CSC3631 Cryptography

Cryptanalysis and Attacks

Changyu Dong

Newcastle University

Cryptanalysis

- The art and science of analyzing weaknesses of cipher algorithms
- Also known as attack.
- Cryptology $=$ Cryptography $+$ Cryptanalysis

Four general types of attack

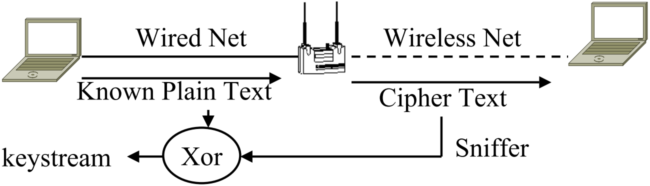
- **Ciphertext-only attack**: adversary can access only ciphertexts.
- **Known-plaintext attack**: adversary possesses some ciphertexts whose plaintexts are known.
- **Chosen-plaintext attack (CPA)**: adversary can choose plaintexts and get the corresponding ciphertexts.
- **Chosen-ciphertext attack**: in addition to chosen plaintexts, adversary can choose ciphertexts and get their decrypted plaintext.

Ciphertext-only Attack

- Example: “MFFMOW” encrypted under a shift cipher
- The weakest in terms of capabilities of the attacker.
- Usually requires a large amount ciphertext
- The easiest to defend against

Known Plaintext Attack

- Example: (shift cipher) “BXMUZFQJF” the first plaintext letter is ”P”
- Example: (Enigma) German ciphertext started with a date
- Example: (WEP)

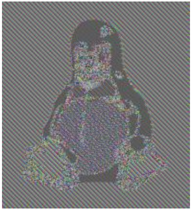


Chosen Plaintext Attack

- Example: (Vigenere cipher) can be broken with a single chosen plaintext.
- Example: (JN-25, Japanese Navy Code)

Chosen Plaintext Attack

- Deterministic ciphers are vulnerable to chosen plaintext attacks.
  - Same plaintext $\rightarrow$ same ciphertext

IND-CPA security

- A modern encryption scheme must be at least IND-CPA secure.
- Indistinguishability in the presence of chosen-plaintext attack
  - The adversary can choose any plaintext and see the ciphertext.
  - But still, for any two equal-length plaintexts, given the ciphertexts, the adversary cannot tell which is produced by which plaintext.
- Ensures no information about the plaintext (other than its size) is leaked.

Modelling CPA attack and Indistinguishability

- When analysing the security of an encryption scheme, we often use an idealised experiment to model the CPA attack and indistinguishability.
- The adversary does not know the key

1. He chooses as many plaintexts as he wants (up to the limit of his computational resources – polynomial in a security parameter), and learns the corresponding ciphertexts
2. When ready, he picks two plaintexts M_0 and M_1
    - There is no limitation excepts the length of the two plaintexts must be equal
    - He can choose plaintexts that he has previously obtained ciphertexts
3. He receives either a ciphertext of M_0, or a ciphertext of M_1
4. Repeat step 1 if necessary.

- He wins if he guesses correctly which one it is

IND-CPA experiment $SymK^{cpa}_{\mathcal{A},\Pi}(n)$

Run between a challenger an an aderversary $\mathcal{A}$. Let $\Pi = (G, E, D)$ be a symmetric-key encryption scheme, n be the security parameter:

1. The challenger generates a key k by running $G(n)$
2. The adversary $\mathcal{A}$ is given n and oracle access to $E_k(\cdot)$ – up to $poly(n)$ queries (can choose any plaintext and get the corresponding ciphertext, but not seeing the key).
3. The adversary generates two plaintexts m_0, m_1 of equal length and sends them to the challenger.
4. The challenger generates a uniformly random bit $b \in \{0, 1\}$ and then $c = E_k(m_b)$ is send back to $\mathcal{A}$.
5. $\mathcal{A}$ continues to have oracle access to $E_k(\cdot)$ – up to $poly(n)$ queries.
6. $\mathcal{A}$ output a bit b'
7. The output of the experiment is 1 if $b' = b$, and 0 otherwise. In the former case, we say that $\mathcal{A}$ wins.

AES-ECB is not IND-CPA secure

- To prove a statement is not true, it is sufficient to give a counter example
- IND-CPA requires for all Probabilistic Polynomial Time (PPT) algorithms, the winning probability is almost $\frac{1}{2}$ (with a negligible difference)
- Then to prove ECB is not IND-CPA secure, we show there exists a PPT algorithm that the winning probability is non-negligibly higher than $\frac{1}{2}$
- The adversary does the following
  - In step 1, the adversary generates m_0, and query the oracle to get $c = E_k(m_0)$.
  - In step 2, the adversary generates m_1 such that $|m_1| = |m_0|$, and sends (m_0, m_1) to the chanlenger.
  - Then, if $c = c$, output 0, otherwise output 1.

Probabilistic Encryption

- Same message, same key, running the encryption algorithm twice results in two different ciphertexts.
- The ciphers we have seen (DES, AES, etc.) are all deterministic.
- Does probabilistic encryption exist?
- We can obtain probabilistic encryption from deterministic encryption.
- The idea is in each encryption, generate some randomness, and combine this randomness with the plaintext to produce the ciphertexts.
- Example
  - CBC mode: a new random IV per encryption, $c_1 = E_k(IV \oplus p_1)$, $c_i = E_k(c_{i-1} \oplus p_i)$

Chosen Ciphertext attack

- Example: (CBC mode) Padding oracle attack

Obtaining chosen ciphertext security

- Chosen ciphertext attack is possible because the ciphertext can be modified without being detected (malleable).
- This can be countered by authenticated encryption "encrypt-then-MAC"

Encrypt-then-MAC

- An encryption scheme and a MAC scheme
- Generate two keys, one for each.
- Ciphertext is a pair (c, t) where $c = E_{k_1}(m), t = MAC_{k_2}(c)$
- To decrypt, first verify $Vrfy_{k_2}(c, t)$, then $m = D_{k_1}(c)$ if valid, return m, else return $\perp$ (error)
  - Decryption is done regardless whether verification is valid or not
  - To prevent timing attack

Why not encrypt-and-MAC

- An encryption scheme and a MAC scheme
- Generate two keys, one for each.
- Ciphertext is a pair (c, t) where $c = E_{k_1}(m), t = MAC_{k_2}(m)$
- To decrypt, first $m = D_{k_1}(c)$ then verify $Vrfy_{k_2}(m, t)$, if valid, return m, else return $\perp$
- What is wrong?

Why not encrypt-and-MAC

- An encryption scheme and a MAC scheme
- Generate two keys, one for each.
- Ciphertext is a pair (c, t) where $c = E_{k_1}(m), t = MAC_{k_2}(m)$
- To decrypt, first $m = D_{k_1}(c)$ then verify $Vrfy_{k_2}(m, t)$, if valid, return m, else return $\perp$
- What is wrong?
- MAC only cares about integrity

Why not MAC-then-encrypt

- An encryption scheme and a MAC scheme
- Generate two keys, one for each.
- Ciphertext is a pair c where $c = E_{k_1}(m||t)$ and $t = MAC_{k_2}(m)$
- To decrypt, first $m||t = D_{k_1}(c)$ then verify $Vrfy_{k_2}(m, t)$, if valid, return m, else return $\perp$
- What is wrong?

Why not MAC-then-encrypt

- An encryption scheme and a MAC scheme
- Generate two keys, one for each.
- Ciphertext is a pair c where $c = E_{k_1}(m||t)$ and $t = MAC_{k_2}(m)$
- To decrypt, first $m||t = D_{k_1}(c)$ then verify $Vrfy_{k_2}(m, t)$, if valid, return m, else return $\perp$
- What is wrong?
- This may not be CCA secure
  - e.g. Padding oracle attack is still possible

Reading

- Introduction to Modern Cryptography §3, 4.5 (relevant sections, skip the math proofs)