Lecture 1: Security

\[ \text{Alice} \rightarrow \text{Bob} \]
\[ \text{Eve} \]

Goals of security:

- **Confidentiality**: data is kept secret from unintended listeners (eavesdroppers).
- **Message Integrity**: the message you receive is the one that was sent.
- **Authentication**: the person you’re communicating with is who you think it is.

\[ \text{Confidentiality} \]

**Symmetric encryption**

Gen \( k \): generate key \( k \)

\[ c = \text{Enc}_k(m) \]

\[ m = \text{Dec}_k(c) \]

Satisfies \( \text{Dec}(\text{Enc}(m)) = m \)

**Def. "perfect secrecy"**

\[ \forall m \in \mathcal{M} \quad \forall c \in \mathcal{C}, \quad \Pr[M = m | C = c] = \Pr[M = m] \quad \text{equivalently} \quad \Pr[C = c | M = m] = \Pr[C = c] \]

**Lemma "perfect indistinguishability"**

An encryption scheme is perfectly secret \( \iff \)

\[ \forall m, m' \in \mathcal{M}, \forall c \in \mathcal{C}, \quad \Pr[C = c | M = m] = \Pr[C = c | M = m'] \]

**Proof**

\[ \Pr[C = c | M = m_0] = \Pr[C = c] = \Pr[C = c | M = m_1] \]

\[ \iff \text{Fix prob. dist. over } \mathcal{M}, \text{ arbitrary } m_0 \text{ and } c. \]

\[ \Pr[C = c] = \sum_m \Pr[C = c | M = m] \cdot \Pr[M = m] \]

\[ = \sum_m \Pr[C = c | M = m_0] \cdot \Pr[M = m] \]

\[ = \Pr[C = c | M = m_0] \cdot \sum_m \Pr[M = m] \]

\[ = \Pr[C = c | M = m_0] \quad \text{true for any } m_0 \]
One-Time Pad

Gen: \( k \in \{0,1\}^n \) uniform

Enc: \( k \in \{0,1\}^n, m \in \{0,1\}^* \) \( c = k \oplus m \)

Dec: \( m = c \oplus k \)

Thm: OTP is perfectly secret

\[
\Pr[c = c' | m = m'] = \Pr[c \oplus k = c' \oplus k] = \Pr[m \oplus k = c] = \Pr[k = m \oplus c] = \frac{1}{2^n}
\]

holds for all \( m, m', c \)

(But not if you misuse keys!)

Thm: Let \( \{\text{Gen}, \text{Enc}, \text{Dec}\} \) be a perfectly secret encryption scheme.

Then \( |\{\text{Enc}\}| > |\{\text{Enc}\}| \)

PF: Show \( |\{\text{Enc}\}| < |\{\text{Enc}\}| \) scheme is not perfectly secret. Assume \( |\{k\}| < |\{m\}| \).

Fix uniform dist. over \( \{m\} \).

Let \( M(c) = \{m \in \text{Dec}_k(c) \text{ for some } k \in \{k\}\} \)

\( |M(c)| \leq |\{k\}| = |\{m\} \times \{\text{Enc}\}) \text{ s.t. } m \in M(c) \).

\[ \Pr[m = m' | c = c'] = 0 \neq \Pr[m = m'] \]

Def: "computational indistinguishability"

A adversary \( V \)\n
\[ V \text{ inputs } (m, m') \]

\[ \text{ chooses } b' \text{ probabilistic polynomially } \]

\[ \text{ outputs } b' \]

A succeeds if \( b = b' \)

scheme is indistinguishable if

\[ \Pr[A \text{ succeeds}] \leq \frac{1}{2} + \varepsilon(n) \]

\( \varepsilon(n) \) negligible: \( \forall \text{ poly } p(), \exists \text{ N s.t. } A \forall n > N \)

\[ \varepsilon(n) < \frac{1}{2^n}, 2^{-n}, \frac{1}{n^{100}} \]

Equivalent to semantic security.
Pseudorandomness

Def: "pseudorandom generator"
G deterministic, poly-time algorithm
G : \{0,1\}^n \to \{0,1\}^{\Omega(n)}

Expansion: \Omega(n) \gg n

Pseudorandomness: A prob. poly. time distinguisher D:
| Pr[D(r)=1] - Pr[D(G(s))=1] | < \epsilon(n)
r \text{ uniform in } \{0,1\}^n
s \text{ uniform in } \{0,1\}^n

(Easy: expander + time distinguisher: compute all G(s) in 2^n time, output 1 if inputted.)
Can convert to variate lengths.

Stream cipher encryption
Gen: k \in \{0,1\}^n \text{ uniformly random}
Enc: c = G(k) \oplus m \text{ (m \in \{0,1\}^n)}
Dec: m = G(k) \oplus c

Thm: If G is PRG, then above has indistinguishable encryption.
Proof: Show if A can distinguish encryptions, then G is not PRG:
\[
\begin{array}{c}
W \xrightarrow{P} c \xrightarrow{A} b' \\
1 \text{ if } b' = b \\
0 \text{ otherwise.} \\
D(r) = \frac{1}{2} \text{ random} \\
D(G(s)) = \frac{1}{2} \pm \epsilon(n) \\
|D(r) - D(G(s))| > \epsilon(n)
\end{array}
\]

Real-life example: RC4 (we'll need next time)

Multiple encryptions are insecure:
- vectors (Enc_k(0^n), Enc_k(0^n)) and (Enc_k(1^n), Enc_k(1^n)) easily distinguished.
- All attacks: compute \text{ m}_0 \oplus \text{ m}, etc.

\Rightarrow need IV: initialization vector
Enc_k(m) = (IV, G(k, IV) \oplus m)
"chosen plaintext attack"

Def

A adversary

______\n| m, m' |
\abar{\text{Gen} (1^n) \rightarrow}
| m' \leftarrow \text{Encr}(m) |
| m \leftarrow \text{Encr}(m') |
| m \leftarrow \text{Encr}(m) |
\bara{n} \rightarrow

Real-world examples: adversary causes you to click on a link

Pseudorandom functions

Def "pseudorandom function"

\[ F : \{0,1\}^n \times \{0,1\}^* \rightarrow \{0,1\}^* \]

\[ \forall \text{ poly-time distinguisher } D \quad \text{prob.} \] 

\[ \left| \text{Pr} \left[ D^{F_k(1^n)} = 1 \right] - \text{Pr} \left[ D^{f_r(1^n)} = 1 \right] \right| \leq \epsilon(n) \]

Can define an actual random function using a look-up table, but herexponential length

Def "pseudorandom permutation"

( a prp is also a prf )

Abstraction of block ciphers like DES, AES

Using block ciphers:

\[ \text{Gen} : \text{choose } k \in \{0,1\}^n \text{ uniformly random} \]

\[ \text{Enc} : \text{choose } r \in \{0,1\}^m \text{ uniformly random} \]

\[ c = (r, F_k(r) \oplus m) \]

\[ \text{Dec} : \text{input } k, c = (r, s) \]

\[ m = F_k(r) \oplus s \]

Can reduce to pseudorandomness at \( f \)
DES

IBM, USA
FIPS standard in 1977
56-bit key length
1997 - DESchallenger, solved by distributed computation
1998 - Deep Crack: $2^{39}$ trials in 56 hours

Solutions:

2DES?
\[ F_{k_1 k_2} (x) = F_{k_2} (F_{k_1} (x)) \]

problem: meet-in-the-middle attack

Adversary sets \((x, y)\) with \(F_{k_2} (F_{k_1} (x))\)

1. Compute \(z_i = F_{k_1} (x) \oplus k_i\) and store \((z_i, k_i)\) in \(L_1\)
2. Compute \(z_i = F_{k_2}^{-1} (y)\) and store \((z_j, k_j)\) in \(L_2\)
3. Sort \(L_1, L_2\) and see any matching \(z_i, z_j\)

If result isn't unique, repeat with \((x, y)\)

3DES

\[ F_{k_1 k_2 k_3} (x) = F_{k_3}^{-1} (F_{k_2}^{-1} (F_{k_1} (x))) \]

why alternate? so \(k_1 = k_2 = k_3 \Rightarrow f_{k_1} = f_{k_3}\)

with 2 keys

\[ F_{k_1 k_2} = F_{k_2} (F_{k_1}^{-1} (F_{k_1} (x))) \]

popular in finance

DES x "Whitening"

\[ F_{k_1 k_2 k_3} (x) = k_3 \oplus F_{k_2} (x \oplus k_1) \]

AES

Chosen in 2000
Rijndael
John Daemen, Vincent Rijmen
(Daemen also co-designed Keccak)

128, 192, 256-bit versions

more secure?
Modes of Operation

Encrypt arbitrary-length messages using block ciphers

**ECB mode** "Electronic Code Book"

\[ m = m_1, m_2, \ldots, m_e \]

\[ c = F_k(m_1), F_k(m_2), \ldots, F_k(m_e) \]

deterministic, messages can be replayed and distinguished

commonly used in practice, should never be used

**CBC mode** "Cipher Block Chaining" is most recommended

1. Choose IV or begin \( c_0 = IV \)
2. \( c_{i+1} = F_k(c_i \oplus m_i) \)
3. output \( (c_0, c_1, \ldots, c_e) \) or use \( E_k \) (message number) "nonce-secured"

IV needs to be random \( \implies \) CPA-secure

**CTR mode** "Counter"

1. Choose random IV, \( ctr \)
2. \( c_i = F_k(ctr + i) \)
3. \( c = c_i \oplus m_i \)

CPA-secure, easy to pre-process, allows decryptions of single bytes
subjected to problem of randomness failure

---

**Chosen Ciphertext Attackers (CCA security)**

A

\[ \begin{aligned}
&\text{oracle access to } E_k() \\
&\text{oracle } D_k() \\
&\overset{m_0,m_i}{\longrightarrow} \\
&c = E_k(m_b) \quad b \in \{0,1\} \\
&\overset{\text{category } D_k(c)}{\longrightarrow} b' \\
&\quad \text{success} \quad b' = b
\end{aligned} \]

---

all encryptions we've seen so far are malleable
If \( E_k(m) = (r, F_k(r) \oplus m) \)
adversary sets \( m_0 = 0^n \), \( m_1 = 1^n \)
or it transmits nonce \( \oplus \)
flips \( b \) bit of \( c \), asks for decryption

---

realistic threat: Bleichenbacher
modified ciphertext, used different responses to decryption as an oracle
Integrity

message authentication codes (MAC)

Gen: generate key k
   input: k
Mac: output tag t = Mac_k(m)

Verify: input key k, message m, tag t
   output 1 if valid, 0 otherwise

Verify_k (m, Mac_k(m)) = 1

existential unforgeability against adaptive chosen message attacks

A \{  \}
V

k = Gen(1^n)

\forall (m, t) \rightarrow success \land Verify_k(m, t) = 1

Def: (Gen, Mac, Verify) secure if \forall \text{prob. poly-time adversaries } A

Pr[success = 1] \leq \epsilon(n)

replay attacks: adversary resends messages.
prevent: include sequence numbers (outsourced cryptos)

MAC construction F [PRF]

Gen: k \in \{0,1\}^n
Mac: t = F_k(m)
Verify: t = F_k(m)

This construction is existentially unforgeable against adaptive chosen message attacks.

D has F- oracle O
1. Run A.
2. Use A to get Mac(m), output O(m)
3. A outputs (m, t)
4. Check that O(m) = t and A always succeed O(m)
Hash function
Gen output key $k$ of length $n$
$H : \{0,1\}^n \rightarrow \{0,1\}^n$

Notions of security:

collision resistance:
Adversary given $s$, finds input $x, x' \text{ s.t. } H^s(x) = H^s(x')$

second pre-image resistance:
Adversary given $s, x$
finds input $x' \text{ s.t. } H^s(x) = H^s(x')$

pre-image resistance:
Adversary given $s, y = H^s(x)$ (but not $x$)
finds $x' \text{ s.t. } H^s(x') = y$

Birthday Attacks

Attack 1: Choose arbitrary inputs $x_1, \ldots, x_9$
1. Compute $y_i = H(x_i)$, store
2. Success if collision found

$\text{Thm: } \Pr(\text{collision after } \Theta(2^{n/2}) \text{ inputs}) = \frac{1}{2}$

$\Pr(\text{k collisions}) = \prod_{i=1}^{k-1} \frac{1}{2^{i-1}} e^{-\frac{k^2}{2n}}$

$\Pr(\text{distinct}) = e^{-\frac{k^2}{2n}}$

$\Pr(\text{collision}) = 1 - \Pr(\text{distinct}) > 1 - e^{-\frac{k^2}{2n}}$

Attack 1 requires $O(2^{n/2})$ memory/hash evaluations
$O(2^{n/2})$ time (sorting)

Improvement: Floyd cycle-finding algorithm
Choose $x_0$, compute $x_i = H(x_{i-1})$, $x_{2i} = H(H(x_{2i-1}))$
Find collision w.p. $\frac{1}{2}$ in $O(2^{n/2})$ steps
constant memory
Converting fixed-length hash functions to arbitrary length

Input $h(x)$ fixed-length

1. Compute $H_i = h(H_{i-1}, m_i)$ \( H_0 \) fixed
2. Output $H_k$

How not to construct arbitrary-length hash functions:

1. Set $h(x) = AES_0(x)$
2. Set $H_0 = 000000...$
3. Let $H_i = AES_0(H_i \oplus m_i)$

Here is a collision: $m = m_1, m_2$  
$H_1, H_2$ unknown:

\[ \begin{align*}
    H(m_1) & = H(m) \\
    m_1' & = m \oplus H_1 \\
    m_2' & = H_2 \oplus m_2 \oplus H_1
\end{align*} \]

Merkel-Damgård Construction

$h$ = fixed length collision-resistant hash function
input of length $2^n$, output $h(x)$

1. Input $m = m_1, \ldots, m_k$
2. Pad $m_k$ to length $2^n$ with zero
3. Append $m_k \leftarrow \text{len}(m) \cdot \text{8-bit representation}$
4. Set $z_0 = 0^8$
5. Compute $z_i = h(z_{i-1}, \| m_i)$

MDS \to totally broken
SHA-1 \to collisions
SHA-2 \to pervasive (SHA-256, SHA-512)
SHA-3 \to compatible with SHA-2