Lecture 4

Encryption II

Suggested Readings:

- Chs 3 & 4 in KPS (recommended)
- Ch 3 in Stinson (optional)

1

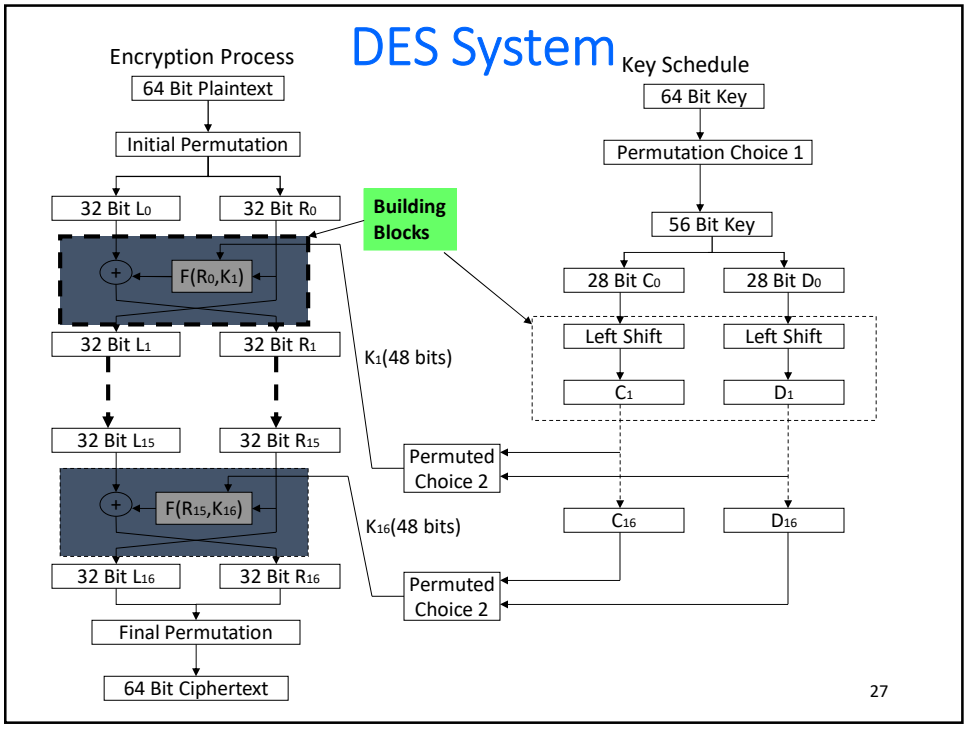
1

(Symmetric crypto)

"Modern" Block Ciphers

Data Encryption Standard (DES)

2



3

Breaking DES (Cryptanalysis)

DES Key size = 56 bits

- Brute force = 2^{55} attempts on avg
- Differential cryptanalysis ➔ 2^{47} chosen plaintexts [BS'89]
- Linear cryptanalysis ➔ 2^{43} known plaintexts [M'93]

•More than 16 rounds do not make it any stronger

•DES Key Problems:

•Weak keys (all 0s, all 1s, a few others)

•Key size = 56 bits = 8 * 7-bit ASCII

•Alphanumeric-only password converted to uppercase

8 * ~5-bit chars = 40 bits

33

4

Average Time for Exhaustive Key Search (for Brute-Force Attacks)

Key Size (bits)	Number of Alternative Keys	Time required at 10^6 Decr/µs
32	$2^{32} = 4.3 \times 10^9$	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	10 hours
128	$2^{128} = 3.4 \times 10^{38}$	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	5.9×10^{30} years

Today, **> 80 bits** can be generally considered "secure"

(but things are changing rapidly due to GPUs and special hardware; better to treat this only as the least length under not-so-rich/resourceful attackers)

5

How to strengthen DES: the case of double DES

- **2DES: C = DES (K1, DES (K2, P))**
- **Seems to be hard to break by "brute force", approx. 2^{111} trials**
- **Assume Eve is trying to break 2DES and has a single (P,C) pair**

Meet-in-the-middle ATTACK:

I. For each possible K'_i (where $0 <= i < 2^{56}$)
   1. Compute $C'_i = DES (K'_i , P)$
   2. Store: $[C'_i , K'_i]$ in look-up table T (indexed by C'_i)

II. For each possible K''_i (where $0 <= i < 2^{56}$)
   1. Compute $C''_i = DES^{-1} (K''_i , C)$
   2. Look up C''_i in T
   3. If lookup succeeds, output: $K1=K'_i$, $K2=K''_i$

TOTAL COST: $O(2^{56} + 2^{56})$ operations + $O(2^{64})$ storage

43

6

DES Variants

- **2-DES:**
  - **C = E(K2,E(K1, P)) → 57 effective key bits (meet-in-the-middle attack)**
- **3-DES (Triple DES)**
  - **C = E(K3, D(K2, E(K1,P))) → 112 effective key bits (meet-in-the-middle attack)**
  - **C = E(K1, D(K2, E(K1,P))) → <=80 effective key bits**
- **DESX**
  - **C= K3 XOR E(K2, (K1 XOR P)) → seems like 184 key bits**
  - **Effective key bits → approx. 118**
- **Another simple variation:**
  - **C = K2 XOR E(K1, P) → weak!**

NOTE: The same variants can be constructed out of any cipher

44

7

DES Summary

- Feistel network based block cipher
- 64-bit data blocks
- 56-bit keys (8 parity bits)
- 16 rounds (shifts, XORs)
- Key schedule
- S-box selection secret …
- DES "aging"
- 2-DES: meet-in-the-middle attack
- 3-DES: 112-bit security
- DESX: 118-bit security

46

8

Advanced Encryption Standard (AES): The Rijndael Block Cipher

55

9

Introduction and History

- National Institute of Science and Technology (NIST) regulates standardization in the US
- By mid-90s, DES was an aging standard that no longer met the needs for strong commercial-grade encryption
- Triple-DES: Endorsed by NIST as a "de facto" standard
- But … slow in software and large footprint (code size)
- **Advanced Encryption Standard (AES)**
  - Goal is to define the Federal Information Processing Standard (FIPS) by selecting a new encryption algorithm suitable for encrypting (non-classified non-military) government documents
  - Candidate algorithms must be:
    - Symmetric-key ciphers supporting 128, 192, and 256 bit keys
    - Royalty-Free
    - Unclassified (i.e., public domain)
    - Available for worldwide export
  - 1997: NIST publishes request for proposal
  - 1998-1999: 15 submissions -> 5 finalists
  - 2000: NIST chooses Rijndael as AES

56

10

Introduction and History

- AES Round-3 Finalist Algorithms (ranked by vote # in AES Round-2, high to low):
  - Rijndael
    - by Joan Daemen and Vincent Rijmen (Belgium)
  - Serpent
    - by Ross Anderson (UK), Eli Biham (ISR) and Lars Knudsen (NO)
  - Twofish
    - From Counterpane Internet Security, Inc. (MN)
  - RC6
    - By Ron Rivest of MIT & RSA Labs, creator of the widely used RC4/RC5 algorithm and "R" in RSA
  - MARS
    - Candidate offering from IBM Research

57

11

Rijndael

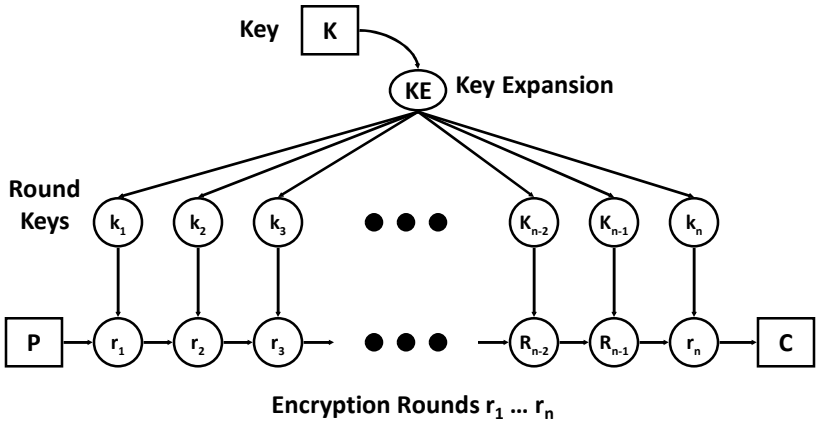
The Winner: Rijndael

- Joan Daemen (of Proton World International) and Vincent Rijmen (of Katholieke Universiteit Leuven).
- Pronounced "Rhine-doll"
- Allows only 128, 192, and 256-bit key sizes (unlike other candidates)
- Variable input block length: 128, 192, or 256 bits. All nine combinations of key-block length possible.
  - A block is the smallest data size the algorithm will encrypt
- Vast speed improvement over DES in both h/w and s/w implementations
  - 8,416 bytes/sec on a 20MHz 8051
  - 8.8 Mbytes/sec on a 200MHz Pentium Pro

58

12

Rijndael



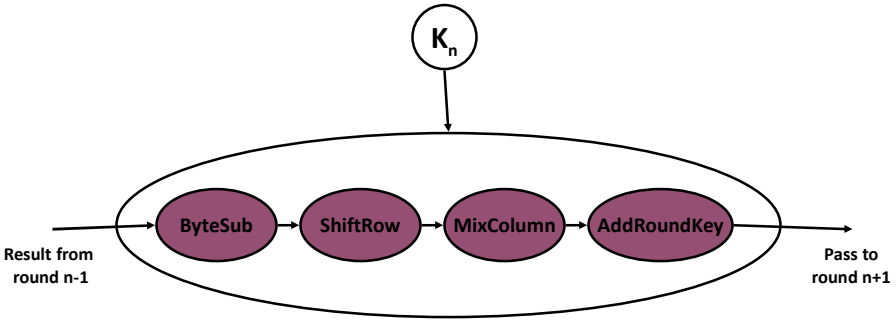
Encryption Rounds r_1 … r_n

- Key is expanded to a set of n round keys
- Input block P put thru n rounds, each with a distinct round sub-key.
- Strength of algorithm relies on difficulty of obtaining intermediate results (or *state*) of round i from round i+1 without the round key.

59

13

Rijndael



Detailed view of round n

- **Each round performs the following operations:**
  - **Non-linear Layer:** No linear relationship between the input and output of a round
  - **Linear Mixing Layer:** Guarantees high diffusion over multiple rounds
    - Very small correlation between bytes of the round input and the bytes of the output
  - **Key Addition Layer:** Bytes of the input are simply XOR'ed with the expanded round key

60

14

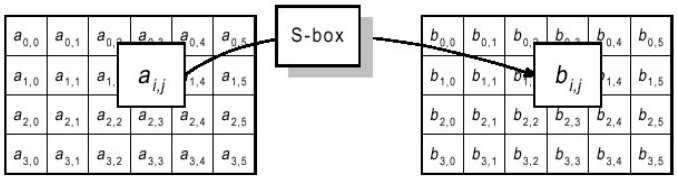
Rijndael

- Three layers provide strength against known types of cryptographic attacks: Rijndael provides "full diffusion" after only two rounds
- Cryptanalysis
  - Key recovery attack:
    - Best one only 4x faster than exhaustive search [BKR'11]
  - Related key attack:
    - AES-256: Given 2^99 input/output pairs from 4 related keys in AES-256 can recover keys in time 2^99 [BK'09]
    - However, how realistic is that?

61

15

Rijndael: ByteSub



Each byte at the input of a round undergoes a non-linear byte substitution according to the following transform:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Substitution ("S")-box

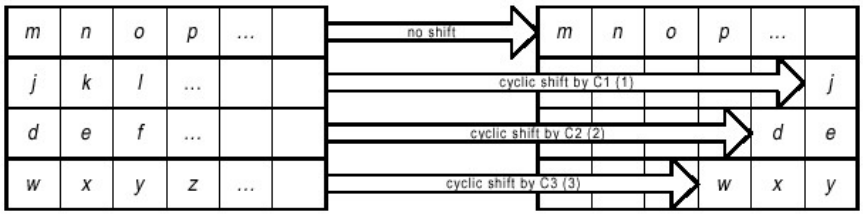
62

16

Rijndael: ShiftRow

Nb	C1	C2	C3
4	1	2	3
6	1	2	3
8	1	3	4

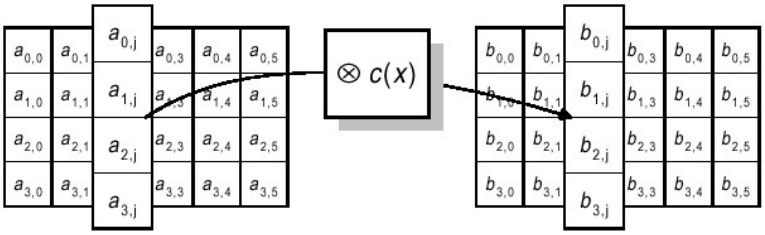
Depending on the block length, each "row" of the block is cyclically shifted according to the above table



63

17

Rijndael: MixColumn



Each column is multiplied by a fixed polynomial
$C(x) = '03'*X^3 + '01'*X^2 + '01'*X + '02'$

This corresponds to matrix multiplication b(x) = c(x) ⊗ a(x):

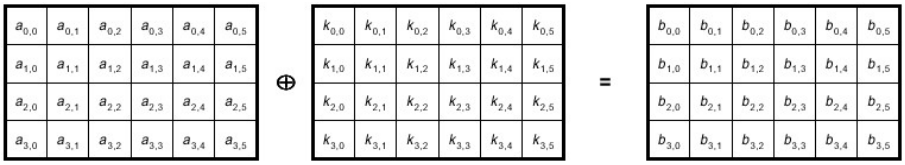
$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Not XOR

64

18

Rijndael: Key Expansion and Addition



Each word is simply XOR'ed with the expanded round key

Key Expansion algorithm:

```
KeyExpansion(int* Key[4*Nk], int* EKey[Nb*(Nr+1)])
{
    for(i = 0; i < Nk; i++)
       EKey[i] = (Key[4*i],Key[4*i+1],Key[4*i+2],Key[4*i+3]);
    for(i = Nk; i < Nb * (Nr + 1); i++)
    {
       temp = EKey[i - 1];
       if (i % Nk == 0)
          temp = SubByte(RotByte(temp)) ^ Rcon[i / Nk];
       EKey[i] = EKey[i - Nk] ^ temp;
    }
}
```

65

19

Rijndael: Implementations

- Well-suited for software implementations on 8-bit processors (important for "Smart Cards")
  - Atomic operations focus on bytes and nibbles, not 32- or 64-bit integers
  - Layers such as ByteSub can be efficiently implemented using small tables in ROM (e.g., < 256 bytes).
  - No special instructions are required to speed up operation, e.g., barrel-shifting registers on some embedded device microprocessors
- For 32-bit implementations:
  - An entire round can be implemented via a fast table lookup routine on machines with 32-bit or higher word lengths
  - Considerable parallelism exists in the algorithm
    - Each layer of Rijndael operates in a parallel manner on the bytes of the round state, all four component transforms act on individual parts of the block
    - Although the Key expansion is complicated and cannot benefit much from parallelism, it only needs to be performed *once* when the two parties switch keys.

66

20

Rijndael: Implementations

- Hardware Implementations
  - Rijndael performs very well in software, but there are cases when better performance is required (e.g., server and VPN applications).
  - Multiple S-Box engines, round-key XORs, and byte shifts can all be implemented efficiently in hardware when absolute speed is required
  - Small amount of hardware can vastly speed up 8-bit implementations
- Inverse Cipher
  - Except for the non-linear ByteSub step, each part of Rijndael has a straightforward inverse and the operations simply need to be undone in the reverse order.
  - However, Rijndael was specially written so that the same code that encrypts a block can also decrypt the same block simply by changing certain tables and polynomials for each layer. The rest of the operation remains identical.

21

Conclusions and The Future

- Rijndael is an extremely fast, state-of-the-art, highly secure algorithm
- Amenable to efficient implementation in both hw and sw; requires no special instructions to obtain good performance on any computing platform
- Triple-DES: officially being retired by NIST
  - In 2018, proposed to retire it: 3DES is deprecated for all new applications and usage is disallowed after 2023

22

Key things to remember for DES & AES

- From now on, treat it as a **permutation function** in the value space of **n-bit values**
  - c = DES(k, m):
    - A **one-to-one mapping** from a 64-bit value to another 64-bit value, and such a mapping is controlled by a 56-bit key
    - Since it's a one-to-one mapping, it's ***reversible from c to m*** with the same key k: $m = DES^{-1}(k, c)$
    - That's why it can be used for encryption/decryption
  - c = AES(k, m)
    - Similarly, one-to-one mapping from 128/192/256-bit value to another, controlled by the key k, also inversible
  - Both are hard to attack with much less efforts than brute-forcing the key space
    - But since the key size for AES can satisfy modern security requirement, AES is used today

36

23

However, a key question still remains...

- Each block cipher function (e.g., DES, AES) is for a **fixed-size of input** (e.g., 64/128/192/256 bits)
- However, common plaintext we need to encrypt can have arbitrary length, typically ***way longer than 8/16/24/32 characters***
- So the question is, **how to use block ciphers as building blocks in real-world use cases?**

36

24

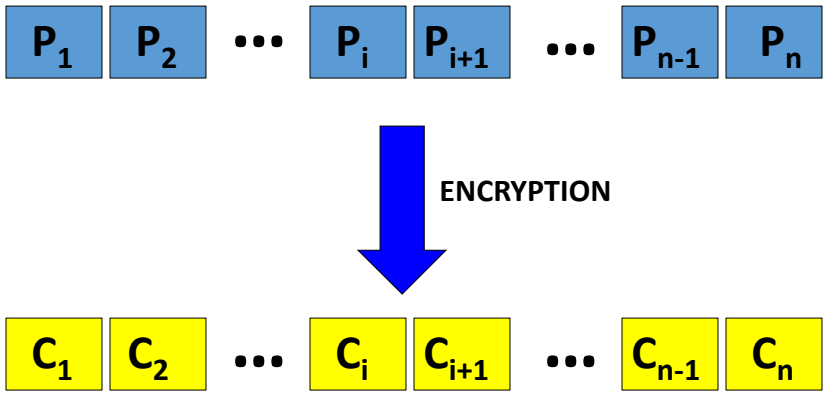
Modes of Operation

(not just for DES/AES, but for any block ciphers)

55

25

Modes of Operation
(for any block cipher like DES/AES)



http://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

35

26

"Native" ECB Mode

Electronic Code-Book (ECB) Mode

•Input to encryption algorithm is current plaintext block:

$$C_i = E (K, P_i)$$
$$P_i = D (K, C_i)$$

•Duplicate plaintext blocks (patterns) visible in ciphertext
- What if Alice encrypts one word per plaintext block?

•Ciphertext block rearrangement is possible
- To detect it, need explicit block numbering in plaintext

•Parallel encryption and decryption (random access)

•Error in one ciphertext block ➔ one-block loss

•One-block loss in ciphertext?

36

27

CBC Mode

Cipher-Block Chaining (CBC) Mode

•Input to encryption algorithm is the XOR of current plaintext block and preceding ciphertext block:

$$C_i = E (K, P_i \textbf{ XOR } C_{i-1}) \qquad C_0 = IV$$
$$P_i = D (K, C_i) \textbf{ XOR } C_{i-1}$$

37

28

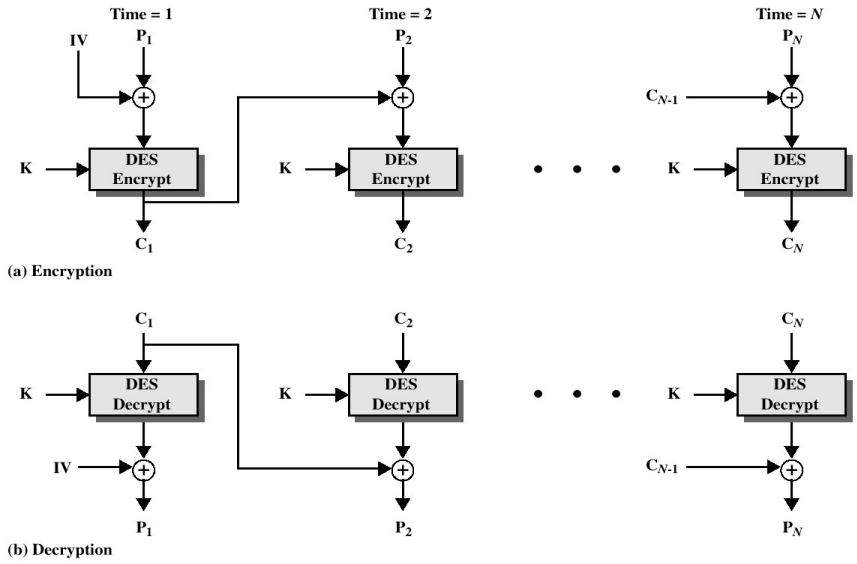


Figure 2.7 Cipher Block Chaining (CBC) Mode

29

CBC Mode

Cipher-Block Chaining (CBC) Mode

•Input to encryption algorithm is the XOR of current plaintext block and preceding ciphertext block:

$$C_i = E (K, P_i \textbf{ XOR } C_{i-1}) \qquad C_0=IV$$

$$P_i = D (K, C_i) \textbf{ XOR } C_{i-1}$$

•Duplicate plaintext blocks (patterns) NOT exposed

•Block rearrangement is detectable

•No parallel encryption

- How about parallel decryption?

•Error in one ciphertext block ➔ two-block loss

•One-block ciphertext loss?

37

30

OFB Mode

Output Feedback (OFB) Mode

• Key-stream is produced by repeated encryption of V_o:

$$C_i = E(K, V_{i-1}) \textbf{ XOR } P_i \qquad V_0=IV, \ldots, V_i = E(K, V_{i-1})$$
$$P_i = E(K, V_{i-1}) \textbf{ XOR } C_i$$

• Duplicate plaintext blocks (patterns) NOT exposed
• Block rearrangement is detectable
• Key-stream is independent of plaintext
  • How does that affect speed of encryption? Parallelism?
• Bit error in one ciphertext block ➔ one-bit error in plaintext
• One-block ciphertext loss ➔ big mess ☺ (unless you know it's lost)
• Can encrypt less than block size
• Adversary can flip selected bits in ciphertext resulting in same bits flipped in plaintext

39

31

CFB Mode

Cipher Feedback (CFB) Mode

•Key-stream is produced by re-encryption of preceding ciphertext -- C_{i-1}:

$$C_i = P_i \textbf{ XOR } E(K, C_{i-1}) \quad C_0=IV$$
$$P_i = E(K, C_{i-1}) \textbf{ XOR } C_i$$

• Duplicate plaintext blocks (patterns) NOT exposed
• Block rearrangement is detectable
• Key-stream is **dependent on** plaintext
  • How does that affect speed of encryption? Parallelism?
• Bit error in one ciphertext block ➔ one-bit + one-block loss in plaintext
• Adversary can still selectively flip/change bits
• One-block ciphertext loss ➔ 1-extra-block loss
• Can encrypt less than block size

40

32

CTR Mode

Counter (CTR) Mode

•Key-stream is produced by encryption increasing counter:

$C_i = E (K, CTR) \textbf{ XOR } P_i$ CTR ++

$P_i = E (K, CTR) \textbf{ XOR } C_i$ CTR++

•Duplicate plaintext blocks (patterns) NOT exposed, **unless**?

•Block rearrangement is detectable

•Key-stream is independent of plaintext

•Parallel encryption and decryption (random access)

•Bit error in one ciphertext block ➔ one-bit error in plaintext

•One-block ciphertext lost ➔ big mess (unless you're aware of it)

•Can encrypt less than block size

•Selective bit flipping still possible

41

33

MAC Mode

Message Authentication Code (MAC) Mode

• Used for integrity/authentication, not confidentiality

• Encryption is the same as in CBC mode, but, ciphertext is NOT sent except for the very last one

$C_i = E (K, P_i \textbf{ XOR } C_{i-1})$ C_0=IV

What is sent or stored: $\mathbf{P_1, . . ., P_n, C_n = MAC}$

Receiver recomputes C_n with K and compares

•Any change in plaintext results in unpredictable changes in MAC

42

34

Key things to remember for DES & AES

- View both as a **permutation functions** for **n-bit block/input:**
  - C = DES(k, m):
    - A **one-to-one mapping** from a 64-bit value to another 64-bit value, controlled by a 56-bit key k
    - Since it's one-to-one, it's ***reversible C → m*** with the same key k: $m = DES^{-1}(k, C)$
    - That's why it can be used for encryption/decryption
  - C = AES(k, m)
    - Also, a one-to-one mapping from 128/192/256-bit value to another, controlled by k, also invertible
  - Both are hard to attack with less effort than brute-force
    - But since AES key size satisfies modern security requirements, we use it today

36

35

DES Variants

Why does 3-DES (or generally n-DES) work?

Because, as a function, DES is not a group…

A "group" is an algebraic structure. One of its properties is that, taking any 2 elements of the group (a,b) and applying an operator F() yields another element c in the group.

Suppose: C = DES(K1,DES(K2,P))

There is no K, such that:

for each possible plaintext P, DES(K,P) = C

45

36

Other Symmetric Ciphers

- RC4 (Ron's Cipher #4) Stream Cipher:
  - Optimized for fast software implementation
  - Character streaming (not bit)
  - 8-bit output
  - Former trade secret of RSADSI,
  - Reverse-engineered and posted to the net in 1994:
  - 2048-bit key
  - Used in many products until about 1999-2000

49

37

Other Symmetric Ciphers (RC4)

```
x=y=0;
while( length-- )
{     /* state[0-255] contains key bytes */
      sx = state[ ++x & 0xFF ];
      y += sx & 0xFF;
      sy = state[ y ];
      state[ y ] = sx;
      state[ x ] = sy;
      *data++ ^= state[ ( sx+sy ) & 0xFF ];
}
```

50

38

Other Symmetric Ciphers

- RC5 (Ron's Cipher #5)
  - Suitable for hardware and software
  - Fast, simple
  - Adaptable to processors of different word lengths
  - Variable number of rounds
  - Variable-length key (0-256 bytes)
  - Very low memory requirements
  - High security (no effective attacks, yet…)
  - Data-dependent rotations

52

39

Other Symmetric Ciphers

- RC5 single round pseudocode:

$$
\begin{aligned}
L &\leftarrow L \text{ XOR } R \\
L &\leftarrow L <<< R \\
L &\leftarrow L + \text{subkey}[2i] \\
R &\leftarrow R \text{ XOR } L \\
R &\leftarrow R <<< L \\
R &\leftarrow R + \text{subkey}[2i+1]
\end{aligned}
$$

52

40

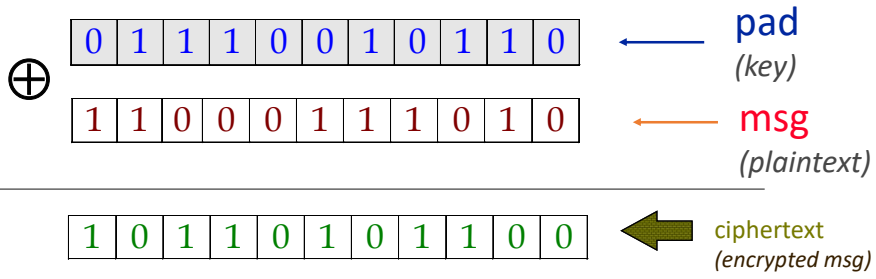
Reminder: World's Best Cipher!

69

41

One-Time Pad (OTP)

For each character:



70

42

One-Time Pad (cont.)

- Symmetric
- Pad is selected at random
- Pad is as long as plaintext
- Perfectly secure, but ...
- One time only:
  so sending the pad is just as hard as sending the msg

71

43

Other Old Symmetric Ciphers

Skipjack
- Classified algorithm originally designed for the NSA-sponsored Clipper chip
- declassified in 1998
- 32 rounds, breakable with 31 rounds
- 80 bit key, inadequate for long-term security

GOST
- GOST 28147, Russian answer to DES
- 32 rounds, 256 bit key
- Incompletely specified

47

44

Other Symmetric Ciphers

- IDEA (X. ILai, J. Massey, ETH)
  - Developed as PES (proposed encryption standard),
  - adapted to resist differential cryptanalysis
  - Gained popularity via PGP, 128 bit key
  - Patented (Ascom CH)
- Blowfish (B. Schneier, Counterpane)
  - Optimized for high-speed execution on 32-bit processors
  - 448 bit key, relatively slow key setup
  - Fast for bulk data on most PCs/laptops
  - Easy to implement, runs in ca. 5K of memory

48

45

Rijndael

- Three layers provide strength against known types of cryptographic attacks: Rijndael provides "full diffusion" after only two rounds
- Immune to:
  - Linear and differential cryptanalysis
  - Related-key attacks
  - Square attack
  - Interpolation attacks
  - Weak keys
- Rijndael has been "shown" *secure*:
  - No key recovery attacks faster than exhaustive search exist
  - No known symmetry properties in the round mapping
  - No weak keys identified
  - No related-key attacks: No two keys have a high number of expanded round keys in common

61

46

Breaking DES (Cryptanalysis)

Differential Cryptanalysis

- Looks for correlations in F()-function input and output

Linear Cryptanalysis

- Looks for correlations between key and cipher input and output

Related-key Cryptanalysis

- Looks for correlations between key changes and cipher input/output

Differential cryptanalysis discovered in 1990; virtually all block ciphers from before that time are vulnerable...

... except DES. IBM (and the NSA) knew about it 15 years earlier

34

47