly known practical attacks

Other Notable Block Ciphers

- KASUMI
  - block cipher used UMTS (3G) cell phone networks (also in GSM as A5/3)
  - derived from MISTY1, a block cipher developed by Mitsubishi Electric in 1995
  - 64-bit blocks, 128-bit key, based on Feistel structure with 8 rounds
  - in 2010, a very efficient related-key attack was published; however, it is not applicable to how KASUMI is used in 3G networks
- Blowfish
  - designed in 1993 by Bruce Schneier
  - 64-bit blocks, 32 - 448-bit key, based on Feistel structure with 16 rounds (similar to DES)
  - small block size may be exploited if large amount of data is encrypted
- Twofish
  - based on Blowfish
  - one of the “AES finalists”, no practical attacks are known
- Serpent
  - substitution-permutation cipher with 32 rounds, operating on 32-bit words
  - one of the “AES finalists”, no practical attacks are known

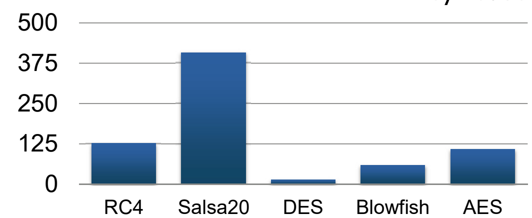
Block Ciphers vs. Stream Ciphers

Stream Ciphers

- can encrypt one bit at a time
- are typically faster and use less memory

Block Ciphers

- can be used to build various other cryptographic primitives
- leak less information with key reuse



Speed of ciphers [MB /second] measured on an Intel Core 2 1.83 GHz using the Crypto++ 5.6 library

Next Topic

- Stream and Block Ciphers
- Block Cipher Modes of Operation
- Public Key Encryption