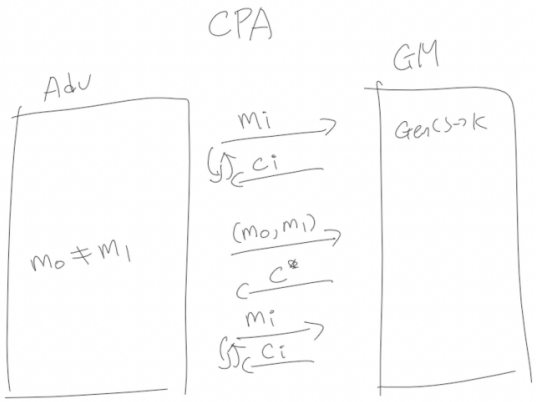
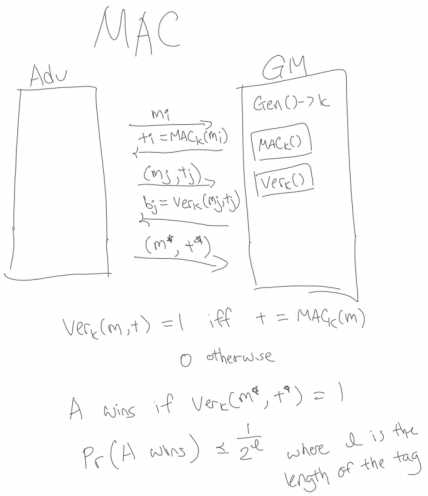
CAS CS 357

Discussion Worksheet 2

- CPA
  - 

  - The user can pass in the message that the user already asked in the past prior to the game master (the idea is that even if the user already sent the message, the encryption is randomized so that it does not produce any pattern or same encrypted message → sending the same message again does not matter)
- MAC
  - 

1. In a Caesar cipher, each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. For example, with a left shift of 3, D would be replaced by A, E would become B, and so on. The key is the number of positions used in the left shift. This cipher is named after Julius Caesar, who used it in his private correspondence.
    a. Present an attack that proves that the Caesar's cipher is NOT CPA secure

        We can extract the key.

        Assume that our message has the length of four

        Adv → AAAA (message) → Gamemaster

        Gamemaster → c (encrypted message) → Adv

        From the encrypted message we received, we can see how many digits are shifted by comparing it with the message we sent. For example, if we get back CCCC, we can see that there is a right shift of 2. Knowing how many shifts occur, we can always win the game.

        OR

        The adversary generates a message m and query for the ciphertext of m. Adversary then sets m0 to m and m1 to some other message. If c*=c, then output 0, otherwise output 1 → The ciphertext shifts to the right or left same number of digits every time → by asking the message m prior to the game and sending it once again during the game, the adversary can always win

2. Let E be any encryption scheme that takes in a key of length λ and message of length l and produces ciphertexts of length l'. Assume that E is CPA secure (E might be encryption in CBC mode, as discussed in class.). Now let E' be an encryption scheme identical to E except with the following modification; the 3rd bit in the ciphertext is equal to the 3rd bit of the plaintext.

    a. Show that E' is not CPA secure

    Adversary can send two messages m_0 and m_1 where the 3rd bit of m_0 equals 0 and the 3rd bit of m_1 equals 1.

    Game master returns back the encrypted message, but if the 3rd bit is 0, that message is m_0 and if 3rd bit is 1, the message is m_1, knowing that the 3rd bit of the encrypted text equals the 3rd bit of the plaintext.

    We always win the game.

3. Alice and Bob share a secret 128-bit key k that they will use to authenticate every message that they send. Every time Alice wants to send a message m to Bob, she breaks the message m up into blocks $m_1, m_2, \ldots, m_n$ and outputs the tag for each block $t_1, t_2, \ldots, t_n$, where $t_i = MAC_k(m_i, i)$ for all i. Alice sends $m_1, m_2, \ldots, m_n$ and $t_1, t_2, \ldots, t_n$ to Bob.

   a. Write down the verification algorithm for this scheme

      Bob:

      For i in [1,n]:

      $T_i' = MAC_k(m_i,i)$

      Check if for all i, $t_i' = t_i$

      Why it works?

      $M_0 \rightarrow m_{0,0} \quad m_{0,1} \quad m_{0,2} \quad \ldots \quad m_{0,n}$

      $T_{0,0} \quad t_{0,1} \quad t_{0,2} \quad \ldots \quad t_{0,n}$

      $T_{0,i} = MACk(m_{0,i},i)$

   b. Prove that the scheme is not a secure MAC

      Adversary sends m_0 to the game master and game master returns t_0 to adversary

      Adversary sends m_1 to the game master and game master returns t_1 to adversary

      Within the game, the adversary can send m* such that m* consists of first half of m_0 and second half of m_1 (combination of message m_0 and m_1) with the tag t* such that t* consists of first half of t_0 and the second half of t_1 (the corresponding sections to the message). The notion here is that the tag has to have specific and sophisticated binding instead of simply the indexes of the messages.

Exercise 4. An airline uses *manifests* to determine which passenger should be on which flight. The airline has the secret key k. Each manifest consists of:

- The flight number x and its date and time d
- A MAC $t = \mathsf{MAC}_k(x||d)$.
- The name of the 1st passenger p_1, and a MAC tag $t_1 = \mathsf{MAC}_{SK}(p_1)$.
- The name of the 2nd passenger p_2, and a MAC tag $t_2 = \mathsf{MAC}_{SK}(p_2)$.

  $\vdots$
- The name of the n-th passenger p_n, and a MAC tag $t_n = \mathsf{MAC}_{SK}(p_n)$.

Notice that n will be different for each flight.

The manifest is checked, using the key k, as passengers board the flight.

1. Suppose you can intercept and modify manifests before they arrive at each flight. Explain how you can travel to Tokyo for the cost of a flight to Chicago.
2. From now on, we modify the manifest to be
   - The flight number x and its date and time d
   - A value $h = H(x||d)$ where H is a collision-resistant hash.
   - The name of the 1st passenger p_1, and a MAC tag $t_n = \mathsf{MAC}_k(h||p_1)$.
   - The name of the 2nd passenger p_2, and a MAC tag $t_2 = \mathsf{MAC}_k(h||p_2)$.

     $\vdots$
   - The name of the n-th passenger p_n, and a MAC tag $t_n = \mathsf{MAC}_k(h||p_n)$.

   Notice that n will be different for each flight.
3. Explain why your attack from the previous part no longer works.
4. However, there is still something terrible about this scheme.
   The number of passengers on the flight, n, is never signed.
   Present an attack that demonstrate why this is a problem.

4.

a. You can move your tag from the manifest for the flight to Chicago to the manifest for a flight to Tokyo and it will verify, since the MAC doesn't authenticate information about the flight or date and time

b. This will no longer work because the tag includes a hash of the flight number and date and time, therefore the tag you obtain in the manifest for the flight to Chicago is not valid (will not verify) in the manifest for a flight to Tokyo
c. You can remove every passengers in the manifest and there's no way for the airline to verify that there was initially n passengers on the flight