Chosen Plaintext Attacks (CPA)

Goals

New Attacks! Chosen Plaintext Attacks (often CPA) is when Eve can choose to see some messages encoded. Formally she has Black Box for ENC_k.

We will:

1. Define Chosen Plaintext Attack for perfect security.
2. Define Chosen Plaintext Attack for computational security.

Perfect CPA-Security via a Game

$\Pi = (\text{GEN}, \text{ENC}, \text{DEC})$ be an enc sch, message space $\mathcal{M}$.
Game: Alice and Eve are the players. Alice has full access to Π.
Eve has access to ENC_k.

1. Alice $k \leftarrow \mathcal{K}$. Eve does NOT know k.
2. Eve picks $m_0, m_1 \in \mathcal{M}$ Eve has black box for ENC_k.
3. Alice picks $m \in \{m_0, m_1\}$, $c \leftarrow ENC_k(m)$
4. Alice sends c to Eve.
5. Eve outputs m_0 or m_1, hoping that her output is $DEC_k(c)$.
6. Eve wins if she is right.

Note: ENC_k is randomized, so Eve can't just compute $ENC_k(m_0)$ and $ENC_k(m_1)$ and see which one is c.
Does Eve has a strategy that wins over half the time?

Perfect CPA-Security

- Π is *secure against chosen-plaintext attacks (CPA-secure)* if for all Eve.

$$Pr[\text{Eve Wins}] \leq \frac{1}{2}$$

Eve always wins if ENC_k is Deterministic

1. Eve picks m_0, m_1. Finds $c_0 = ENC_k(m_0)$, $c_1 = ENC_k(m_1)$.
2. Alice sends Eve $c = ENC_k(m_b)$. Eve has to determine b.
3. If $c = c_0$ then Eve sets $b' = 0$, if $c = c_1$ then Eve sets $b' = 1$.

Upshot: ALL deterministic schemes are CPA-insecure.

Comp CPA-Security

$\Pi = (\text{GEN}, \text{ENC}, \text{DEC})$ be an enc sch, message space $\mathcal{M}$.
n is a security parameter.
Game: Alice and Eve are the players. Alice has full access to Π.
Eve has access to ENC_k.

1. Alice $k \leftarrow \mathcal{K} \cap \{0,1\}^n$. Eve does NOT know k.
2. Eve picks $m_0, m_1 \in \mathcal{M}$, $|m_0| = |m_1|$
3. Alice picks $m \in \{m_0, m_1\}$, $c \leftarrow ENC_k(m)$
4. Alice sends c to Eve.
5. Eve outputs m_0 or m_1, hoping that her output is $DEC_k(c)$.
6. Eve wins if she is right.

Does Eve has a strategy that wins over half the time?

Comp. CPA-Security

- Π is CPA-Secure if for all Polynomial Prob Time Eves, there is a neg function $\epsilon(n)$ such that

$$\Pr[\text{Eve Wins}] \leq \frac{1}{2} + \epsilon(n)$$

Randomized Encryption

1. Any Deterministic Encryption will NOT be CPA-secure.
2. Hence we have to use Randomized Encryption.
3. The issue is *not* an artifact of our definition: Even being able to tell if two messages are the same is a leak.
4. Next three slides defines Det Encryption, Keyed Functions, Rand Encryption.

Deterministic Encryption (for contrast)

n is a security parameter. A Deterministic Private-Key Encryption Scheme has message space $\mathcal{M}$, Key space $\mathcal{K} = \{0,1\}^n$, and algorithms (GEN, ENC, DEC):

1. GEN generates keys $k \in \mathcal{K}$.
2. ENC_k encrypts messages, DEC_k decrypts messages.
3. $(\forall k \in \mathcal{K})(\forall m \in \mathcal{M}), DEC_k(ENC_k(m)) = m$

Keyed functions

1. Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be an efficient, deterministic algorithm
2. Define $F_k(x) = F(k, x)$
3. The first input is called the key
4. Choosing a uniform $k \in \{0,1\}^n$ is equivalent to choosing the function $F_k : \{0,1\}^n \to \{0,1\}^n$

Note: In literature and the textbook Keyed functions k, x can be diff sizes, but we never do.

Keyed functions

1. Let $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ be an efficient, deterministic algorithm
2. Define $F_k(x) = F(k,x)$
3. The first input is called the key
4. Choosing a uniform $k \in \{0,1\}^n$ is equivalent to choosing the function $F_k : \{0,1\}^n \to \{0,1\}^n$

Note: In literature and the textbook Keyed functions k, x can be diff sizes, but we never do. They are wrong, we are right.

Randomized Encryption

A Randomized Private-Key Encryption Scheme has message space $\mathcal{M}$, Key space $\mathcal{K} = \{0,1\}^n$, algorithms (GEN,ENC,DEC).

1. GEN generates keys $k \in \mathcal{K}$ (Think: picking an F_k rand.)
2. ENC_k: on input m it picks a rand $r \in \{0,1\}^n$ and outputs $(r, m \oplus F_k(r))$.
3. $DEC_k(r,c) = c \oplus F_k(r)$.

Note:

1. $ENC_k(m)$ is not a function- it can return many different pairs.
2. Easy to see that Encrypt-Decrypt works.
3. Rand Shift is *not* an example, but is the same spirit.
4. General definition that encompasses Rand Shift: Can replace $\oplus$ with any invertible operation.

Pseudorandom functions

Pseudorandom functions

- Informally, a pseudorandom function "looks like" a random (i.e. uniform) function
- Can define formally via a Game. We won't. Might be HW or Exam Question.
- From now on PRF means Pseudorandom function.
- Will actually get Psuedorandom Permutations for real world use.

Constructing a CPA-Secure Encryption

Theorem: If F_k is a PRF then the following encryption scheme is CPA-secure.

1. GEN generates keys $k \in \mathcal{K}$ (Think: picking an F_k rand.)
2. ENC_k: on input m it picks a rand $r \in \{0,1\}^n$ and outputs $(r, m \oplus F_k(r))$.
3. $DEC_k(r,c) = c \oplus F_k(r)$.

Proof Sketch: If not CPA-secure then F_k is not a PRF.

A Real World (probably) PRF: Substitution-Permutation Networks (SPNs)

Recall...

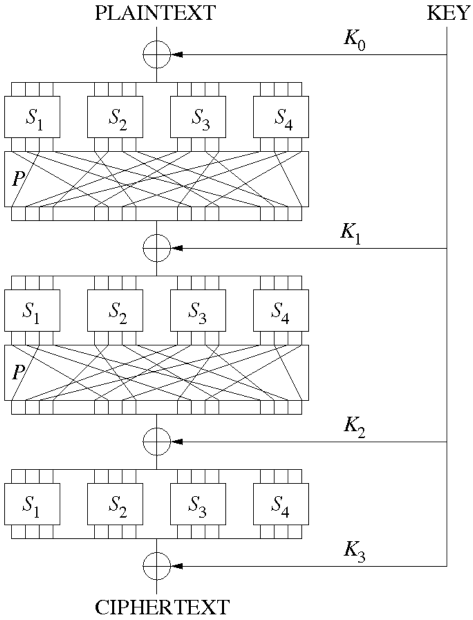
- Want keyed permutation

$$F : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^\ell$$

n = key length, ℓ = block length

- Want F_k (for uniform, unknown key k) to be indistinguishable from a uniform permutation over $\{0,1\}^\ell$

Substitution-Permutation Networks (SPNs)



Substitution-Permutation Networks (SPNs)

For r-rounds:
Key will be $k = k_1 \cdots k_r$ and k_i's will be used along with public S-box to create perms.

- $f_{k_i}(x) = S_i(k_i \oplus x)$, where S_i is a public permutation
- S_i are called "S-boxes" (substitution boxes)
- XORing the key is called "key mixing"
- Note that SPN is invertible (given the key)

S-Boxes are HARD to Create

Building them so that an SPN is a PRF is a major challenge.

Titles of Papers that tried:

The Design of S-Boxes by Simulated Annealing

A New Chaotic Substitution Box Design for Block ciphers

Perfect Nonlinear S-Boxes

S-Boxes are HARD to Create

Building them so that an SPN is a PRF is a major challenge.

Titles of Papers that tried:

The Design of S-Boxes by Simulated Annealing

A New Chaotic Substitution Box Design for Block ciphers

Perfect Nonlinear S-Boxes

If you type in S-Boxes into Google Scholar how many papers to you find?

S-Boxes are HARD to Create

Building them so that an SPN is a PRF is a major challenge.

Titles of Papers that tried:

The Design of S-Boxes by Simulated Annealing

A New Chaotic Substitution Box Design for Block ciphers

Perfect Nonlinear S-Boxes

If you type in S-Boxes into Google Scholar how many papers to you find?

20,000. Given repeats and conference-Journal repeats, there are approx 10,000 papers on S-boxes.

Substitution-Permutation Networks (SPNs)

1) There are attacks on 1-round and 2-round SPN's

2) Can extend attacks to r rounds but time complexity goes up.

3) These attacks are better than naive but still too slow.

4) SPN considered secure if r is large enough.

5) AES, a widely used SPN, uses 8-bit S-boxes and at least 9 rounds (and other things) and is thought to be secure.

Substitution-Permutation Networks (SPNs)

1) There are attacks on 1-round and 2-round SPN's

2) Can extend attacks to r rounds but time complexity goes up.

3) These attacks are better than naive but still too slow.

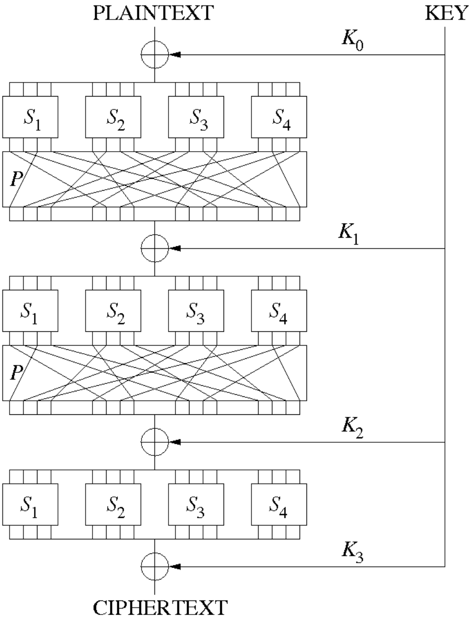
4) SPN considered secure if r is large enough.

5) AES, a widely used SPN, uses 8-bit S-boxes and at least 9 rounds (and other things) and is thought to be secure. For now.

7) Takeway: AES is a real world SPN that is really used and is believed to be a PRF.

Feistel networks

In SPN Network S-boxes Invertible



SPN: PROS and CONS

PRO: With enough rounds secure.

CON: Hard to come up with invertible S-boxes.

Feistel Networks will not need invertible components but will be secure.

Feistel networks

1) Message length is ℓ. Just like SPN.

2) Key $k = k_1 \cdots k_r$ of length n. r rounds. Just like SPN.

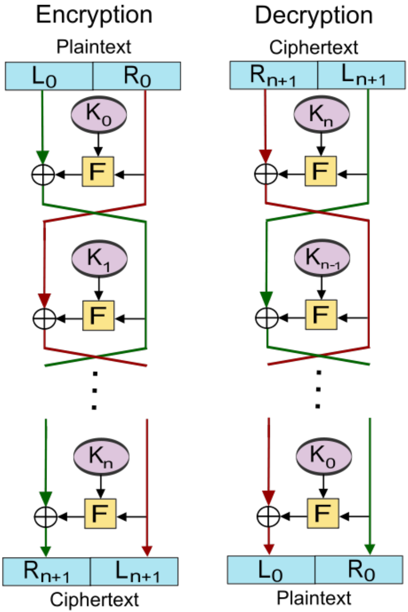
3) $|k_i| = n/r$. Need NOT be ℓ. Unlike SPN.

4) Use key k_i in ith round. Just like SPN.

5) Instead of S-boxes we have public functions $\hat{f}_i$. Need not be invertible! Unlike SPN. We derive $f_i(R) = \hat{f}_i(k_i, R)$ from them.

For 1-round:
Input: L_0R_0, $|L_0| = |R_0| = \ell/2$.
Output: L_1R_1 where $L_1 = R_0$, $R_1 = L_0 \oplus f_1(R_0)$
Invertible! The nature of $f_1(R)$ does not matter.
1) Input(L_1R_1)
2) $R_0 = L_1$.
3) Can compute $f_1(R_0)$ and hence $L_0 = R_1 \oplus f_1(R_0)$.

Feistel Network



r-round Feistel networks

1) Message length is ℓ. Just like SPN.

2) Key $k = k_1 \cdots k_r$ of length n. r rounds. Just like SPN.

3) $|k_i| = n/r$. Need NOT be ℓ. Unlike SPN.

4) Use key k_i in ith round. Just like SPN.

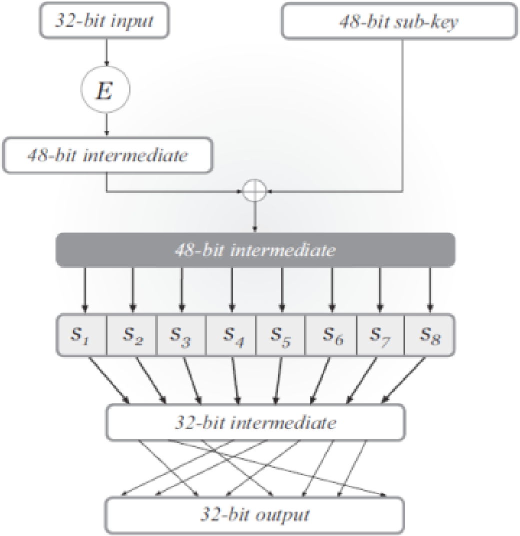
5) Public functions $\hat{f}_i$. Need not be invertible! Unlike SPN.
$f_i(R) = \hat{f}_i(k_i, R)$ from

Input: L_0R_0, $|L_0| = |R_0| = \ell/2$.
Output or Round 1: L_1R_1 where $L_1 = R_0$, $R_1 = L_0 \oplus f_1(R_0)$
Output or Round 2: L_2R_2 where $L_2 = R_1$, $R_2 = L_1 \oplus f_2(R_1)$
$\vdots \quad \vdots \quad \vdots$
Output or Round r: L_rR_r where $L_r = R_{r-1}$, $R_r = L_{r-1} \oplus f_r(R_{r-1})$

Data Encryption Standard (DES)

- Standardized in 1977
- 56-bit keys, 64-bit block length
- 16-round Feistel network
  - Same round function in all rounds (but different sub-keys)
  - Basically an SPN design! But easier to build.

DES mangler function is $\hat{f}_i$



Security of DES

PRO: DES is extremely well-designed

Security of DES

PRO: DES is extremely well-designed

PRO: Known attacks brute force or need lots of Plaintext.

Security of DES

PRO: DES is extremely well-designed

PRO: Known attacks brute force or need lots of Plaintext.

BIG CON: Parameters are too small! Brute-force search is feasible

56-bit key length

- A concern as soon as DES was released.
- Released in 1975, but that was then, this is now.
- Brute-force search over 2^{56} keys is possible
  - 1997: 1000s of computers, 96 days
  - 1998: distributed.net, 41 days
  - 1999: Deep Crack ($250,000), 56 hours
  - 2018: 48 FPGAs, 1 day
  - 2019: Will do as Classroom demo when teach this course in Fall of 2019.

Increasing key length?

- DES has a key that is too short
- How to fix?
  - Design new cipher. HARD!
  - Tweak DES so that it takes a larger key. Since this is Hardware not Software this is HARD!
  - Build a new cipher using DES as a black box. EASY?

Double encryption

- Let $F : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^\ell$
  - (i.e. n=56, ℓ=64 for DES)
- Define $F^2 : \{0,1\}^{2n} \times \{0,1\}^\ell \to \{0,1\}^\ell$ as follows:

$$F^2_{k_1,k_2}(x) = F_{k_1}(F_{k_2}(x))$$

  (still invertible)

- If best known attack on F takes time 2^n, is it reasonable to assume that the best known attack on F^2 takes time 2^{2n}? Vote! YES, NO, UNKNOWN TO SCIENCE

Double encryption

- Let $F : \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^\ell$
  - (i.e. n=56, ℓ=64 for DES)
- Define $F^2 : \{0,1\}^{2n} \times \{0,1\}^\ell \to \{0,1\}^\ell$ as follows:

$$F^2_{k_1,k_2}(x) = F_{k_1}(F_{k_2}(x))$$

  (still invertible)

- If best known attack on F takes time 2^n, is it reasonable to assume that the best known attack on F^2 takes time 2^{2n}?
  Vote! YES, NO, UNKNOWN TO SCIENCE
  NO The Meet-in-the-Middle attack takes 2^n time. We omit details.

Triple encryption

- Define $F^3 : \{0,1\}^{3n} \times \{0,1\}^{\ell} \to \{0,1\}^{\ell}$ as follows:

$$F^3_{k_1,k_2,k_3}(x) = F_{k_1}(F_{k_2}(F_{k_3}(x)))$$

- Can do meet-in-the-middle but would be 2^{2n}.
- No better attack known.

Two-key triple encryption

- Define $F^3 : \{0,1\}^{2n} \times \{0,1\}^{\ell} \to \{0,1\}^{\ell}$ as follows:

$$F^3_{k_1,k_2}(x) = F_{k_1}(F_{k_2}(F_{k_1}(x)))$$

- Best attacks take time 2^{2n} — optimal given the key length!
- Sames on key length.
- Good for some backward-compatibility issues