Special Topics in Cryptography

Mohammad Mahmoody

Last time

- Pseudorandom Functions
- PRFs → CPA secure encryption

Today

- Authentication (MAC) using shared keys
- Getting MACs from PRFs
- Security against active attacks (CCA security)

PS 2 extension

- Due end (10pm) of 28th (Wed).

PS2 clarification for problem 3

(x, y)

$m = m_1 m_2 m_3 \ldots m_\ell = m_1, m_2, \ldots, m_\ell = m_1 \| m_2 \ldots \| m_\ell$

- $\mathrm{Enc}(key, m; r) = [\mathrm{Enc}(key, m_1; r_1) \,||\, \ldots \,||\, \mathrm{Enc}(key, m_\ell; r_\ell)]$

$r = (r_1, r_2, \ldots, r_\ell)$

CPA security scales.

Single-message security does NOT scale.

$\mathrm{Enc}(k, [m_1, m_2, m_\ell]) = (\mathrm{Enc}(k, m_1), \mathrm{Enc}(k, m_2) \ldots)$

$m = [m_1, m_1]$

$m' = [m_1 \neq m_2]$

Review: randomness in encryption

$\mathrm{Enc}(k, m; \underline{r})$

$\mathrm{Enc}_k(m) \to c$ (arrow means randomized)

- Encryption's own randomness is usually *not* revealed (even though we did reveal it in our specific construction last time)

if $F_k(x) \to y$ (k: secret key), $x \in \{0,1\}^*$, $|k| = n$: sec param. $|y| = \ell$

to encrypt $|m| = \ell$:

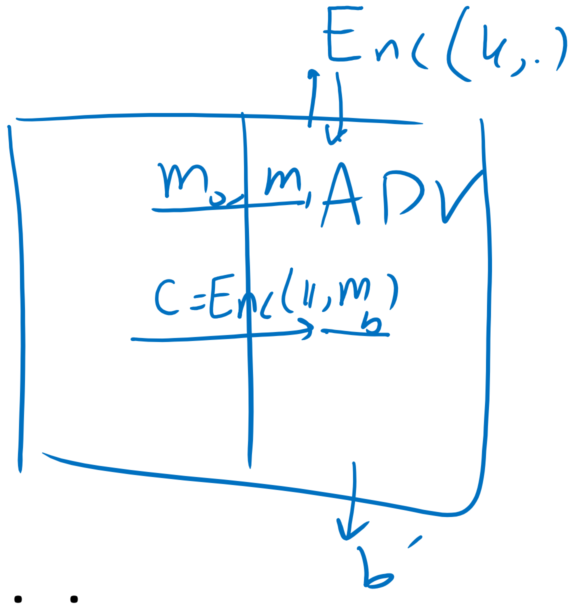
Pick r of length n

output $c = [r, m \oplus F_k(r)]$ ($\oplus$: bitwise XOR; k: secret key)

$\mathrm{Dec}(k, c)$, $c = [r, y]$

output $m = F_k(r) \oplus y$

What CPA security guarantees



- It guarantees multi-message security (passive attacker)
- It also guarantees a semi-active attacker (somehow obtaining encryptions of messages that they choose.)
- It does not say anything about "active" attacks. What are they?

What could go wrong with a CPA secure scheme?

① resending a message.

② re-sending fresh encryptions?

③ modify the message : flip the last bit of ciphertext.

$c = [r, \underbrace{m \oplus F_k(r)}_{y}]$

$y \rightarrow$ flip the last bit of y

c'

if Bob decrypts $c' \rightarrow m' = ?$ m but last bit flipped.

Authentication:

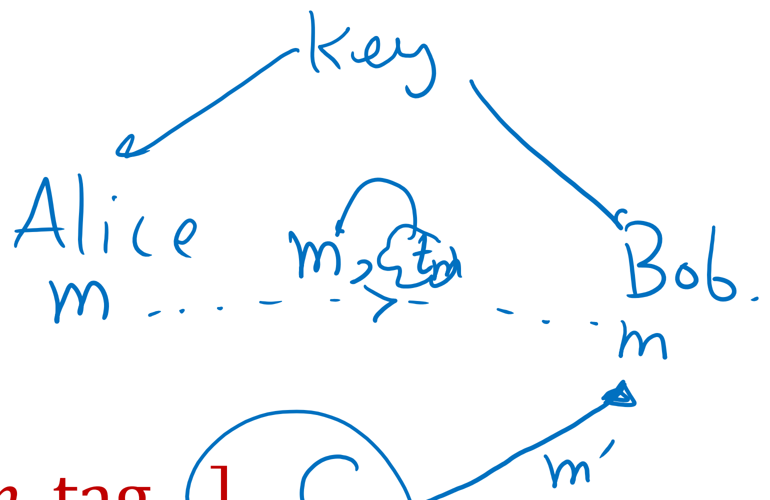
How would Bob know Alice sent this message?

… if Eve is not passive anymore…

Authentication

- Could be applied to ciphertexts, but it is a meaningful notion on its own, even for plaintexts without any encryption involved…
- In the private-key (i.e. symmetric-key) setting it is called: **Message Authentication Code (MAC)**
- There is a "public-key" version of the same thing known as: "Digital Signatures". We will talk about it later.
- If combined with CPA-secure encryption **properly**, gives rise to a more secure encryption that handles "active" attacks as well..

Message Authentication Code



- Alice and Bob share key k.

potentially randomness could be there too

- Alice generates $\mathrm{MAC}_k(m) \rightarrow \mathrm{tag}_m$ and sends: $[m, \mathrm{tag}_m]$
- Bob receives $[m, \mathrm{tag}_m]$ runs $\mathrm{Verify}_k(m, \mathrm{tag}_m)$ and accepts or rejects

not knowing key.

Completeness: $\forall m, \forall k, \mathrm{Verf}_k[m, \mathrm{MAC}_k(m)] = 1$

Soundness: How to define security?

- Infeasible for Adv to generate a valid $[m, \mathrm{tag}_m]$ (Bob thinks so)
- Adv gets to see $[m, \mathrm{tag}_m]$ for many chosen m's before forging for a **new** $\boldsymbol{m}$

① number — poly(n) many $[m, \mathrm{tag}_m]$ seen by adv.

② :

Formal definition of security

Sec. game for MAC

The message authentication experiment $\mathsf{Mac\text{-}forge}_{\mathcal{A},\Pi}(n)$**:**

1. *A key k is generated by running* $\mathsf{Gen}(1^n)$.
2. *The adversary $\mathcal{A}$ is given input 1^n and oracle access to* $\mathsf{Mac}_k(\cdot)$*. The adversary eventually outputs (m, t). Let $\mathcal{Q}$ denote the set of all queries that $\mathcal{A}$ asked to its oracle.*
3. *$\mathcal{A}$* succeeds *if and only if (1)* $\mathsf{Vrfy}_k(m, t) = 1$ *and (2) $m \notin \mathcal{Q}$. In that case the output of the experiment is defined to be 1.*

Q: think about it as list of messages honestly tagged by Alice using key k

think of it as the new message chosen by Adv

DEFINITION 4.2 *A message authentication code* $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ *is* existentially unforgeable under an adaptive chosen-message attack, *or just* secure, *if for all probabilistic polynomial-time adversaries $\mathcal{A}$, there is a negligible function* negl *such that:*

$$\Pr[\mathsf{Mac\text{-}forge}_{\mathcal{A},\Pi}(n) = 1] \leq \mathsf{negl}(n).$$

Constructing MACs using PRFs

- Suppose $F_k(\cdot)$ is a PRF with key, input, output lengths $n, *, \ell$ = n
- How do we generate MAC tags for messages?

arbitrary

$\{0,1\}^\ell$

k

Alice: $MAC_k(m) \to t_m$

Bob: $Verf_k(m, t_m) \to \{0, 1\}$

$tag_m = F_k(m)$

$Vrf_k(m,t)$: 1 iff $F_k(m) = t$; 0 oth.

$\ell = ?$ $\ell = |m|$; $\ell = n$

$\ell \geq lg^2(n)$

$\ell < O(lg(n))$ bad!

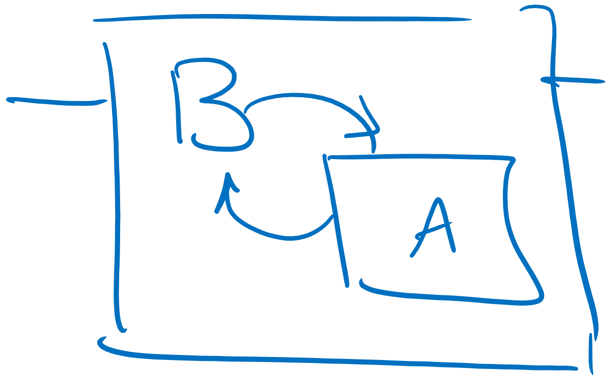
Proof of Security

Start by assuming
$\exists$ poly-time (poly(n)) Adv. A
ε- breaking our scheme Π.
$\varepsilon(n) \geq \frac{1}{\text{poly}(n)}$

→ then we show

PRF F secure → our Π (MAC, Vrfy) is secure.

→ if Π is NOT secure.
→ F is also not secure.



B breaks PRF in poly time by ε' chance

$\varepsilon' \geq \frac{1}{\text{poly}}$

Proof of Security

Ideal World: (n)

$F_k(\cdot)$ is substituted with $R(\cdot)$

R is a function (randomly chosen) from all functions mapping $\{0,1\}^n$ to $\{0,1\}^n$

R has no pre-chosen answer. given any x. it pick $R(x)$ at random and saves it in case x is asked again.

Real (n): $MAC(\cdot) = F_k(\cdot)$, $k \leftarrow \$$

ADV sends m_0, receives tag_{m_0}; ADV outputs m, t

wins if $t = F_k(m)$. also $m \notin Q = \{$set of queries of ADV$\}$

P_R: prob winning in Real

Ideal (n): ADV sends m, receives $R(m) \leftarrow \{0,1\}^n$; ADV outputs m, t

wins if $m \notin Q = \{$set of queries asked$\}$, $R(m) \stackrel{?}{=} t$

(P_I)

Goal 1: $P_I <$ neg.

Goal 2: $|P_I - P_R| =$ neg.

Proof. $R(m)$ is picked randomly independent of t

$\Pr[R(m) = t] = 2^{-n}$

Formally: attacker A_{PRF} first runs ADV $\rightarrow m, t$

then A_{PRF} asks m from Oracle $O(m) \stackrel{?}{=} t$

if they were equal $\longrightarrow$ output $b = 1$

if $|P_I - P_R| >$ non-neg. $\rightarrow$ A_{PRF} wins in breaking PRF $F_k(\cdot)$

Chosen cipher-text security:

- combining CPA security with MACs to handle active attacks.

Password verification example