Admin:

MIT Bitcoin Expo March 7-8

Today:

- Shamir's secret sharing
- Block ciphers
  - DES
  - AES
  - Modes of operation (ECB, CTR, CBC, CFB)

Readings:

Ferguson et al.: Chapter 3, Chapter 21, 9
Paar & Pelzl: Chapters 3, 4
Katz/Lindell: Chapters 6.2.3, 6.2.5, 13.3

Project idea:

- Do a source-code review of an open-source implementation of a crypto library or crypto product.
Key management

Start with "secret sharing" (threshold cryptography).

- Assume Alice has a secret $s$, (e.g., a key)

- She wants to protect $s$ as follows:
  
  She has $n$ friends $A_1, A_2, \ldots, A_n$

  She picks a "threshold" $t$, $1 \leq t \leq n$.

  She wants to give each friend $A_i$ a "share" $s_i$ of $s$, so that

  - any $t$ or more friends can reconstruct $s$

  - any set of $\leq t$ friends can not.

Easy cases:

$t = 1$:

$s_i = s$

$t = n$:

$s_1, s_2, \ldots, s_{n-1}$ random

$s_n$ chosen so that

$s = s_1 \oplus s_2 \oplus \ldots \oplus s_n$

What about $1 < t < n$?

ref: bitcoin "multisig"
Shamir's method ("How to Share a Secret", 1979)

Idea:
2 points determine a line
3 points determine a quadratic
... 
\( t \) points determine a degree \((t-1)\) curve

Let \( f(x) = a_{t-1} x^{t-1} + a_{t-2} x^{t-2} + \ldots + a_1 x + a_0 \)

There are \( t \) coefficients. Let's work modulo prime \( p \).

We can have \( t \) points: \((x_i, y_i)\) for \( 1 \leq i \leq t \)

They determine coefficients, and vice versa.

Polynomial Evaluation

\[ \{ (x_i, y_i) \} \rightarrow (a_{t-1}, a_{t-2}, \ldots, a_1, a_0) \]

Pt/value pairs \rightarrow Polynomial Interpolation

To share secret \( s \) (here \( 0 \leq s < p \)):

Let \( y_0 = a_0 = s \)

Pick \( a_1, a_2, \ldots, a_{t-1} \) at random from \( \mathbb{Z}_p \)

Let share \( s_i = (i, y_i) \) where \( y_i = f(i) \), \( 1 \leq i \leq n \).

Evaluation is easy.
**Interpolation**

Given \((x_i, y_i)\), \(1 \leq i \leq t\) (wlog)

Then \(f(x) = \sum_{i=1}^{t} f_i(x) \cdot y_i\)

where \(f_i(x) = \begin{cases} \frac{1}{\prod_{j \neq i} (x-x_j)} & \text{at } x = x_i \\ 0 & \text{for } x = x_j, j \neq i, 1 \leq j \leq t \end{cases}\)

Furthermore:

\[ f_i(x) = \frac{\prod_{j \neq i} (x-x_j)}{\prod_{j \neq i} (x_i-x_j)} \]

This is a polynomial of degree \(t-1\).

So \(f\) also has degree \(t-1\).

Evaluating \(f(0)\) to get \(s\) simplifies to

\[ s = f(0) = \sum_{i=1}^{t} y_i \cdot \frac{\prod_{j \neq i} (-x_j)}{\prod_{j \neq i} (x_i-x_j)} \]

**Theorem:** Secret sharing with Shamir's method is information-theoretically secure. Adversary with \(< t\) shares has no information about \(s\).

**Proof:** A degree \(t-1\) curve can go through any point \((0, s)\) as well as any given \(d\) pts \((x_i, y_i)\), if \(d < t\). \(\Box\)

**Refs:** Reed-Solomon codes, erasure codes, error correction, information dispersal (Rabin).
Block ciphers:

\[
P \downarrow \quad \text{plaintext block} \\
\text{key } K \rightarrow \text{Enc} \\
\downarrow \\
C \\
\text{fixed-length } P, C, K
\]

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

"Feistel structure":

16 rounds total

16 rounds total

Note: 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:

Subject to linear attacks:

\[
e.g. \quad M_3 \oplus M_{15} \oplus C_2 \oplus K_1 = 0 \quad \text{(eqn on bits)}
\]

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

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, (Belgians))

Specs:
- 128-bit plaintext/cipher text 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 $4 \times 4$ byte array:

\[
\begin{array}{cccc}
4 \times 4 \times 8 &=& 128
\end{array}
\]

For version with 128-bit keys, 10 rounds:

- Derive 11 "round keys", each 128 bits ($4 \times 4 \times $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..