CAS CS 357

In-Class Note 11

1. Several important crypto properties
    a. Integrity: message is not altered
    b. Authenticity: message came from the correct source
2. How do we define a symmetric MAC (message authentication code)?

Message m	key k	tag $t = MAC_k(m)$	$Ver_k(m,t) = \{0,1\}$
Alice	k	$t = MAC_k(m)$	$Ver_k(m,t)$? 1

    a. Correctness: $Ver_k(m,MAC_k(m)) = 1 \rightarrow$ validly MAC'd message verifies correctly
    b. Security: The adversary cannot forge the MAC without k
    c. This is similar to the idea of signing
    d. The verification would output 1 if valid or 0 otherwise
    e. Assures both integrity and authenticity
    f. The adversary can create a new message or modifies the existing message:
        - Security would be broken if adversary created his own message m*,t* such that $Ver_k(m^*,t^*) = 1$ (adversary cannot create her own message)
        - Security would be broken if adversary modified the existing message m_3,k_3 into m*, t* such that $Ver(m_3,k_3) \rightarrow (Ver_k(m^*,t^*) = 1$ (adversary cannot alter the message)
    g. Definition does not include confidentiality (adversary can still read the message sent)

3. What the adversary does
    a. Key recovery: if the adversary somehow learns the key, the security is broken (Kirckoff's principle ensures that the adversary knows everything about the security algorithm without the key, but if adversary learns the key, the security is broken)
    b. Forges a "target message" (a message that is selected for the adversary): Adversary can receive message m* and produce t* such that $Ver_k(m^*, t^*) = 1$
    c. Forges a chosen message (a message that the adversary selects): Adversary produces m*,t* such that $Ver_k(m^*,t^*)=1$ → typically used in our definition of security
    d. Adversary forging a "target message" → Adversary forging a chosen message (since if adversary can forge the message that is given by others, the idea is that the adversary can select the same message to forge)
    e. Choose the definition that is easiest for the adversary (even if the method is easy, the idea is that adversaries should not break the security)
4. What the adversary learns:
    a. Some valid message-tag pairs:

        The adversary learns few message-tag pairs (m_1,t_1), (m_2, t_2), (m_n,t_n) such that $Ver_k(m_i,t_i)=1$

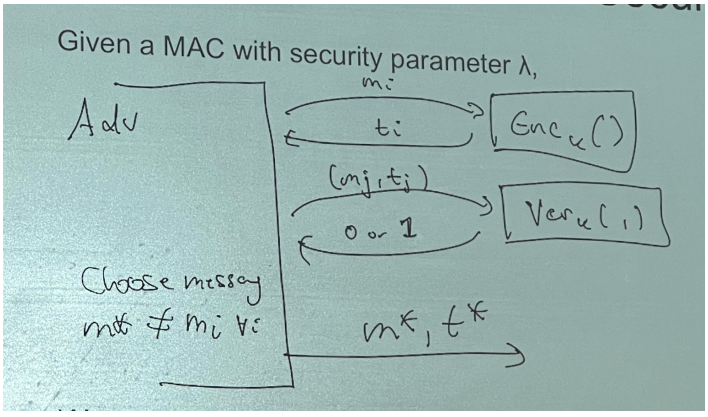
        After learning the messages, the adversary chooses m*,t* and checks its verification
    b. The tags for any message of the adversary's choosing:

        The adversary learns few tags of the messages

The adversary chooses the message and learns the tag of the chosen messages

Adversary → m_i → $MAC_k()$

$MAC_k()$ → t_i → adversary (such that $Ver_k(m_i,t_i) = 1$)

5. CMA-security MACs (definition)
    a. Given a MAC with security parameter λ, an adversary running in time polynomial in λ cannot forge a message of its own choosing even if the adversary has the power to
        - choose messages and see their tags AND
        - To know whether a give message-tag pair is valid or not
6. Actual definition of "CMA-secure" MACs
    a. Adversary gets to generate his own messages and learn its tags by sending it to MAC
    b. Adversary gets to generate his own message and tag pairs and learns whether the message-tag pair generates 0 or 1 by sending it to Ver_k
    c. 

    d. Pr[Adv outputs m*,t* such that Verk(m*,t*) = 1] > 2^(-λ)
7. Contract signing protocol (scheme example)
    a. m_1, m_2, m_3 → 3 page contract

b. "MAC of the contract" m_1, m_2, m_3, t_3 where $t_3 = MAC_k(m_3)$ where MAC is secure one

c. The problem here is that m_1 and m_2 are floating

d. Construct adversary that wins the game above (show insecurity of the contract signing protocol)

- Adversary chooses m_a, m_b, m_c and send it to MAC → the adversary receives m_a, m_b, m_c, t_c
- Adversary outputs m_a', m_b, m_c, t_c (where the first page is different) → Ver is 1
- The idea here is to modify the first two messages (m_a or m_b) since the third message is the only message that is tagged with the key
- Adversary wins with the probability of 1 (attack works 100%)
- **The idea is that people need to MAC the entire message, not only parts of the message**

8. Cryptographic hash functions (MACs are made from hash functions)

a. Roughly speaking: cryptographic hash function is an algorithm H such that the mapping from input m to output H(m) "looks random"

b. Function where every input maps to an output (multiple inputs can output same output) → many to one relationship

c. *Important* → A cryptographic hash function is not keyed

d. One way property of cryptographic hash function: H:{0,1}^n → {0,1}^256

e. Easy: input x → H() → output y

f. Difficult: y → ___ → x (such that H(x)=y) → difficult to go back from the output and find out the exact input (decryption part is difficult)

g. More formally, there does not exist an adversary running in polynomial time that given y, can find a valid x such that H(x) = y

9. Properties of cryptographic hash functions

a. Collision resistance: It is hard to find two "preimages" x, x' where x does not equal x' but H(x) = H(x')

b. Hard to find x does not equal x' such that H(x) = H(x')

10. HMAC

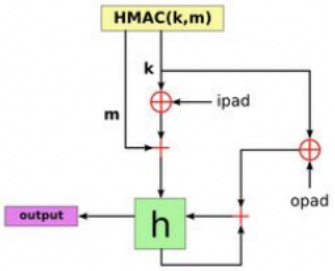
a. HMAC is an algorithm that satisfies the definition for a CMA-secure MAC

b. HMAC uses a key K and a cryptographic hash function H (HMAC is sometimes called a "keyed cryptographic hash function")

$$\mathrm{HMAC}(K, m) = \mathrm{H}\left(\left(K' \oplus opad\right) \parallel \mathrm{H}\left(\left(K' \oplus ipad\right) \parallel m\right)\right)$$

$$K' = \begin{cases} \mathrm{H}(K) & K \text{ is larger than block size} \\ K & \text{otherwise} \end{cases}$$

c.



d.