Admins

Today:

Project #2 due 3/13

Block ciphers:

- DES
- AES

Modes of operation

- Ideal cipher (ECB, CTR, CBC)
  - Desai's "UFE" mode

Readings:

- Ferguson Ch 3,
- Paar Ch 3, 4
- Katz Ch 5

Wikipedia: "Block cipher modes of operation"
  "Ciphertext stealing"
Block ciphers:

- Plaintext block: \( P \) → Enc (key \( k \) → C) → Ciphertext block

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

"Feistel structure":

16 rounds total

one "round"

plaintext 64 bits

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

Note: Invertible for any \( f \) and any key schedule.

\( f \) uses 8 "S-boxes" mapping 6-bits \( \rightarrow \) 4 bits nonlinearly.

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

Subject to differential attacks:
\[
\begin{align*}
M & \leftrightarrow M \oplus \Delta \\
\downarrow & \\
K & \xrightarrow{\text{DES}} K \\
\downarrow & \\
C & \leftrightarrow C \oplus \delta
\end{align*}
\]

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

Subject to linear attacks:

e.g. if \( M_3 \oplus M_{15} \oplus C_3 \oplus K_4 = 0 \) (eqn on bit 3)

with prob \( p = \frac{1}{2} + \varepsilon \)

then need \( \frac{1}{\varepsilon^2} \) samples to break (Matsui, \( 2^{43} \) PT/CT pairs)
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, (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 4x4 byte array:

\[
\begin{array}{cccc}
  &  &  & \\
  &  &  & \\
  &  &  & \\
  &  &  & \\
\end{array}
\]

4x4 x 8 = 128

For version with 128-bit keys, 10 rounds:

- Derive 11 “round keys”, each 128 bits (4x4xbyte)

  - 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 reading for details.

There are very fast implementations. Also Intel has put
supporting hardware into its CPU’s.

Security: Good? Perhaps 11 rounds should be a bit larger...
For practical purposes, can treat AES as ideal block cipher:

For each key, mapping Enc$(K_j, \cdot)$ 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\textsuperscript{+}" = "Cipher-block chaining" (\&CBC-MAC)
- "CFB" = "Cipher feedback"
- "" (others...)

**ECB:**

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 $i$, $i+1$, $i+2$, etc.
XOR with message to obtain ciphertext.

\[
\begin{array}{ccc}
K \rightarrow E & \downarrow & E \\
X_i \rightarrow & \downarrow & X_{i+1} \\
M_1 \rightarrow \oplus & \downarrow & M_2 \rightarrow \oplus \\
C_1 \rightarrow & \downarrow & C_2 \\
\end{array}
\]

"pad" like OTP

Initial counter value can be transmitted first:

\[
(i, C_1, C_2, \ldots)
\]

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