Administrivia: next week: Eran Tromer

Outline:

- AES
- Modes of operation
  - ECB
  - CTR
  - CBC
  - CFB
  - CTRX feedback
  - IND-CCA model
  - UFE

- MAC's
AES (Advanced Encryption Standard)

- Replaced DES (Data encryption std, adopted 1976, 56-bit key)
- "Contest" 1997-1999, 15 entries: RC6, Mars, Twofish, Rijndael,...
- Winner = Rijndael (by Joan Daemen & Vincent Rijmen, Belgians)
- Specs: 128-bit input/output blocks
  - 128, 192, or 256-bit key
  - 10, 12, or 14 rounds internally
- How it works (128-bit key, 10 rounds)
  - Byte-oriented spec
  - Does some math in $GF(2^8)$
  - View input as 4x4 array of bytes

- Derive 10 round keys, each 128-bits
- In each round:
  1. Add Round Key: Bytewise XOR round key into array
  2. SubBytes: Invert (over $GF(2^8)$) each elt of array
     (0 to 0)
     Apply affine transform to bits of each elt
     $ax + b$
     $a$ is $8x8$ matrix $\gamma$ over $GF(2)$
     $b$ is vector of size 8
  3. ShiftRows:
     $\begin{bmatrix}
        a & b & c & d \\
        e & f & g & h \\
        i & j & k & l \\
        m & n & o & p
     \end{bmatrix}
     \Rightarrow
     \begin{bmatrix}
        a & b & c & d \\
        f & g & h & e \\
        k & l & i & j \\
        p & m & n & o
     \end{bmatrix}
     \begin{bmatrix}
        0 \\
        1 \\
        2 \\
        3
     \end{bmatrix}$
(4) MixColumns
For each column \( \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} \)
treat it as polynomial \( a + bx + cx^2 + dx^3 \) over GF(2^n)
multiply by \( 3x^3 + x^2 + x + 2 \)
reduce modulo \( x^{n+1} \)
(since \( \gcd(x^{n+1}, 3x^3 + x^2 + x + 2) = 1 \), this is invertible).

[In last round: no MixColumns; instead use another AddRoundKey]

- Decryption: run backwards, invert each step
  - Every fast implementations: \( \sqrt{16 \text{ lookups in } 256 \times 32\text{-bit tables}} \) per round
    \[ \approx \text{gigabit/sec in hardware} \]
  - Security: # rounds could be larger (?)
$E_k : \{0, 1\}^b \rightarrow \{0, 1\}^b$ given

Modes of Operation (there are many! we'll see a few...)

- How to use given block cipher to encrypt data of arbitrary length?

**ECB** "Electronic Code Book"

- Divide $M$ into $b$-bit blocks, encrypt each separately

```
\begin{array}{cccc}
  M_1 & M_2 & M_3 & \cdots & M_n \\
  \downarrow & \downarrow & \downarrow & \ddots & \downarrow \\
  C_1 & C_2 & C_3 & \cdots & C_n \\
\end{array}
```

Patterns in data $\Rightarrow$ patterns in ciphertext (e.g. all-zero blocks...)

- Only really good for encrypting random data (e.g. keys)

To handle data not multiple of $b$ bits:

- Can "pad" by appending 1 & just enough 0's to give length a multiple of $b$.

$$01101 \overset{\text{pad}}{\longrightarrow} 10000...0$$

$\{0, 1\}^* \leftrightarrow (\{0, 1\}^b)^* \text{ (except } 0^b \text{)}$

Padding is invertible.

Pad before encrypting; unpad upon decryption... (true 0$^b$ as $\varepsilon$ ?)
Counter mode

- generate a PR sequence by encrypting $i$, $i+1$, ...

\[ \begin{array}{c}
k \quad E \quad X_i \\
M_1 \quad \oplus \quad C_1
\end{array} \]

- need way to share $i$ between sender & receiver
  could send $(i, C_1, C_2, \ldots)$

CBC

\[ \begin{array}{c}
IV \quad E \quad M_1 \\
K \quad C_1
\end{array} \]

\[ \begin{array}{c}
M_2 \quad E \quad M_3 \\
K \quad C_2
\end{array} \]

\[ \begin{array}{c}
M_n \quad E \\
K \quad C_n
\end{array} \]

- should generate IV randomly, xmit with cipher-text
  $IV, C_1, C_2, \ldots, C_n$

- Decryption easy ( overlapping! ) \Rightarrow little error propagation
- Last block $C_n$ is "CBC-MAC" for message $M_1, \ldots, M_n$ (with fixed IV)
Cipher feedback

IV

E

m_1 → C_1

E

m_2 → C_2

...
Do these modes give us what we want?

What do we want?

mode give us \( E'_k: \mathcal{X}_0,1^b \to \mathcal{X}_0,1^b \) (invertible)
based on block cipher \( E_k: \mathcal{X}_0,1^b \to \mathcal{X}_0,1^b \)

"IND-CCA" (Indistinguishability based on chosen-plaintext attack)
gives us standard to measure encryption modes.

Defined as game with adversary. Mode is secure (IND-CCA) if adversary can win with \( \Pr = \frac{1}{2} + \varepsilon \) for small \( \varepsilon \).

Phase I ("Find")

Adversary outputs two messages \( m_0 \), \( m_1 \) \( (m_0 \neq m_1) \) \( (1m_0 \| 1m_1) \)

\( + \) state info \( s \)

Adversary can use \( E_k, D_k \) freely during this phase

Phase II ("Guess")

\( d \leftarrow \mathcal{X}_0,1^b \) \( \) random bit chosen by examiner, unknown to adversary

\( y \leftarrow E_k(m_d) \) \( \) prepares challenge ciphertext

Adversary now given \( y, s \), access to \( E_k \), and

access to \( D_k \) (except on \( y \))

Adversary must produce guess \( \hat{d} \) for \( d \).

Adversary's advantage is \( \left| \Pr [\hat{d} = d] - \frac{1}{2} \right| \)

Scheme is secure against CCA attack if advantage is negligible.

Fact: to be IND-CCA secure, scheme must be randomized.

(since Adv can encrypt \( m_0, m \), himself... )
Previous modes are not IND-CCA secure!

(Decrypting 1st half of long ciphertext ⇒
1st half of message...)

Here is sketch of one IND-CCA secure method (UFE - Desai)

\[ M = \text{long input message of length } n \text{ b-bit blocks} \quad K = (k_1, k_2) \]

\[ r \xleftarrow{\$} \{0,1\}^b \quad \text{b-bit random value} \]

\[ \text{pad} = P_1, P_2, \ldots, P_n \]

\[ \text{where } P_i = E_{k_1}(r + i) \quad \text{(CTR mode PRG)} \]

\[ C = C_1 \ldots C_n \]

\[ \text{where } C_i = M_i \oplus P_i \]

Let \( X_0 = 0^b \)

\[ X_i = E_{k_2} (X_{i-1} \oplus C_i) \quad i \leq n-1 \quad \text{(CBC mode)} \]

\[ X_n = E_{k_3} (X_{n-1} \oplus C_n) \]

Output \((C_1, C_2, \ldots, C_n, \sigma)\)

\[ \sigma = r \oplus X_n \]

\[ \text{UFE} \equiv \]

"unbalanced Feistel encryption"

[only designed for confidentiality, so that null IND-CCA cares about...]
**MAC’s (Message Authentication Codes)**

- Bob recomputes $\text{MAC}_k(M)$ and rejects message as not from Alice, or damaged, if he gets different result.
- Integrity, not confidentiality
- Can layer on top of encryption (e.g., $M = \text{ciphertext for some other msg}$)
- MAC can be short, e.g. $b = 64$ bits
- Infeasible for an adversary to create new valid pair $M, \text{MAC}_k(M)$ (that Bob will accept), even after seeing many previous valid pairs (even for messages of his choice).
- CBC-MAC (use CBC mode, return last block)
- HMAC $\approx \text{hash}(K_1 \parallel \text{hash}(K_2 \parallel M))$

**Combined modes**

For confidentiality and integrity, you can encrypt then MAC, or use one of the combined modes of operation (CCM, EAX, OCB modes, ...)

**G.857 Rivest**

2/18/09 L5.9