TOPIC: 6.857 – Computer & Network Security

DATE: 3/4/13

FILE UNDER:

PAGE: L8.1

Admin: Pset #1 back at end of class

Today:
- Block ciphers
- DES (Data Encryption Standard)
- AES (Advanced Encryption Standard)
- Ideal Block Cipher
- Modes of Operation:
  - ECB
  - CTR
  - CBC
  - CFB
- IND-CCA Security defn
- UFE mode

Block ciphers:

fixed-length P, C, K

DES: $|P| = |C| = 64$ bits $|K| = 56$ bits

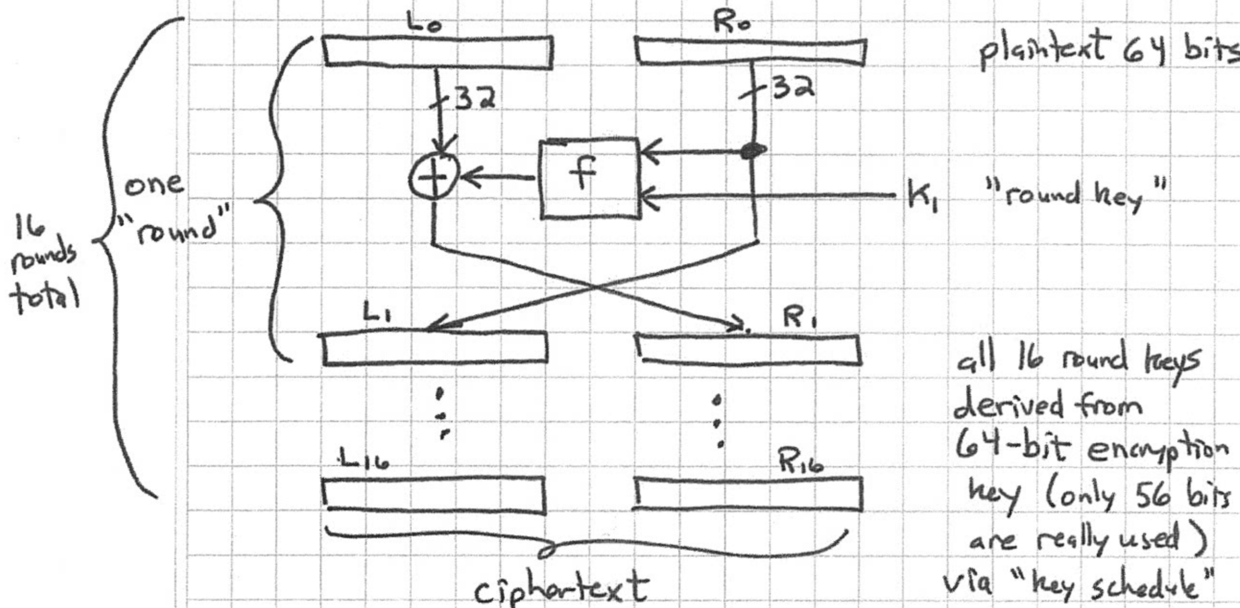
AES: $|P| = |C| = 128$ bits $|K| = 128, 192, 256$ bits

Use a "mode of operation" to handle variable-length input.

DES

"Data Encryption Standard"
Standardized in 1976. Now deprecated in favor of AES.

"Feistel structure":

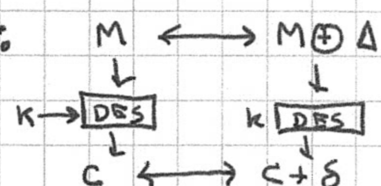


Note: Invertible for any f and any key schedule.

f uses 8 "S-boxes" mapping 6-bits ⇒ 4 bits nonlinearly.

Key is too short! (Breakable now quite easily by brute-force)

Subject to differential attacks:



2^{47} chosen pairs (Biham/Shamir)

Subject to linear attacks:

e.g. if $M_3 \oplus M_{15} \oplus C_2 \oplus K_{14} = 0$ (eqn on bits)
with prob $p = 1/2 + \varepsilon$

then need $1/\varepsilon^2$ samples to break (Matsui, 2^{43} PT/CT pairs)

TOPIC:

DATE:

FILE UNDER:

PAGE: L5.4

AES

"Advanced Encryption Standard" (U.S. govt)

Replaces DES

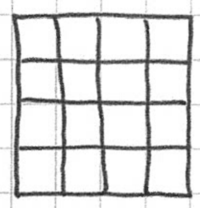
AES "contest" 1997–1999:
15 algorithms submitted: RC6, Mars, Twofish, Rijndael, ...
winner = Rijndael (by Joan Daemen & Vincent Rijmen, (Belgians))

Specs: 128-bit plaintext/ciphertext blocks
128, 192, or 256-bit key
10, 12, or 14 rounds (dep. on key length)

Byte-oriented design (some math done in Galois field $GF(2^8)$)

View input as 4x4 byte array:

$4 \times 4 \times 8 = 128$



For version with 128-bit keys, 10 rounds:

- Derive 11 "round keys", each 128 bits (4x4x byte)
- In each round:
  1. XOR round key
  2. Substitute bytes (lookup table)
  3. Rotate rows (by different amts)
  4. Mix each column (by linear opn)
- Output final state

See readings for details.
There are very fast implementations. Also Intel has put supporting hardware into its CPU's.

Security: Good; perhaps # rounds should be a bit larger...

For practical purposes, can treat AES as <u>ideal block cipher:</u>

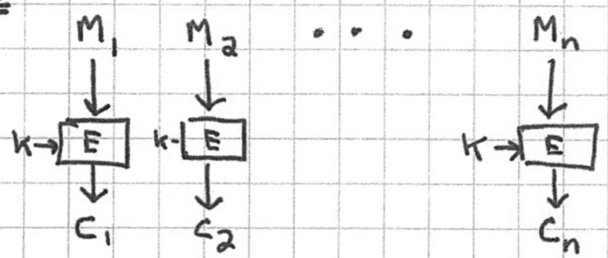
[For each key, mapping Enc(K, •) is a random independent permutation of $\{0,1\}^{128}$ to itself.

Modes of Operation:

How to encrypt variable-length messages? (using AES)

"ECB" = "Electronic code book"
"CTR" = "Counter mode"
"CBC" = "Cipher-block chaining" (& CBC-MAC)
"CFB" = "Cipher feedback"
... (others...)

ECB:



← divide data into b-bit blocks, where b = input block size

To handle data that is not a multiple of b bits in length:

- Append a "1" bit (<u>always</u>)
- Append enough "0" bits to make length a multiple of b bits.

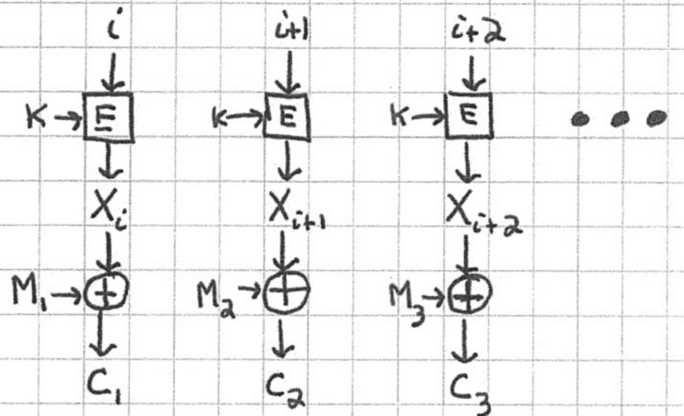
This gives <u>invertible</u> (1-1) "padding" operation.
Pad before encryption; unpad after decryption.

ECB preserves many patterns: repeated message blocks ⇒ repeated ciphertext blocks

ECB really only good for encrypting <u>random</u> data (e.g. keys)

CTR (Counter mode):

Generate a PR (pseudorandom) sequence by encrypting $i, i+1, \ldots$
XOR with message to obtain ciphertext.



Initial counter value can be transmitted first:

$i, C_1, C_2, \ldots$

Of course, no counter value should be re-used!

TOPIC:

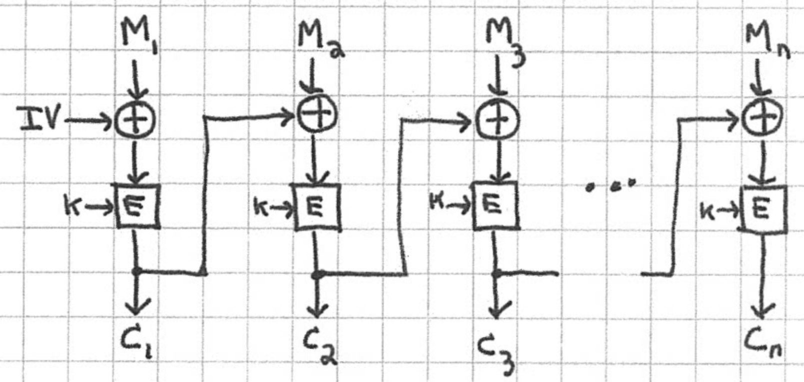
DATE:

FILE UNDER:

CBC (Cipher-block chaining):

Choose IV ("initialization value") randomly, then use each C_i is "IV" for M_{i+1}. Transmit IV with ciphertext:

$$IV, C_1, C_2, \ldots, C_n$$



Decryption easy, and parallelizable (∴ little error propagation)

Lookup "ciphertext stealing" for cute way of handling messages that are not a multiple of b bits in length. This method give ciphertext length = message length.

Last block C_n is the "CBC-MAC" (CBC Message Authentication code) for message M. [A <u>fixed</u> IV is used here.] The MAC is a "cryptographic checksum" (more later...) (If messages have variable length then key for last block should be different.)

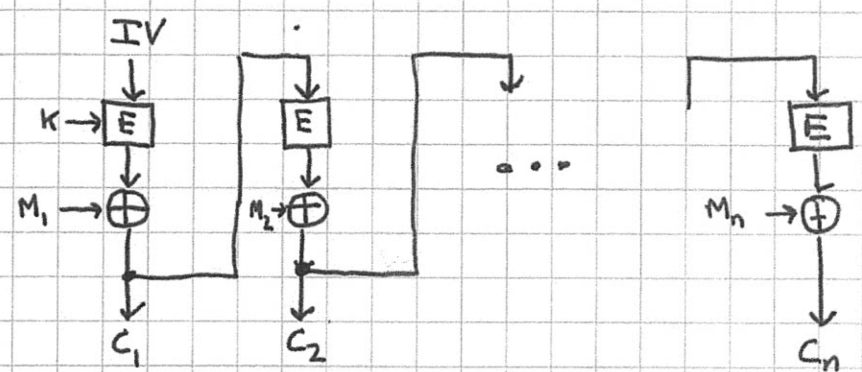
TOPIC:

DATE:

FILE UNDER:

CFB (Cipher feedback mode)

Similar to CBC mode. Uses random IV transmitted with ciphertext.



If M is not a multiple of b bits in length, can just transmit shortened ciphertext. (No need for ciphertext stealing.)