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:

\[ P \xrightarrow{\text{key } K} \text{Enc} \xrightarrow{} C \]

plaintext block

ciphertext block

fixed-length \( P, C, K \)

**DES:** \(|P| = |C| = 64 \text{ bits} \quad |K| = 56 \text{ bits} \)

**AES:** \(|P| = |C| = 128 \text{ bits} \quad |K| = 128, 192, 256 \text{ bits} \)

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":

![Diagram of Feistel structure]

16 rounds total

- Plaintext 64 bits
- All 16 round keys derived from 64-bit encryption key (only 56 bits are really used) via "key schedule"

Notes:
- Invertible for any f and any key schedule.
- F uses 8 "S-boxes" mapping 6 bits \( \Rightarrow \) 4 bits non-linearly.
- Key is too short! (Breakable now quite easily by brute-force)

Subject to differential attacks:

\[
M \xrightarrow{\oplus} M \oplus \Delta
\]

Subject to linear attacks:

\[
e.g., \quad M_3 \oplus M_{15} \oplus C_2 \oplus K_{14} = 0 \quad \text{([eqn on bits])}
\]

with prob \( p = 1/2 + \varepsilon \)

Then need \( 1/\varepsilon^2 \) samples to break (Matsui, \( 2^{43} \) PT/Ct pairs)
AES

"Advanced Encryption Standard" (U.S. gov't)

Replaces DES

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

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 4 x 4 byte array:
- \(4 \times 4 \times 8 = 128\)

For version with 128-bit keys, 10 rounds:
- Derive 11 "round keys", each 128 bits (4 x 4 x byte)
- In each round:
  1. XOR round key
  2. Substitute bytes (lookup table)
  3. Rotate rows (by different amounts)
  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 more rounds should be a bit larger...
For practical purposes, one can treat AES as an ideal block cipher: 

\[ \text{For each key, mapping } \text{Enc}(K, \cdot) \text{ is a random independent permutation of } \{0,1\}^b \text{ to itself.} \]

**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 (always)
- Append enough "0" bits to make length a multiple of \( b \) bits,

This gives invertible \((1+1)\) "padding" operation.

Pad before encryption; unpad after decryption.

ECB preserves many patterns: repeated message blocks

\( \Rightarrow \) repeated ciphertext blocks

ECB really only good for encrypting

random data (e.g., keys)
CTR (Counter mode):

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

\[ \begin{array}{ccc}
  i & i+1 & i+2 \\
  \downarrow & \downarrow & \downarrow \\
  K \rightarrow E & K \rightarrow E & K \rightarrow E \\
  \downarrow & \downarrow & \downarrow \\
  X_i & X_{i+1} & X_{i+2} \\
  \downarrow & \downarrow & \downarrow \\
  M_i \rightarrow \oplus & M_{i+1} \rightarrow \oplus & M_{i+2} \rightarrow \oplus \\
  \downarrow & \downarrow & \downarrow \\
  C_i & C_{i+1} & C_{i+2} \\
\end{array} \]

Initial counter value can be transmitted first:

\[ i, C_i, C_{i+1}, \ldots \]

Of course, no counter value should be re-used!
CBC (Cipher-block chaining):

Choose IV ("initialization value") randomly, then use each $C_i$ is “IV” for $M_i$. Transmit IV with ciphertext:

$$EV, C_1, C_2, \ldots, C_n$$

Decryption easy, and parallelizable (little error propagation)

Lookup “cipher text stealing” for cute way of handling messages that are not a multiple of 64 bits in length. This method gives ciphertext length = message length.

Last block $C_n$ is the “CBC-MAC” (CBC Message Authentication code) for message $M$. [A fixed 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.)
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.)