Cryptography

This is my note of *Cryptography 1* by ***Dan Boneh***

Overview

Crypto core

- Secret key establishment
- Secure communication
    - confidentially
    - integrity
- Digital signature
- Anonymous communication
- Anonymous **digital** cash
    - Can I spend a **digital coin** without anyone knowing who I am?
    - How to prevent double spending?
- Secure multi-party computation
    - ***Thm:*** anything that can done with trusted authority can also be done without trusted authority

- Privately outsourcing computation
  - Get search result from Google while Google doesn't know what you search for
- Zero knowledge
  - one party (the prover) can prove to another party (the verifier) that they know a value x, **without conveying any information** apart from the fact that they know the value x

Three steps in cryptography

1. Precisely specify threat model
   - What an attacker can do
   - What an attacker's goal is
2. Propose a construction
3. Prove that breaking construction under threat mode will solve an underlying hard problem

History

Substitution Cipher

- Use frequency of English letters and pairs of letters to easily break it

Vigener Cipher

- The key is a word
- Just break it as substitution cipher

Roter Machines

- Early example: the Hebern machine (single rotor)
- Most famous: the Enigma (3-5 rotors)
  - Designed to defend frequency attack (statistic attack)
  - keys = $26^4 = 2^{18}$
  - Still can't defend the ciphertext only attack

Data Encryption Standard

- DES: #keys = 2^{56} , block size = 64 bits
- Today: AES(2001), Salsa20(2008) (and many others)

Randomized algorithm

- Deterministic algorithm: $y \leftarrow A(m)$
  - output is a deterministic value

- Randomized algorithm: $y \leftarrow A(m;r)$ where $r \xleftarrow{R} \{0,1\}^n$
  - output is a random variable $y \xleftarrow{R} A(m)$

An important property of XOR

Thm: Y a rand. var. on $\{0,1\}^n$, X an indep. **uniform** var. on $\{0,1\}^n$, Then $Z = Y \bigoplus X$ is a a **uniform** var. on $\{0,1\}^n$ s

Proof: Just consider n = 1

Y	Pr.
0	p_0
1	p_1

X	Pr.
0	$\frac{1}{2}$
1	$\frac{1}{2}$

X	Y	Pr.
0	0	$\frac{p_0}{2}$
0	1	$\frac{p_1}{2}$
1	0	$\frac{p_0}{2}$
1	1	$\frac{p_1}{2}$

$$\begin{aligned} P\{Z=0\} &= P\{\ (x,y)=(0,0) \cup (x,y)=(1,1)\ \} \\ &= p_0/2 + p_1/2 \\ &= 1/2 \end{aligned}$$

The birthday paradox

Let $r_1, \cdots, r_n \in U$ be indep. identically distributed random vars.

Thm: when $n = 1.2 \times |U|^{1/2}$ then

$$P\{\exists\, i \neq j : r_i = r_j\} \geq 1/2$$

Stream Cipher

The One Time Pad

- First example of a **secure** cipher
- key = (random bit string as long the message)

$$E(k, m) = k \bigoplus m$$
$$D(k, c) = k \bigoplus c$$

- Very fast enc/dec
- but long keys as long as PT

Information Theoretic Security

- Shannon 1949
- ***Def:*** A cipher (E, D) over (K, M, C) has ***perfect secrecy*** if

$$\forall m_0, m_1 \in M \ (|m_0| = |m_1|), \forall c \in C$$
$$P\{E(k, m_0) = c\} = P\{E(k, m_1) = c\}$$

- **No CT only attack**
- ***Lemma:*** OTP has ***perfect secrecy***
  - ***Proof:*** $\forall m \in M, c \in C$, There is exactly one key $(m \bigoplus c)$ maps m to c
- ***Thm:*** perfect secrecy $\Rightarrow |K| \geq |M|$
- Hard to use in practice!

Pseudorandom Generators

- ***Stream Ciphers:*** making OTP practical
  - Idea: replace **random** key by "pseudorandom" key
  - Goal: decrease the length of key
- Stream ciphers **cannot** have perfect secrecy
  - Need a different definition of security
  - Security will depend on specific PRG
- **WEAK** PRG: glibc random():

$$r[i] \leftarrow (r[i-3] + r[i-31]) \% 2^{32}$$

$$return\ r[i] >> 1$$

- PRG **MUST** be unpredictable
- We say that $G : K \rightarrow \{0,1\}^n$ is **predictable** if:

$$\exists\ alg.\ A, \exists\ 0 \leq i \leq n-1$$

$$P\{A(G(k))|_{1,\cdots,i} = G(k)|_{i+1}\} > 1/2 + \varepsilon$$

- ***Def:*** PRG is **unpredictable** if it is not predictable
  $\forall i$, no **efficient** alg. can predict bit_{i+1} for **non-negligible** ε
- Negligible
  - In practice: ε is a scalar
    - non-neg: $\varepsilon \geq 1/2^{30}$ (likely to happen over 1 GB of data)
    - negligible: $\varepsilon \leq 1/2^{80}$ (won't happen over life of key)
  - In theory: ε is a function $\mathbb{Z}^{\geq 0} \rightarrow \mathbb{R}^{\geq 0}$
    - non-neg: $\exists d : \varepsilon(\lambda) >= 1/\lambda^d$ inf. often
    - negligible: $\forall d, \lambda \geq \lambda_d : \varepsilon(\lambda) < 1/\lambda^d$

Attacks on OTP and stream ciphers

Attack 1: two time pad is insecure

Never use stream cipher key more than once

$$c_1 \leftarrow m_1 \bigoplus PRG(k)$$

$$c_2 \leftarrow m_2 \bigoplus PRG(k)$$

Then Adv. does:

$$c_1 \bigoplus c_2 \rightarrow m_1 \bigoplus m_2$$

Enough redundancy in English and ASCII encoding that:

$$m_1 \bigoplus m_2 \rightarrow m_1, m_2$$

- 802.11b WEP

$$c = m \bigoplus PRG(IV||k)$$

1. IV increases by one every frame, but length of IV is only 24 bits. After $2^{24} \approx$ 16M frames it repeats.

2. keys are related(only 24 of 1048 bits are different), And PRG used in WEP(RC4) is not secure when you use related keys.
   A better construction
   Also use PRG(k) to generate new keys so that each frame has a pseudorandom key.

- Disk encryption
  When the file changes, it's easy to tell where the change occurred instantly. That leaks information that attackers shouldn't actually know. Essentially it's another example of two time pad.
  Typically do not use a stream cipher in disk encryption

Attack 2: no integrity

OTP is malleable. Modifications to CT are undetected and have **predictable** impact on PT.

$$D(E(m,k) \bigoplus p, k) = m \bigoplus p$$

Attackers can choose p to modify PT.

Real-world stream ciphers

RC4

Software cipher

$$k(128\ bits) \rightarrow k'(2048\ bits) \nrightarrow 1\ byte\ per\ round$$

- used in HTTPS and WEP
- Weaknesses:
  - Bias in initial output: $P\{2^{nd} byte = 0\} = 2/256 > 1/256$
  - $P\{(0,0)\} = 1/256^2 + 1/256^3 > 1/256^2$
  - Related key attacks

CSS

Hardware cipher **(badly broken)**

Using Linear-feedback shift register (LFSR)

eStream

a new kind of PRG: $\{0,1\}^s \times Nonce \rightarrow \{0,1\}^n$

$Nonce$: a non-repeating value for a given key.

$$E(k, m; r) = m \bigoplus PRG(k; r)$$

The pair (k,r) is never used more than once.

$Nonce$ is designed to reuse the key more than once.

A famous and successful example: Salsa 20

PRG Security Defs

Let $G{:}K \rightarrow \{0,1\}^n$ be a PRG

Goal: define what it means that

$$k \xleftarrow{R} K, output\ G(k)$$

is **indistinguishable** from

$$r \xleftarrow{R} R, output\ r$$

Statistical Tests

Def: an alg. A s.t. $A(x)$ outputs "0" or "1"

Advantage

Let $G{:}K \rightarrow \{0,1\}^n$ be a PRG, $\{0,1\}^n \rightarrow r$ and A a **statistical test** on $\{0,1\}^n$

Def:

$$Adv_{PRG}[A, G] = |Pr[A(G(k)) = 1] - Pr[A(r)] = 1| \in [0,1]$$

Adv close to 1 $\Rightarrow$ A can dist. G from random

Adv close to 0 $\Rightarrow$ A cannot dist. G from random

Def: We say that $G : K \rightarrow \{0,1\}^n$ is a **secure** PRG if $\forall$ "eff" stat. tests A: $Adv_{PRG}[A, G]$ is **negligible**.

Thm: a secure PRG is unpredictable.
Proof: Just show a predictable is insecure.

Thm: an unpredictable PRG is secure.
Proof: https://en.wikipedia.org/wiki/Yao's_test

More generally, let P_1 and P_2 be two distributions over $\{0,1\}^n$
Def: We say that P_1 and P_2 are **computationally indistinguishable** if no **eff.** stat. tests that can distinguish P_1 and P_2 (denoted $P_1 \approx_p P_2$)

Semantic Security

Adv. A gives Chal. two message m_0 and m_1, Chal. returns c=$E(k, m_0)$ or c=$E(k, m_1)$

Def: E is **semantically secure** if for all efficient A:

$$Adv_{SS}[A, E] = |Pr[c = E(k, m_0)] - Pr[c = E(k, m_1)]| < negligible$$

$\Rightarrow$ for all explicit $m_0, m_1 \in M, k \leftarrow K : \{E(k, m_0)\} \approx_p \{E(k, m_1)\}$

Thm: $G : K \rightarrow \{0,1\}^n$ is a secure $PRG \Rightarrow$ stream cipher E derived from G is semantically secure

Proof: Just to prove: $\forall$ sem. sec. adversary A, $\exists$ a PRG adversary $B s.t.

$$Adv_{SS}[A, E] \leq 2Adv_{PRG}[B, G]$$

Block Cipher

PRFs and PRPs

Def: Pseudo Random Function **(PRF)** defined over (K, X, Y):

$$F : K \times X \rightarrow Y$$

such that exists **efficient** algorithm to evaluate $F(k, x)$

Def: Pseudo Random Permutation **(PRP)** defined over (K, X)

$$E : K \times X \rightarrow X$$

such that:

1. Exist **efficient** algorithm to evaluate $E(k, x)$
2. The function $E(k, \cdot)$ is one-to-one
3. Exists **efficient** inversion algorithm $D(k, y)$

Block cipher is actually a PRP

AES: $K \times X \to X$ where $K = X = \{0,1\}^{128}$

3DES: $K \times X \to$ where $X = \{0,1\}^{64}, K = \{0,1\}^{168}$

Secure PRFs and PRPs

Let $F : K \times X \to Y$ be a PRF

- S_U: the set of all functions from $|X|$ to $|Y|$, $|S_U| = |Y|^{|X|}$
- $S_F = \{F(k,\cdot)\ , k \in K\} \subseteq S_U, |S_F| = |K|$

Def: a PRF is **secure** if a random function in S_U is **indistinguishable** from a random function in S_F

If we replace S_U with the set of all **one to one** functions from X to X, then we get **def.** of secure PRP, and actually we also get secure block cipher.

An easy application: PRF $\Rightarrow$ PRG

Let $F : K \times \{0,1\}^n \to \{0,1\}^n$ be a secure PRF.

Then following $G : K \to \{0,1\}^{nt}$ is a secure PRG:

$$G(k) = F(k,0)||F(k,1)||\cdots||F(k,n-1)$$

It's security is easy to prove from the security of PRF.

Data Encryption Standard

Core idea : Feistel Network

Given functions $f_1, \cdots, f_d : \{0,1\}^n \to \{0,1\}^n$

Goal: build **invertible** function $F : \{0,1\}^{2n} \to \{0,1\}^{2n}$

Feistel Network uses f as round functions. For each round $i = 0, 1, \cdots d$

1. Split the text block into two equal pieces L_i, R_i
2. compute

$$\begin{cases} R_i = f_i(R_{i-1}) \bigoplus L_{i-1} \\ L_i = R_{i-1} \end{cases}$$

Then $L_d||R_d$ is the ciphertext.

Decryption is just to replace the second step with

$$\begin{cases} L_{i-1} = f_i(L_i) \bigoplus R_i \\ R_{i-1} = L_i \end{cases}$$

Then $L_0||R_0$ is the plaintext.

Thm(Luby-Rackoff '85): If $f : K \times \{0,1\}^n \to \{0,1\}^n$ is a secure PRF, Then 3-round Feistel $F : K^3 \times \{0,1\}^{2n}$ is a secure PRP.

Now we have a method to build a secure PRP from a secure PRF. Essentially we have an efficient method to construct invertible functions from normal functions.

The Feistel structure has the advantage that encryption and decryption operations are very similar, even identical in some cases, requiring only a reversal of the key schedule. Therefore, the size of the code or circuitry required to implement such a cipher is nearly halved.

Details about DES

DES is a 16-round Feistel Network. The key has 64 bits but only 56 bits are effective.

Since DES is based on Feistel Network, we can focus on the function f used in F-Network.

f_i is a function $F(k_i, x)$ and k_i is from the DES key k.

The F-function, operates on half a block (32 bits) at a time and consists of four stages:

1. **Expansion:** the 32-bit half-block is expanded to 48 bits using the expansion permutation.
2. **Key mixing:** the 48-bit result is combined with k_i using an XOR operation.
3. **Substitution:** after mixing in the k_i, the block is divided into eight 6-bit pieces before processing by the S-boxes, or substitution boxes. Each of the eight S-boxes replaces its 6 input bits with 4 output bits according to a non-linear transformation, provided in the form of a lookup table.
4. **Permutation:** finally, the 32 outputs from the S-boxes are rearranged according to a fixed permutation, the P-box. This is designed so that, after permutation, the bits from the output of each S-box in this round are spread across four different S-boxes in the next round.

More details see: https://en.wikipedia.org/wiki/Data_Encryption_Standard

The S-boxes

$$S_i : \{0,1\}^6 \to \{0,1\}^4$$

If S-boxes are linear or almost linear, formally

$$S_i(\vec{x}) \equiv A_i \cdot \vec{x} \ (mod\ 2)$$

We say that S_i is a linear function. Then the entire DES would be linear so that the entire encryption function is actually matrix multiplication in modulo 2. DES would be so easy to break.

DONOT choose S-boxes at random.

Strengthening DES against exhaustive search

Because of the 56-bit key, the naive DES can now be cracked by anyone using a exhaustive search. As a result, many methods have emerged to strengthen DES.

Method 1: Triple-DES

Let $E : K \times M \rightarrow M$ be a block cipher

Def: $3E((k_1, k_2, k_3), m) = E(k_1, D(k_2, E(k_3, m)))$

For 3DES: $|K| = 2^{56*3} = 2^{168}$

but easy attack in time 2^{112} (still secure)

- Why not $E(k_1, E(k_2, E(k_3, m)))$?
  If $k_1 = k_2 = k_3$, then $E(k_1, D(k_2, E(k_3, m))) = E(k_1, m)$. We can implement naive DES with circuits that implement 3DES.
- Why not Doubyle-DES?
  With 112-bits key, Double-DES has easy attack in 2^{57} time and is not enough to resist exhaustive search. This attack will be covered in detail soon.

Method 2: DESX

Let $E : K \times M \rightarrow M$ be a block cipher

Def: $EX((k_1, k_2, k_3), m) = k_1 \bigoplus E(k_2, m \bigoplus k_3)$

for DESX: $|K| = 2^{64+56+64} = 2^{184}$

but easy attack in time $2^{64+56} = 2^{120}$(still secure)

Note: $k_1 \bigoplus E(k_2, m)$ or $E(k_2, m \bigoplus k_1)$ does NOTHING !!

Meet-in-the-middle attack

Def: $2E((k_1, k_2), m) = E(k_1, E(k_2, m))$

Goal: Find (k_1, k_2) s.t. $E(k_1, E(k_2, m) = c)$

Equivalently: $E(k_2, m) = D(k1, c)$

Attack:

1. build a table containing all $E(k_2, m)$ for all possible k_2, the size of table is 2^{56}.
2. For each possible c decryption result, loot it up in the table for the same. It can be implemented by binary search or hash table.

The complexity of Meet-in-the-middle attack depends on the complexity of the lookup algorithm. If a hash lookup is used, it can theoretically be completed in $2^{56} + 2^{56} = 2^{27}$ time.

In the same way, we can crack 3DES in 2^{112} time. That's why 3DES and 2DES are less secure than they seem. 3DES is still secure, but never use 2DES.

Advanced Encryption Standard

Advanced Encryption Standard (AES) is the first (and only) publicly accessible cipher approved by the National Security Agency (NSA) for top secret information when used in an NSA approved cryptographic module.

AES is based on a design principle known as a substitution–permutation network, and is efficient in both software and hardware. Unlike its predecessor DES, AES does not use a Feistel network.

Key sizes: 128, 192, 256 bits.
Block size: 128 bits

High-level description of the algorithm

1. **KeyExpansion** — round keys are derived from the cipher key using Rijndael's key schedule. AES requires a separate 128-bit round key block for each round plus one more.
2. **Initial round key addition:**
   AddRoundKey — each byte of the state is combined with a byte of the round key using bitwise xor.
3. **9, 11 or 13 rounds:**
   - **SubBytes** — a non-linear substitution step where each byte is replaced with another according to a lookup table.
   - **ShiftRows** — a transposition step where the last three rows of the state are shifted cyclically a certain number of steps.
   - **MixColumns** — a linear mixing operation which operates on the columns of the state, combining the four bytes in each column.

    - **AddRoundKey**
4. **Final round** (making 10, 12 or 14 rounds in total):
    - **SubBytes**
    - **ShiftRows**
    - **AddRoundKey**

More details see: https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

More attacks on block ciphers

Side-channel attack

- Measure time or power to do enc/dec.
- Computing errors in the last round expose the secret key k

$\Rightarrow$ DONOT even try to implement crypto primitives ourselves

Linear cryptanalysis

Linear cryptanalysis is a general form of cryptanalysis based on finding affine approximations to the action of a cipher.

For DES, attack time $\approx 2^{43}$ with 2^{42} random in/out pairs.

Differential cryptanalysis

Differential cryptanalysis is a general form of cryptanalysis applicable primarily to block ciphers, but also to stream ciphers and cryptographic hash functions. In the broadest sense, it is the study of how differences in information input can affect the resultant difference at the output.

Quantum attack

Generic search problem

Let $f : X \to \{0, 1\}$ be a function

Goal: find $x \in X$ s.t. $f(x) = 1$

Classical computer: best generic algorithm time = $O(|X|)$

Quantum computer[Grover'96]: time = $O(|X|^{1/2})$

See: https://en.wikipedia.org/wiki/Grover's_algorithm

$$f(x) = \begin{cases} 1, \ E(k,m) = c \\ 0, \ otherwise \end{cases}$$

Cracking DES has now become a generic search problem that can be solved in square root time.

GGM PRF: PRG $\Rightarrow$ PRF

$G : K \rightarrow K^2$

$F(k, x \in \{0,1\}) = G(k)[x]$

Thm: G is a secure PRG $\Rightarrow$ F is a secure 1-bit PRF

In a similar way, we can construct a secure PRF $F : K \times \{0,1\}^n \rightarrow K$ from a secure PRG $G : K \rightarrow K^2$. Then use Luby-Rackoff theorem, we can get a secure PRP (block cipher).

This method is not practical because it is too slow since it has to run PRG n times to generate n bits. That's why we still need DES or AES which is a heuristic PRF.

PRF Switching Lemma

Any secure PRP is also a secure PRF, if $|X|$ is sufficiently large.

Lemma: Let E be a PRP over (K, X), Then for any q-query adversary A:

$$|Adv_{PRF}[A, E] - Adv_{PRP}[A, E]| < q^2/2|X|$$

Using block cipher

CPA security

Adversary's power: chosen-plaintext attack (CPA)

- Can obtain the encryption of arbitrary messages of his choice(conservative modeling of real life)

Adversary's goal: Break sematic security

Thm: Any encryption that always outputs same ciphertext for message m is not semantically secure under CPA.

Solution 1: randomized encryption

$$E(k,m) = [r \xleftarrow{R} R, output \ (r, F(k,r) \bigoplus m)]$$

E is semantically secure under CPA if $|R|$ is large enough so r never repeat.

Solution 2: nonce-based encryption

$$E(k,m) = [r = r + 1, output\ (r, F(k,r) \bigoplus m)]$$

E is semantically secure under CPA if $|R|$ is large enough so r never repeat.

Block cipher mode of operation

In cryptography, a block cipher mode of operation is an algorithm that uses a block cipher to provide information security such as confidentiality or authenticity. A block cipher by itself is only suitable for the secure cryptographic transformation (encryption or decryption) of one fixed-length group of bits called a block. A mode of operation describes how to repeatedly apply a cipher's single-block operation to securely transform amounts of data larger than a block.

See: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

Message Integrity

Goal: **integrity**, no confidentially.

Message Authentication Codes

In cryptography, a message authentication code (MAC), sometimes known as a tag, is a short piece of information used to authenticate a message—in other words, to confirm that the message came from the stated sender (its authenticity) and has not been changed.

Def: T: tag space, K: key space, M: message space. Message Authentication Codes (MAC) $I = (S, V)$ defined over (K, M, T) is a pair of algorithms:

- $S(k, m)$ outputs $t \in T$
- $V(k, m, t)$ outputs yes or no

Why integrity requires a secret key?

CRC is an error-detecting code with no secret key commonly used in digital networks and storage devices to detect **accidental** changes to raw data. But in cryptography, attacker can modify message at will and easily re-compute CRC.

The truth is very simple. Without a secret key, the sender and the attacker are actually equivalent to the receiver. It is impossible to tell whether the message has been tampered with by the attacker.

Secure MACS

Attacker's power: chosen message attack

- for $m_1, \cdots, m_q$ attacker is given $t_i \leftarrow S(k, m_i)$

Attacker's goal: existential forgery

- produce **any** new valid pair (m, t) s.t. $V(m, t, k) = yes$
- note: given (m,t) attacker cannot even produce (m, t') for $t' \neq t$, although t' may exist.

Def: $I = (S, V)$ is a secure MAC if for all efficient Adv. A $Adv_{MAC}[A, I] = P[V(\cdot, m, t) = yes]$ is **negligible**.

Thm: If $F : K \times X \rightarrow Y$ is a secure PRF and $1/|Y|$ is negligible, say $|Y| = 2^{80}$ then I_F is a secure MAC.

AES is a secure MAC for 16-byte messages.

Main problem: how to convert Small-MAC into Big-MAC?

Two main constructions used in practice:

- CBC-MAC
- HMAC

Both convert a small-PRF into a big-PRF.