lec 5

- Intro to cryptography
  - secure communication over insecure communication channels
  - ↑: a way to provide formal guarantees in the presence of an attacker.

- Main goal of cryptography.
  1° Confidentiality: preventing adversaries from reading our private data.
  2° Integrity: preventing attackers from altering some data.
  3° Authenticity: ensuring that the expected user created data.

A can verify a msg oriented from B.

- Symmetric-key cryptography
- Public-key (asymmetric-key) cryptography

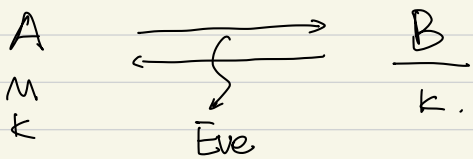
- Kerkhoff principle.

cryptography should remain secure even when the attacker knows all internal details of the algorithm / system.

key is the only thing must be kept secret. ⇒ when leak, easy to change.

- Concept
  - Encryption and decryption using a shared key
- Goals
  - Support arbitrarily long messages
  - Confidentiality - even when encrypting multiple, possibly related, things
- Ideas
  - Use block ciphers with chaining modes
  - Indistinguishability Under Chosen-Plaintext Attack (IND-CPA) security game

Symmetric key



CT: ciphertext

$C \leftarrow Enc(k, M)$ → C → $m \leftarrow Dec(k, C)$

needs to provide confidentiality:

i.e. C to hide all info about M besides the length

why? Assume static CT size n

i) Can't encrypt messages larger than n. otherwise lose info.

ii) Encrypting small message is wasteful

✡ Symmetric Encryption Scheme (API)

keygen() → k : needs to be secure

Enc(k, M) → C

Dec(k, C) → M

Correctness : $\forall k, \forall M, C \leftarrow Enc(k, M)$: $Dec(k, C) \rightarrow M$

Security:?

— Adv. knows keygen, Enc, Dec, but doesn't know k

Naive idea:

— Given C, an Adv. can't recover M

⇒ not good enough. not deal with partial info leakage.

Ex 1: db which holds deterministic encryptions of students' grades ⇒ same grade → same encryption.

⇒ Adv. can learn which students have the same grades

⇒ Given value of one ciphertext, the Adv. can decrypt many

Ex 2: db holds records which indicate whether True or False. Enc leaks first letter of the message.

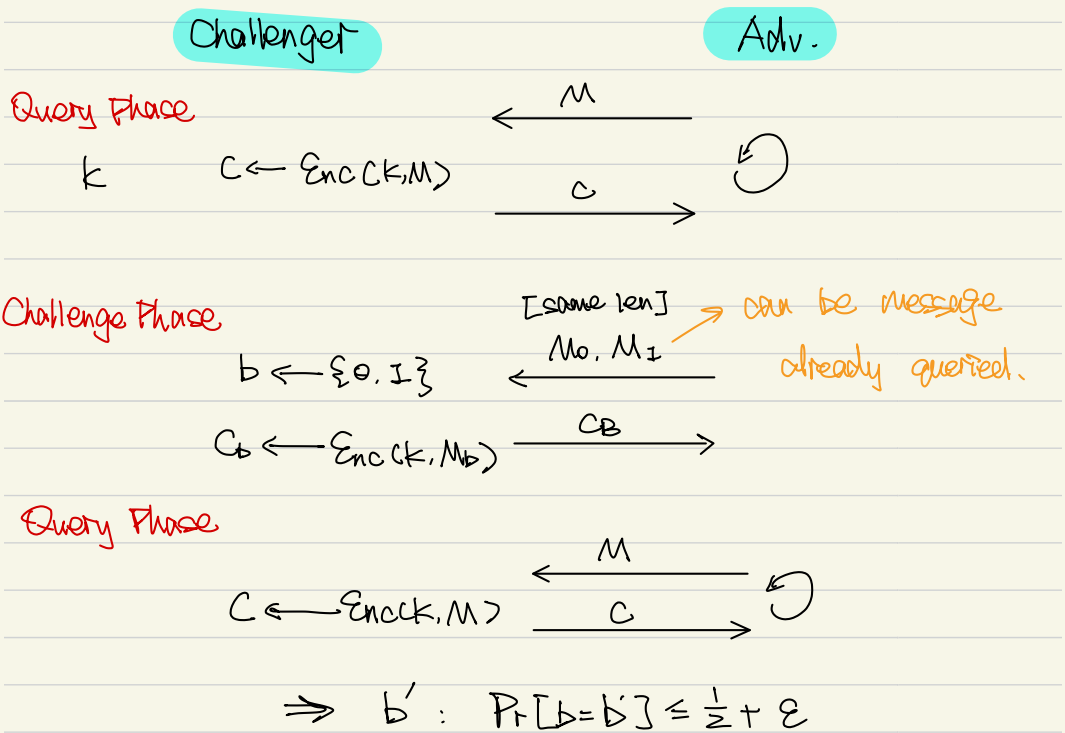
general security goal:

No partial info about M may leak b/c an Adv. can couple it with side info. to reconstruct M.

→ No Attacker should be able to distinguish two messages based on their encryption.

→ formalize the security property.

IND-CPA : indistinguishable - chosen plain text attack



$\Rightarrow b' : Pr[b = b'] \leq \frac{1}{2} + \varepsilon$

① Ex1 : deterministic encryption "X pass."

② Ex2 : close M0, M1 with different first letter "X pass"

Ex: $Enc(k, M) = 2 \cdot M$ ×

$Enc(k, M)$ = random number ✓ × correctness

$Enc(k, M) = (k + M) \bmod p$ × $M = 0 \Rightarrow$ 得 $k \bmod p$

$Enc(k, M) = 3$ ✓ IND-CPA × correctness

IND-CPA ensures a correct scheme is:

1) Non-deterministic

→ If not: we can query the same messages used in the challenge.

2) Confidence:

→ If not: we can make queries to leak which challenge messages was chosen.

For all adversaries!

- For an IND-CPA correct scheme we need:
  1. one time pad.
  2. Block cipher

A

n: size

keygen():

$K = k_1 \cdots k_n$

$M = M_1 \cdots M_n$

$Enc(K, M) = K \oplus M$

B

$k = k_1 \cdots k_n$

$Dec(k, M)$

$= k \oplus k \oplus M$

$= M$

✗ Not IND-CPA. Choose $M = 0$, get k.!

Property:

use the key only once, it is secure.

Given a ciphertext C. ($k \xleftarrow{\$} : C = k \oplus M$)

$$Pr[Adv(C) = M] \le \varepsilon$$

$$Pr[Adv(C, M_0, M_1) = M_b] = \frac{1}{2}$$

$C = M_0 \oplus (M_0 \oplus C)$ (k_0)

$C = M_1 \oplus (M_1 \oplus C)$ (k_1)

$k \xleftarrow{\$}$, each is equally likely

Question 2 ***PRNGs and stream ciphers*** ()

(a) Suppose you have access to function R that takes a 128-bit seed s and integers n, m as input. R outputs the n^{th} (inclusive) through m^{th} (exclusive) bits produced by the a pseudorandom generator $PRNG$ when it is seeded with seed s.

$$R(s, n, m) = PRNG(s)[n : m]$$

Use R to make a secure symmetric-key encryption scheme. That is, define the key generation algorithm, the encryption algorithm, and the decryption algorithm.

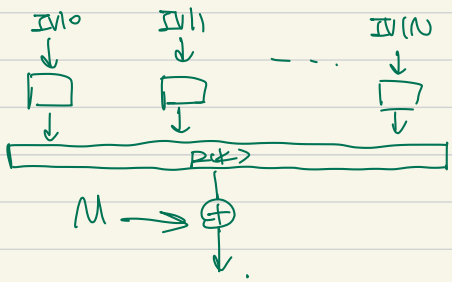
(b) Explain how using a block cipher in counter (CTR) mode is similar to the scenario described above.

(a) 1. $keygen() = k \xleftarrow{R} \{0,1\}^{128}$

2. $Enc(k, M) = R(k, \cdot, \cdot + length(M)) \oplus M$

3. $Dec(k, C) = R(k, \cdot, \cdot + length(M)) \oplus C$

(b)

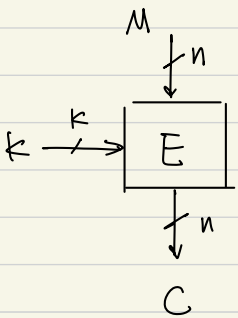


CTR ⇒ stream cipher.

Lec 6.

⇒ use random permutation : bijective, can be decryptable

- Block Cipher

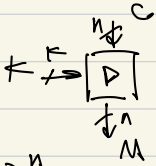


$E: \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$

$E(k, M)$

$E_k(M)$

$E_k: \{0,1\}^n \to \{0,1\}^n$



< random function
random permutation

uniform distribution ⇒ random

F:

in	out
00	01
01	11
10	00
11	10

$\{0,1\}^2 \to \{0,1\}^2$

P:

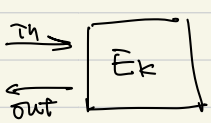
in	out
00	11
01	10
10	01
11	00

not use random perm b/c need to transfer whole table, while block cipher only needs key.

- Block Cipher Security

brute force key ☆☆

$E_k \approx$ RANDOM PERMUTATION.



secure <=>

$Pr[\text{ATTACKER WINS}] \leq \frac{1}{2} + \text{NEGL}$

for ∀ ATTACKERS

$C = E_K(M)$

1) recover M? X

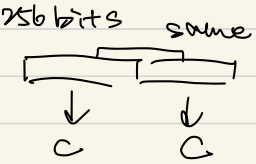
2) LSB of M? suppose can, then can win distinguish game

- attacker can learn $M_1 \neq M_2$ ⇒ deterministic

Ex: M size > 128, every 128 bits use AES, secure?

X.

i)

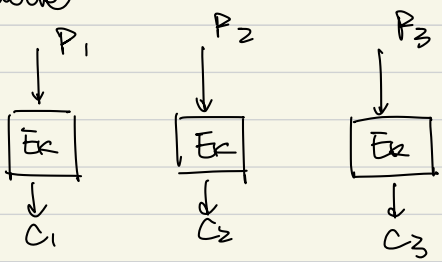


⇒ partial info.

M_0, M_1 : M_0 first and second half is same

ii) deterministic

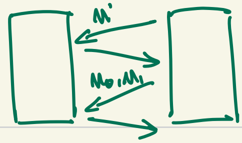
1° ECB mode



not IND-CPA!

deterministic

$Enc(K, P_1|P_2|P_3) = C_1|C_2|C_3$



$M' \to E(k, IV')$

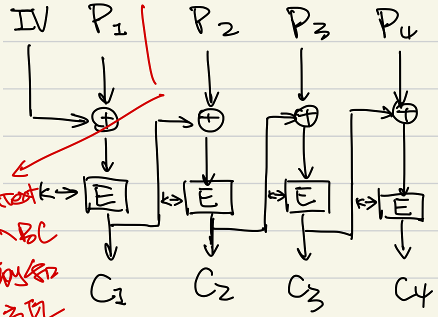
$M_0 \oplus IV_0 = M' \oplus IV'$

$M_0 = M' \oplus IV' \oplus IV_0$.

IV不可预测

2° CBC mode.

→ sender choose: every time randomly choose [initialization vector]



plaintext 放入BC 之前和 IV 交互 可预测.

decryptable ⇒ block cipher needs to be bijective i.e. invertible

- Query: 0110--110, C
- Query: 0110--111, C'
-

$Enc(k, P_1 | P_2 | \cdots) = IV | C_1 | C_2 | C_3 \cdots$

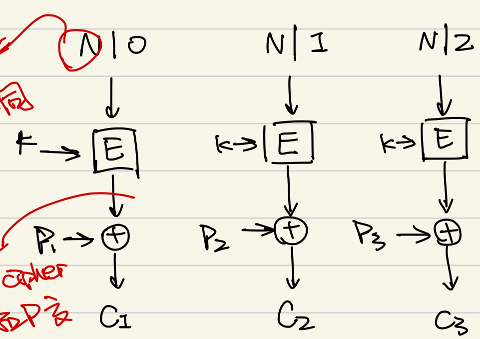
0
1
2
3

Rk: i) same IV ⇒ deterministic

ii) hardware get source of randomness ⇒ environment: key presses time, temperature...

3° CTR Mode (counter mode)

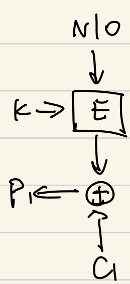
视为 block cipher → stream cipher, 最终和 msg 异或



每个 N 都不同

block cipher 运行后和 P 交互.

decrypt:



→ × necessarily invertible.

→ different ok

$Enc(k, P_1 | P_2 | \cdots)$

$= N | C_1 | C_2 \cdots$

N → public.

i) needs N to keep same message non-deterministic

ii) needs counters to keep blocks different.

* CTR can parallel.

decryptability
security

○ padding.

M|0---0

→ not decryptable : and

{ M|0---0 , if M end in 0
{ M|1---1 , if M end in 1.

M||0---0

Pk.

① only first bit unknown

1 ⟼ padding 0 ⟼ padding

⇒ IND-CPA : secure M_0, M_1

— even message is known,
can not know the key.

○ pseudo number random generation (PRG) ⇒ 生成随机数.

short key / seed

G: $\{0,1\}^{\in}$ → $\{0,1\}^{n}$ ⇒ 生成任意长度. (IV, nounce ...)

G(k) P.R.G.

stream cipher (利用PRG进行加密) ⇒ G(k, IV)

Enc(k, P) = IV | P ⊕ G(k, IV) (近似随机数)

⇒ 类似OTP: Enc(k, P) = P ⊕ k → truly random. ①key use only once.
②very long.

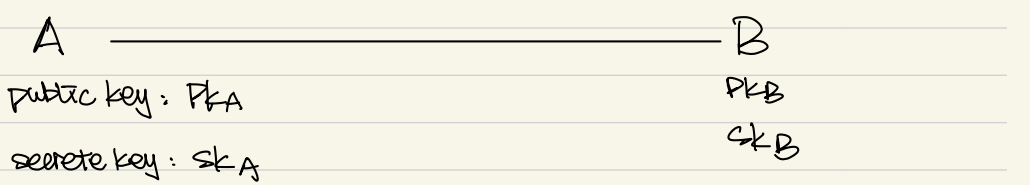
⇒ 类似CTR: G(k, IV) = Ek(IV|0) | Ek(IV|1) | ...
(used in stream cipher)

* block cipher is not a type of PRG. for PRG: output much longer than input, for block cipher: same size. can use block cipher to build PRG.

- Concept
  - Two parties should be able to communicate without prior interaction
- Goals
  - Use asymmetric keys to derive shared symmetric keys
  - Use asymmetric keys to securely communicate
  - Use asymmetric keys to authenticate ('sign') messages
- Ideas
  - Take advantage of hard problems in mathematics:
    - Discrete logarithm
    - Factoring
    - .... (and many others)

lec 7

- Asymmetric cryptography



$E_K(PK_B, M) = C$ - - - - - - $\longrightarrow$ $Dec(SK_B, C) = M$

Syntax:

$Keygen() \longrightarrow (Pk, Sk)$

$Enc(Pk, m) \longrightarrow C$

$Dec(sk, C) \longrightarrow m$

Correctness:

$\forall M. \forall Sk, Pk \leftarrow keygen()$

$C \leftarrow Enc(Pk, M);$

$Dec(Sk, C) = M$

Security:

One-way functions:

A function f is one way if: (poly-time)

(1) Given X, it is easy to compute $f(x)$

(2) Given y, it is hard to find any x s.t. $f(x) = y$ (no poly-time)

Ex:

$f(x) = x$, No, easy find x

$f(x) = 1$, No, any $x \to 1$

$f(x) = E_K(x)$, yes it is indistinguishable from random permutation.

k unknown

$f(x) = O(P(k, x))$, no, attacker can leak key in one query

- **Discrete logarithm Problem**

$f(x) = g^x \bmod p$ where p is a large prime (2048 bits long), g is a random value in $[2, p-1]$

Assumption: f_{DLP} is OWF

i) easy to compute:

say x is also 2048 bits (快速幂)

ii) hard to invert.

- **Diffie Hellman key Exchange** (1976 Turing)

large prime p

$1 < g < p-1$

A		B
$a \leftarrow \{1, \cdots p-2\}$ secret random	$\xrightarrow{PK_A}$	$b \leftarrow \{1, \cdots p-2\}$ secret random
$PK_A = g^a \bmod p$	$\xleftarrow{PK_B}$	$PK_B = g^b \bmod p$
$PK_B^a = g^{ab} \bmod p$		$PK_A^b = g^{ab} \bmod p$
$\Vert$		$\Vert$
k	$\longleftrightarrow$	k

symmetric-key encryption communication using k.

✡ Security.

E: sees: $A = g^a \bmod p$ ⇏ a

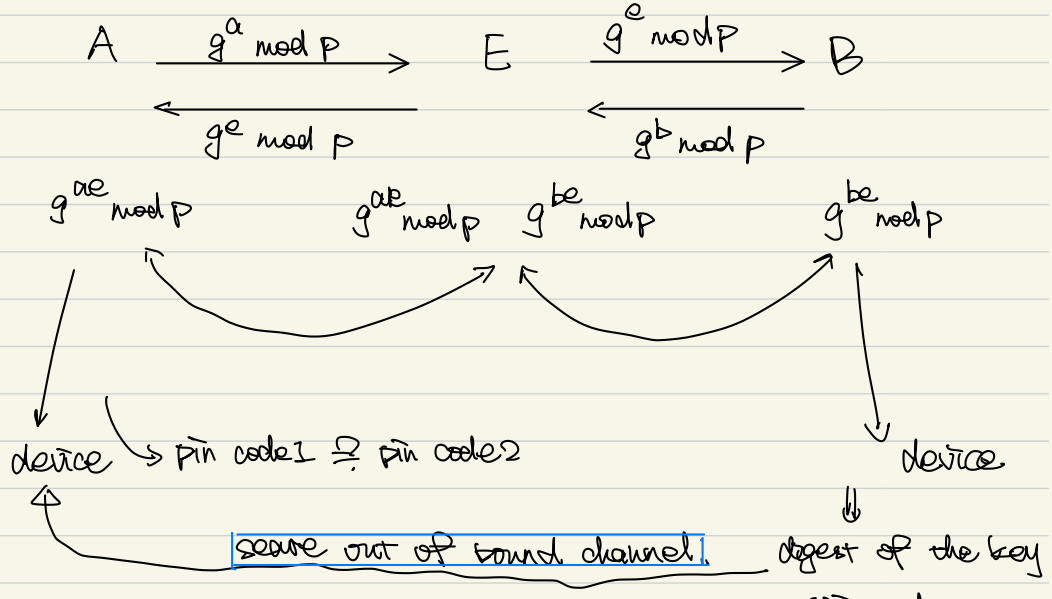
$B = g^b \bmod p$ ⇏ b

⇏ $g^{ab} \bmod p$.

Assumption: you can not break DLP (DLP is OWF) ⇐ necessary

you can not compute $g^{ab} \bmod p$ from $g^a, g^b \bmod p$.

— Man in the middle attack (MiTM)



- Assume adv not control both channels.

◦ Public-key encryption

A —— $Enc(pk_B, m)$ ——→ B

↑ E

pk_A, sk_A (A) pk_B, sk_B (B)

$Dec(sk_B, m) = M$

1. keygen (pk, sk)
2. $Enc(pk, m) \to c$
3. $Dec(sk, c) \to m$

Correctness: $\forall\ pk, sk \leftarrow \text{KeyGen},\ \forall m,\ c = Enc(pk, m)$

$Dec(sk, c) = M$

Security: similar in spirit with IND-CPA

— Semantic Security Games

Ch		Adv.
$\text{KeyGen}() \to pk, sk$ (sk circled → not about security)	pk ——→	pk (can encrypt himself using pk.)
choose a message	←—— m_1, m_0 s.t. $\lvert m_0\rvert = \lvert m_1\rvert$	
random $b \xleftarrow{\$} \{0, 1\}$	$Enc(pk, m_b)$ ——→	
	←—— b'.	

和IND-CPA的区别：
没有query阶段.
传给Adv Pk.

$\forall$ Adv,

$$\Pr[\text{Adv. wins } (b' = b)] \le 1/2 + \varepsilon$$

ElGamal cryptography (1985)

keygen() ⇒ generate # a large prime p (2048-bit)

⇒ $g \in [2 \cdots p-1]$

⇒ generate a random secret key $k \in [2 \cdots p-2]$ → Sk.

→ $Pk = g^k \bmod p$

g, p public

i) Publish Pk. keep Sk secret

ii) Due to the DLP assumption, cannot guess k.

Enc(Pk, m): $m \in [1, \cdots, p-1]$ ⇒ $|m| < |p|$ in case mod p lose info. ⇒ can not be 0. → $C_2 = 0$.

- pick # $r \in [1, \cdots, p-1]$ 预处理

① encryption: $C = (g^r \bmod p \; ; \; m \cdot Pk^r \bmod p)$

$g^r \bmod p$: keep r secret → C_1: like a one time pad.

Pk^r: secret → $C_2 = m \cdot g^{kr} \bmod p$

② decryption:

Dec(Sk, C_1, C_2))

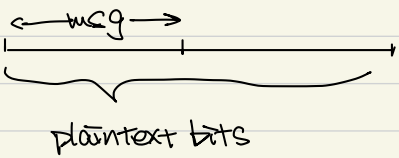
$\frac{C_2}{C_1^k} \bmod p = m$

i) correctness:

$\frac{C_2}{C_1^k} \bmod p = \frac{m \cdot (g^r \bmod p)^k}{(g^r \bmod p)^k} = m$

Lec 8

- Padding



- padding 10...0:

only works for size of msg < plaintext bits

→ using this, able to encrypt 0 with ElGamal.

Enc: add padding
Dec: remove padding

- encrypt very long message:

idea: using public-key scheme to encrypt symmetric key and using symmetric encryption to encrypt msg.

① Encrypt (Pk, very long M):

generate $ Sym key k (AES-CTR)

$\underbrace{Enc_{sym}(k, M)}_{C_1}$ $\underbrace{Enc_{Pub}(Pk, k)}_{C_2}$

② Decrypt (Sk, $(C_1; C_2)$)

$Dec_{Pub}(Sk, C_2)$

$Dec_{sym}(Pk, C_1)$

- El Gamel Security
  - i) Discrete log problem holds.
  - ii) $g, p, g^k, C_1, C_2 \xleftrightarrow{\text{indistinguishable}} g, p, g^k, C_1, R$ (R: \$ value.)

[Public key Exchange]

- 密鑰交換: Deffie Hellman 不能在發送接收 key.
  El Gamel 可以密鑰交換.

☆ Integrity: modify message

- Cryptographic Hash Functions.

$H: \{0,1\}^* \longrightarrow \{0,1\}^L$

SHA256 256 bits

→ deterministic

SHA-1, SHA-2, SHA-3.

$H(x)$ = hash of x
digest
fingerprint.

Property:

① correctness: deterministic

② Efficiency: computing H should be easy

③ Security: i) one-way: "preimage resistance"

$\forall Adv, Pr[x \leftarrow \$ rand; y = H(x); Adv(y) \to x]$
$=$ negligible $(\frac{1}{\geq \text{size of output of } H})$

$\forall Adv, \forall x, Pr[Adv(H(x)) \to x] = negl$ → not attainable

"2" → H(2)

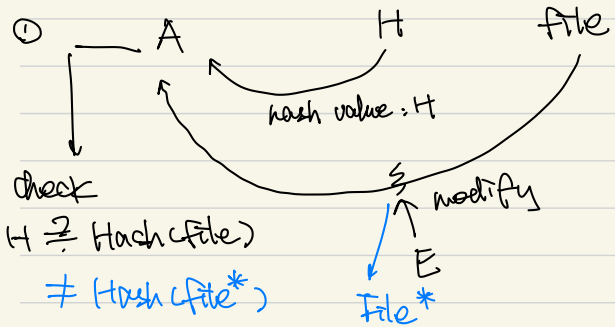
↳ can invert for some specific value, but hard for random value.

2) collision-resistance.

it is infeasible to find (x, x') s.t. $(x \neq x')$ and $H(x) = H(x')$

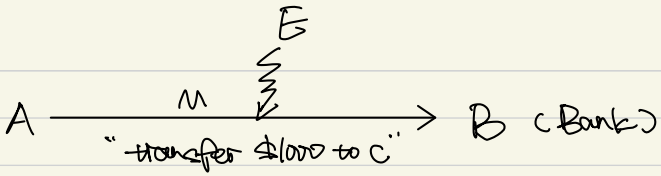
SHA256 is assumed currently to be CR

Application:



⇒ collision resistense

② password encryption ⇒ ties on one-way func



authenticity: M from the expected user

integrity: M was not modified

- El Gamel: $(g^r \bmod p ; M \cdot g^k \bmod p)$
  - ↳ ×2: $2M \cdot g^k \bmod p$ ⇒ $2000
- Encryption does not provide authenticity or integrity.

sum:

	Symmetric-key	Asymmetric-key
Confidentiality	symmetric key encryption (AES-CBC)	public key encryption (El Gamel)
Integrity & authenticity	MAC (AES-EMAC)	Digital Signature (RSA)

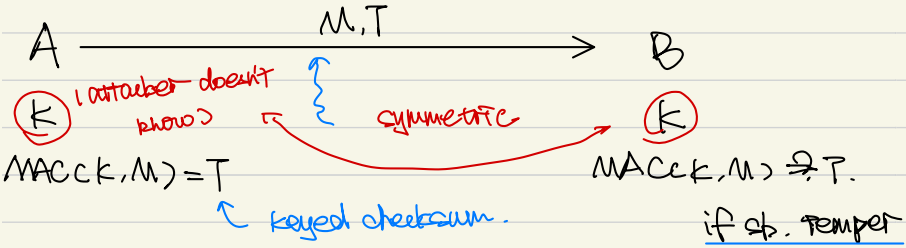
- MAC (Message authentication code)

Goal: hard to forge a valid tag without knowing the secret key.

⇒ ensure integrity / authenticity of a msg using symmetric key.

① HMAC: use hash func

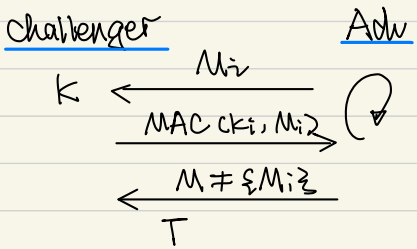
② EMAC: use block ciphers



Correctness: deterministic ⇒ not IND-CPA (privacy)

efficiency: compute mac should be poly time

security: EU-CPA existentially-unforgeable under chosen plaintext attack 不可伪造



$\forall$ Adv. $Pr[k \leftarrow keygen(); Adv() \rightarrow (M, T)$ s.t. $M \neq M_i;$

$MAC(k, M) = T] = negl.$

⇒ Adv can not forge a MAC for a new message.

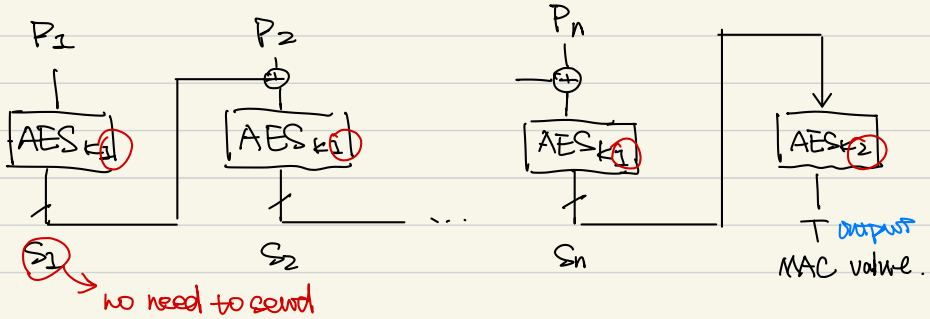
- AES-EMAC

unforgeable.

Keygen(): choose 2 # keys (256) $(k_1, k_2) = k$

MAC(k, M): (128)

split M into $P_1 || P_2 \cdots || P_N$



* Propose $MAC'(k, M) = (S_1, \cdots, S_n)$

can forge MAC': Adv asks MAC, $M = (P_1 || P_2) \rightarrow (S_1, S_2)$

$\Rightarrow$ $M^* = P_1$; MAC is S_1.

$\Rightarrow$ insecure

* Propose $MAC''(k, M) = S_n$.

Adv. asks for $P_1 \xrightarrow{\text{receive}} S_1$

$P_1' \rightarrow S_1'$

$P_1 || P_2 \rightarrow T$

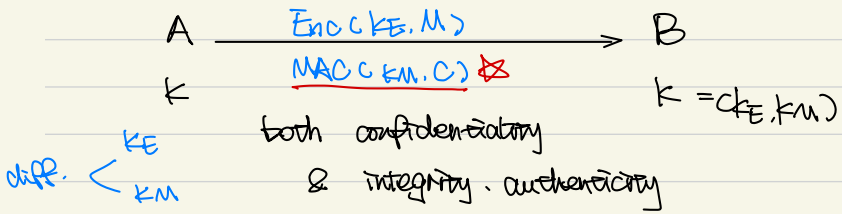
Adv forges MAC for $(P_1'; S_1 \oplus P_2 \oplus S_1')$ is T

P_1 猜猜.

$P_1' \rightarrow S_1' \oplus S_1 \oplus P_2 \oplus S_1' \rightarrow AES(S_1 \oplus P_2) = AES(AES(P_1) \oplus P_2) = T$

$\Rightarrow$ forgeable $\rightarrow$ points to needing k_2 at the end.

o authenticated encryption.



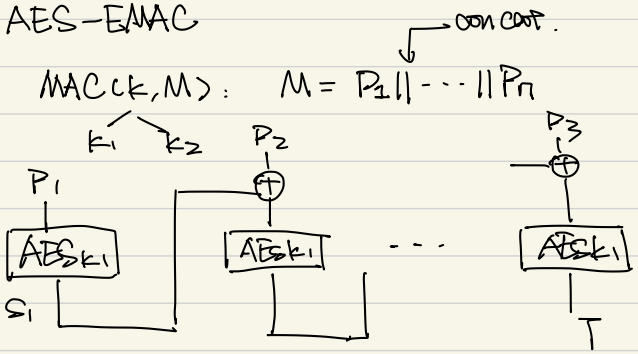
MAC在加密文本基础上生成，否则

(MAC(kM, M) 不满足 IND-CPA，即 confidentiality.

⇒ 根据 (MAC(kM, M) 是否相同，就可 distinguish.

Lec 9

① AES-EMAC



collision resistent

Consider $H(M) = \underbrace{MAC(k, M)}$

Hash definition ⇒ k known to the attacker

$P_1 || P_2 \cdots || P_n \to T$

$(S_1 \oplus P_2) || \cdots || P_n \to T$

② HMAC both a MAC and collision resistent when the attacker has key k

inner-padding ↑ out-padding

$HMAC(k, M) = H(k \oplus opad \,||\, H(k \oplus ipad \,||\, M))$

$0x5C..5C$ $0x36..36$

- Assume H is collision consistent

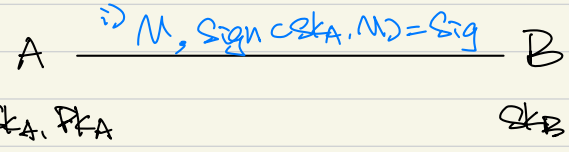
suppose $HMAC(k, M_1) = HMAC(k, M_2)$

$\Rightarrow k \oplus opad \,||\, H(k \oplus ipad \,||\, M_1) = k \oplus opad \,||\, H(k \oplus ipad \,||\, M_2)$

$\Rightarrow H(k \oplus ipad \,||\, M_1) = H(k \oplus ipad \,||\, M_2)$

$\Rightarrow M_1 = M_2$

③ Digital Signatures : integrity & authenticity in the assignment setting.



Syntax:

KeyGen() → Sk, Pk

Sign(Sk, M) → Sig

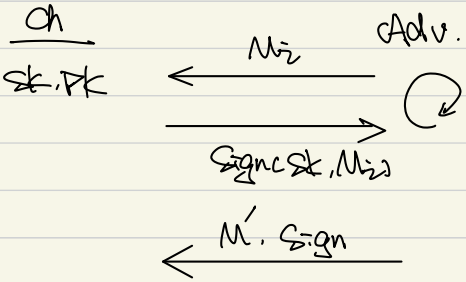
Verify(Pk, m, Sig) → 0/1

Correctness: $\forall$ m. Sk. Pk

Verify(Pk, m, Sign(Sk, M)) = 1

Security: EU-CPA

existential unforgable under CPA



Adv wins if $M' \neq \{M_i\}$ and Verify(Pk, M', Sig) = 1.

$\forall$ Adv. Pr[Adv wins] < negl

③ RSA Signature Scheme security.

i) keygen(): pick two $ primes p and q of 1024 bits (both 2 mod 3) — secret ☆ — 都3互质

$n = p \cdot q = Pk$

$\phi(n)$ = Euler's ~~totient~~ function.

= # of integers ≥ 0 that are $\gcd(\cdot, n) = 1$

$\phi(n) = (p-1)(q-1)$ order of group modulo n. ⇒ 小于等于n且和n互质的数有多少个

$\forall a,\ a^{\phi(n)} \equiv 1 \bmod n$ → $3d = 1\phi(n) + 1$

compute d s.t $3d \equiv 1 \bmod \phi(n)$

$sk = d$

ii) $Sign(sk, m) = hash(m)^d \bmod n$.

$Verify(Pk, m, sig)$: $sig^3 \bmod n \stackrel{?}{=} H(m) \bmod n$.

- Correctness

$$(hash(m)^d)^3 \bmod n = hash(m)^{3d} \bmod n$$
$$= hash(m)^{1\phi(n)+1} \bmod n$$
$$= (hash(m)^{\phi(n)})^1 \cdot hash(m) \bmod n$$
$$= hash(m) \bmod n$$

goal: $(M^e)^d \equiv M \bmod n$.

Euler's $a^{\phi(n)} \equiv 1 \bmod m$ if a and n are coprime

$\Rightarrow a^{k\phi(n)} \equiv 1 \bmod m$

$\Rightarrow a^{k\phi(n)+1} \equiv a \bmod m$

$k\phi(n) + 1 = ed$.

$\Rightarrow ed \equiv 1 \bmod \phi(n)$

$\Rightarrow \boxed{d = e^{-1} \bmod \phi(n)} \Rightarrow$ Sk

let $e = 3$

$M^{k\phi(n)+1} \bmod n$

$= M^{k\phi(n)} \cdot M \bmod n$

$= M \bmod n$

M: Hash(m)

⑭ RSA Signature Scheme Security

* why hash m. → how to forge the signature. ?

  $Sign(sk, m) = m^d \bmod n$.

  forge m:

  signature for I is I

  0 is 0.

RSA unforgeable ⇒ difficult to factor n.

if can factor n ⇒ get p, q ⇒ get $\phi(n)$

⇒ get d: which is the secret key ⇒ able to sign().

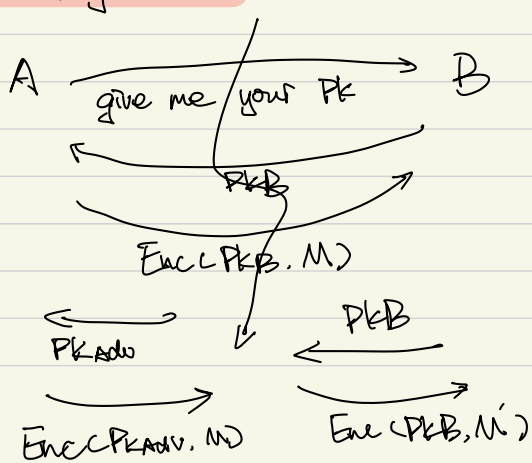
* necessary assumption for security:

  No adversary can factor large numbers.

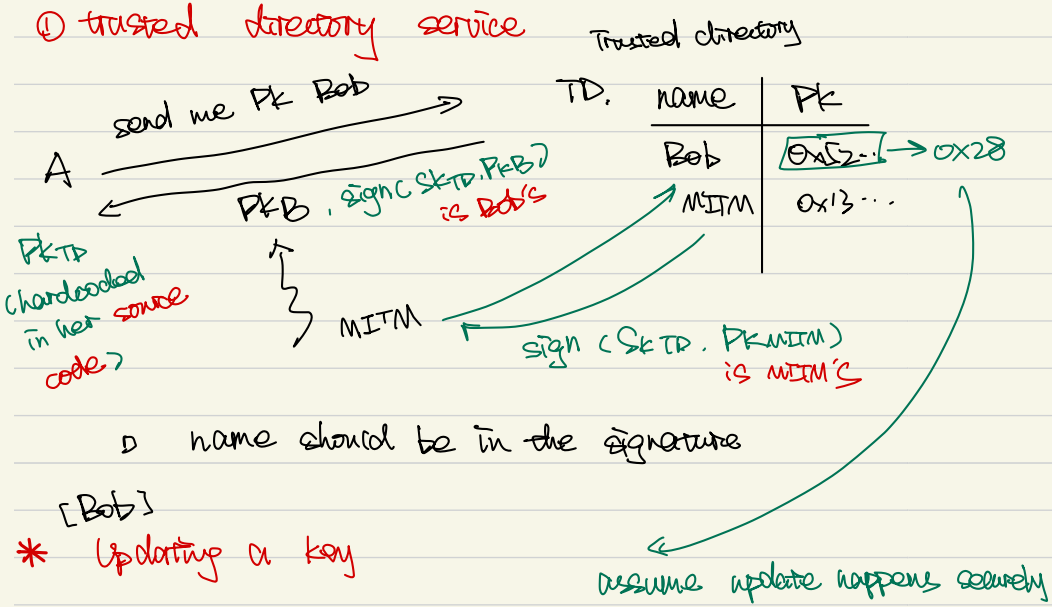
  Difficulty of factoring problem

key Management.

MITM



① trusted directory service



- name should be in the signature

[Bob]

* Updating a key

→ Replay attack:

* Bob update his key but
attacker replays old information (old sig with old Pk)

* A doesn't know Bob update his key

⇒ A embeds # value nonce in the request.

⇒ A checks sig from TD to contain nonce & to verify with PkTD & contains B's name

⇒ ☆ goal: PkB is latest

* drawback of TD:

- scalability
- TD is a central point of attack / trust
- difficult to recover from TD compromise
- updating key requires trust
- TD has to be always available
  * central point of failure.

Approach 2: Digital Certificate

⇒ association between name & PK by a CA (certificate authority)

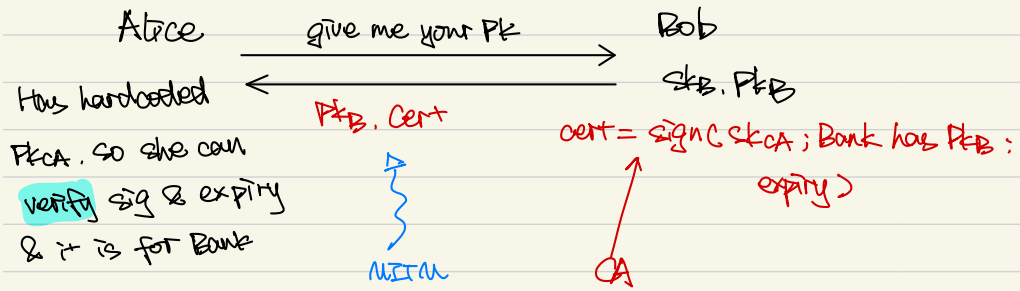
* certificate: sign (SK_{CA}, Bob's PK is 0x.... , expiry date)

assume browser have PK_{CA} hardcoded.

pro: [local servers] Anyone can serve PK_{Bob}, $cert_{Bob}$ ⇒ reliability available.

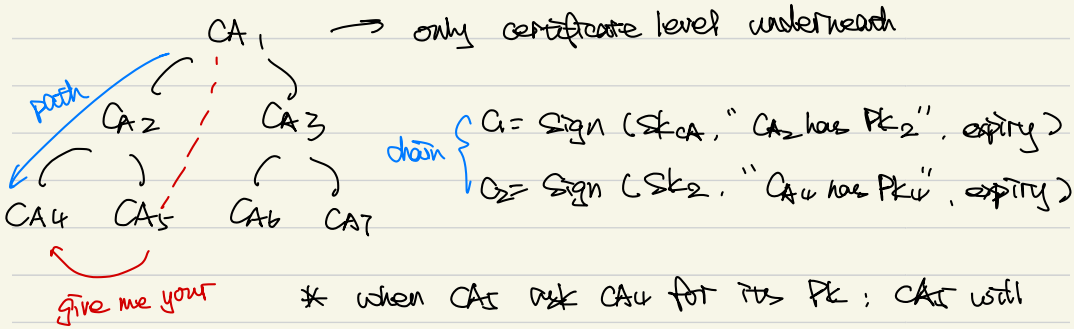
con: central point attack

* check: $cert_{Bob}$ verifies with PK_{CA} is not expired. is for Bob.



- pro: — can contact bank (or anyone) to obtain PK
  con: — one CA certificate all.

Certificate hierarchies & chains



* when CA5 ask CA4 for its PK: CA5 will receive C1, C2, PK4, PK2 ⇒ hardcoded CA1.
  i) use PK1 + C1 to check PK2
  ii) use PK2 + C2 to check PK4.

Revocation

how to revoke a certificate that has not yet expired ?

— wait till expiry. making expiry shorter

— revocation lists : CA push revocation

sign (SK$_{CA}$, "revoke cert") into browers.

not ideal. depends on browers to download it.

* CA can be compromised or could be deceived to sign incorrect certificate.

— transparency logs promise to address this problem.

Password Management

usability and security problem

- online guessing attack. ⇒ rate-limiting / captchas
- social engineering and phishing
- eavesdropping : plaintext. ⇒ TLS
- client-side malware. ⇒ two-factor authentication.
- server compromise

* mitigation for server compromise ⇒ × plaintext

Hashing Passwords.

* why not use encryption?

  ⇒ need to store key in somewhere.

* why hash?

  ⇒ one way, no key. hard to recover

- hash (password) for each user using a cryptographic hash function.
- offline password attack:
  - dictionary attack. enumerate all passwords against each hash(w)
- Amortized password hashing
  - D is dictionary size, n is number of hashed password.
  - build table (H(password), password) for all 2^{80} passwords
  - One brute force all hashes

2^{80} ↓ 50%.

D.N.

* Salted Hashes.
  - randomize hashes with salt
  - store (salt, hash(password, salt)), salt is random

⇒ 需要密码的随机性，⇒ 生成的table多对一个用户有用

* slow hashes
  - H(x) = hash(hash(hash(...hash(x)...))) x ∈ password (1 save.
  - time for authentication v.s. attacker time

* 抵抗重放攻击 (replay attack).

— Add a non-repeating nonce or timestamp in the MAC or in the signature

— $m \rightarrow$ timestamp $\| m$.