CS 88: Security and Privacy

12: Symmetric Key Cryptography

03-05-2024

slides adapted from Dave Levine, Jonathan Katz, Kevin Du
Multiple message secrecy

We are not going to formally define a notion of multiple-message secrecy
• Instead, define something stronger: **security against chosen-plaintext attacks (CPA-security)**
• *minimal notion of security an encryption scheme should satisfy*
Security against Chosen Plaintext Attack: Impossible?

It really is a problem if an attacker can tell when the same message is encrypted twice!

This attack only works if encryption is deterministic!
Random Functions

Out of all possible function mappings between $X$ and $Y$ we choose one uniformly at random.

- e.g. for a 2 bit string mappings between $X$: $\{0, 1\}^2$ and $Y$: $\{0, 1\}^2$
- one possible mapping that we could choose:

<table>
<thead>
<tr>
<th>$x$</th>
<th>00</th>
<th>01</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F(x)$</td>
<td>01</td>
<td>11</td>
<td>00</td>
<td>10</td>
</tr>
</tbody>
</table>

Properties of function $F(X)$ chosen uniformly at random:

- for any given $x \in X$, the probability that $F(x) = y$ is $1/2^n$
- in our example example:
  - given $x \in X$, the probability that $F(x) = 1/2^2 = \frac{1}{4} = 0.25$
- $F(x)$ property:
  - if $x$ changes by one bit to give $x'$ then
  - $F(x')$ is completely independent of $F(x)$. 
Random Permutations

- Variant of random function is random permutation
  - treat them equivalently for our purposes.

- E.g.: random permutation over bit strings of length 2
  Encryption: \( \{0, 1\}^2 \rightarrow \{0, 1\}^2 \)

<table>
<thead>
<tr>
<th>x</th>
<th>F(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>01</td>
</tr>
<tr>
<td>01</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>00</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
</tr>
</tbody>
</table>

Important Property of the Random Permutation:

A permutation is invertible (bijective) function

Given \( F(x) \) it is impossible to determine \( x \) without resorting to a brute force attack.

If \( |X| \) is very large? brute force not possible by an efficient (probabilistic polynomial time) attacker.
What we have, ideally: Random Functions

Consider the set of all permutations $F_k: X \rightarrow X$

Think of $X$ as all 128-bit bit strings

<table>
<thead>
<tr>
<th>$f_1$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_2$</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_{</td>
<td>X</td>
<td>}$</td>
<td>7</td>
<td>9</td>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>

If you know $k$, then $F_k(x)$ is trivial to invert

If you don’t know $k$, then $F_k(x)$ is one-way

One-way function
What we have, ideally: Random Functions

Consider the set of all permutations $F_k: X \rightarrow X$

Think of $X$ as all 128-bit bit strings

Shared secret: index $k$ chosen uniformly, at random

Without knowing $k$, Eve learns nothing about $m$

$k$ is our key!
What we have, approximately: Pseudo-Random Functions

In essence, this protocol is saying “Let’s use the $i$th permutation function”

Infeasible to store all permutation functions – so instead cryptographers construct pseudorandom functions
A Perfectly Secure Encryption Scheme

*Regardless of any prior information the attacker has about the plaintext, the ciphertext observed by the attacker should leak no additional information about the plaintext.*

Alice can only observe one ciphertext going over the network.
Computational Secrecy

Would be okay if a scheme leaked information with a tiny probability to eavesdroppers with bounded computational resources.

- **Allowing security to fail with a tiny probability** (negligible in key length $n$)
  - how tiny is tiny? $2^{-60}$: probability of an event occurring every 100 billion years!

- **Only consider efficient attackers** (bounded in polynomial time by key length)
  - attackers that can brute-force the key space in bounded time.
  - try testing $2^{112}$ keys? Would take a supercomputer since Big Bang!
  - modern key space? $2^{128}$ or more!
BLACKBOXES

To this end, we’ll cover several “blackboxes”: what properties do they provide, and how can we responsibly put them together.

- Block ciphers
- MACs
- Hash functions
- Public key crypto
Scenarios and Goals

Confidentiality
- Keep others from reading Alice’s messages/data

Integrity
- Keep others from undetectably tampering with Alice’s messages/data

Authenticity
- Keep others from undetectably impersonating Alice (keep her to her word too!)

Block Ciphers

Alice
- Public network

Bob
- Disk
Block Ciphers

**Encryption**

Encryption Function: \( E: \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n \)

Fix the key \( K \), then, \( E_k: \{0, 1\}^n \rightarrow \{0, 1\}^n \)

- **plaintext size**: \( n \)
- **ciphertext size**: \( n \)

\( E_k \): permutation on \( n \)-bit strings.

- invertible (bijective function) given the key

Once the key is fixed: \( E(k,m) \) is indistinguishable from a function chosen uniformly at random from all possible functions between block-sized binary strings.
Once the key is fixed: $E(k,m)$ is indistinguishable from a function chosen uniformly at random from all possible functions between block-sized binary strings.

Attacker has no way of knowing which random function was chosen to permute the plaintext to the ciphertext.
Block Ciphers

**DECRIPTION**

Inverse mapping of the permutation is the decryption algorithm, given the key
\[ D_k(E_k(M)) = M \]

without the key: best attack is a brute force exhaustive search over the entire key space!

Attacker has no way of knowing which random function was chosen to permute the plaintext to the ciphertext
**Block Ciphers**

**Encryption**

Key $K \rightarrow E \rightarrow c$

AES key sizes: 128, 192, 256

**Decryption**

$K \rightarrow D \rightarrow m$

Plaintext (“message”)

Same fixed block size (AES: 128 bits, 3DES: 64 bits)

Ciphertext

**PROPERTY:**

Small changes to the inputs cause big changes in the output

**Confusion:** Each bit of the ciphertext should depend on each bit of the key

**Diffusion:** Flipping a bit in $m$ should flip each bit in $c$ with $Pr = 1/2$

$\{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$
Chosen Plaintext Attack

Eve: Attacker

Bob: Challenger

repeat

M
Enc(K, M)

M₀ and M₁
Enc(K, Mᵦ)

repeat

M = M₀ In Step 2?

Enc(K, M)
Problem #1: Block Ciphers Are Deterministic

Property:

Block ciphers are deterministic
For a given $m$ and $K$, $E(K,m)$ always returns the same $c$

An eavesdropper could determine when messages are re-sent

A Fix:

$m \oplus r$ is the same size as $m$
Choose random $r$
Send $c$ and $r$

Also known as an Initialization Vector or Nonce
Initialization Vector (nonce)

Choose random $r$

$K \rightarrow E$

$m \oplus r$

$c$

Send $c$ and $r$

**Random:** Must send $r$ with the message
This is good if messages can be reordered

**Counter:** Don’t need to send $r$; the receiver can infer it from the message number
This is good if messages are delivered in-order
Problem #2: Block Ciphers have fixed size

Fixed block size $m$

If we want to encrypt a message larger than the block size (128 bits), we simply break up the message into block-size length pieces...

$$\text{m} = m_1 \, m_2 \, m_3 \, m_4 \, \ldots \, m_n$$

...and encrypt each block
Modes of Encryption: Electronic Codebook Mode (ECB)

Each block in AES 128 bits

Encryption:
inputs: plaintext: m, key: k,
ciphertext: c[i] = E(k, m[i])

Decryption:
inputs: ciphertext: c, key: k,
plaintext: m[i] = D(k, c[i])

spot the problem?
NEVER use ECB
(but over 50% of Android apps do)
Modes of Encryption: Cipher Block Chaining Mode (CBC)

Encryption
input: plaintext $m$, key $k$, initialization vector IV
$c[0] = IV$
$c[i] = E(k, m[i] \oplus c[i-1])$ for $i \geq 1$

Decryption
input: ciphertext $c$, key $k$, initialization vector IV
$m[i] = D(k, c[i]) \oplus c[i-1]$
Modes of Encryption: Cipher Block Chaining Mode (CBC)

Security

Input to the Encryption algorithm at each step is extremely likely to be different from the previous step.

Performance

Encryption: Not Parallelizable

Decryption: Parallelizable recovering \( m[i] \) does not require \( m[i-1] \). Only requires \( c[i-1] \) which is already known.
Symmetric Key Cryptography

Confidentiality
- Keep others from reading Alice’s messages/data

Integrity
- Keep others from undetectably tampering with Alice’s messages/data

Authenticity
- Keep others from undetectably impersonating Alice (keep her to her word too!)

Block Ciphers

Limitations?
- what if Eve modifies the packet in transit?
- How do we share keys?