1-23-2017: Introduction to Applied Cryptography

Tuesday, January 24, 2017 1:53 PM

- **Logistics**
  - Lab 1 will be released tonight (1-23-2017)
    - Lab will be graded, so do it!
    - Collaboration is encouraged, but don't plagiarize!
  - The 2 groups of 2 for the presentation need to become 1 group
  - First part of the class (applied cryptography) is the part that will be on the midterm
- **Introduction to Encryption**
  - Definition of Encryption: you want to ensure **confidentiality, authentication and integrity**
    - Confidentiality: Make sure that nobody can read the message except for the sender and receiver
    - Authentication: Ensure the message came from the correct person
    - Integrity: Ensure that the message is unchanged from how it was originally sent
    - We do these things with Message Authentication Codes and Digital Signatures
  - We have had cryptography since ancient times
    - Documents were signed with a seal that only the king could use, and thus the seal served as "authentication"
  - One of the very first forms of encryption: Caesar Cipher
    - Shift letters in message by a fixed value for all letters in the word
  - Let's break a Caesar Cipher right now!
    - "Encrypted" message **(cipher text)**: br zkr kbr pbrrnr
    - Every letter in this **cipher text** corresponds to a letter in the alphabet
      - Example: maybe Q corresponds to an I in plaintext
    - We can use intuition; for example, vowels are very common, and E is the most common vowel. Since R is very common in the cipher text, we can substitute R for E
    - As the message becomes decrypted, we can see more and we eventually decrypt:
      - He ate the cheese.
  - What did we do to break this cipher?
    - We used the intuition that a lot of R's may correspond to a lot of E's in plaintext, and likewise given more context
    - This is known as **frequency analysis**
    - What does this tell us?
      - It's not a good idea to have a one-to-one matching to plaintext values in an encrypted message
    - This is called a **plaintext recovery attack**
  - We define a certain kind of encryption with an **encryption scheme:**

Sender Alice k

Receiver Bob k

$\mathrm{Dec}_k(c) = m'$

plain text m

cipher text $c = \mathrm{Enc}_k(m)$

  - This is known as **symmetric encryption**: both sender and receiver share a key **k** which they use to encrypt a message **m** and decrypt the cipher text **c**.

We want correctness: $\mathrm{Dec}_k\left(\mathrm{Enc}_k(m)\right) = m$

  - How do we ensure correctness? With a **hash function.** We use SHA-256 as an example**:**

$$\text{SHA256}: \{0,1\}^* \longrightarrow \{0,1\}^{256}$$

bit strings
$*$ = of arbitrary length

to

bit strings of length 256

... strings ... to

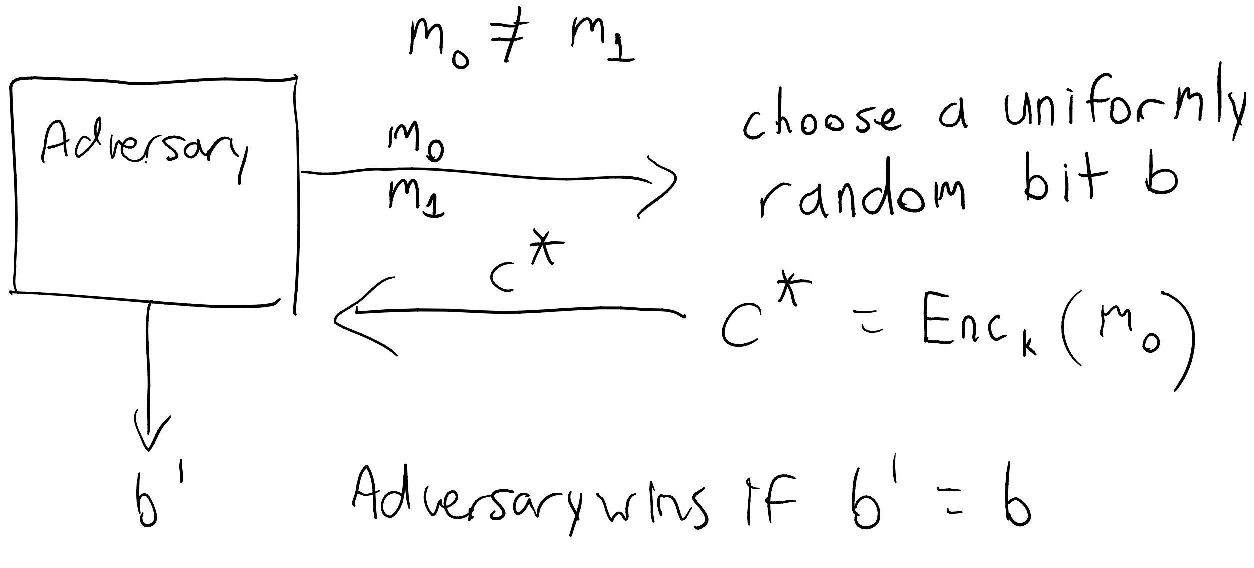
$* =$ of arbitrary length ... of length 256

SHA 256 ("Sharon")

$= \underbrace{0 \ldots 10 \ldots 0 \ldots 0}_{\text{256 bits long}}$

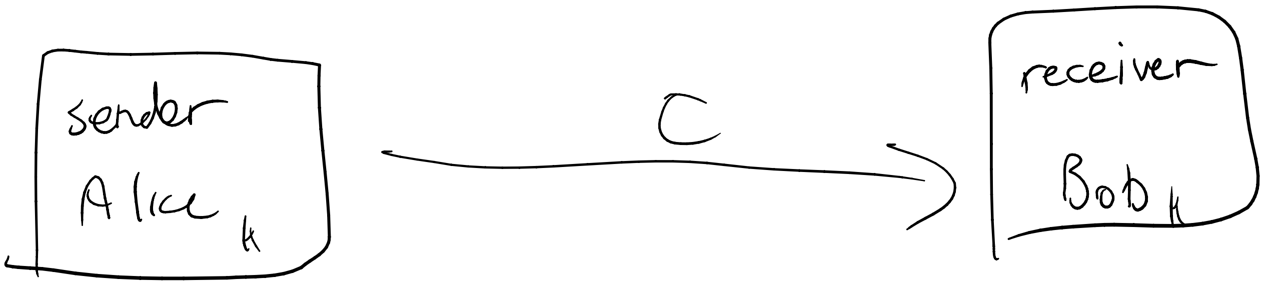
- Maps bit strings of arbitrary length (indicated by *) to bit strings of length 256
- Each input to the hash function maps to the same (unique) output
- BUT, any output does not match to a unique input
- Because there is no mapping from encrypted to decrypted messages, **hash functions are not an encryption scheme.**
- Hash function ensures correctness.

- **Security**
  - Intuition behind different kinds of attacks:
    - **Plaintext recovery**: Given the cipher text, learn the plain text
    - **Key recovery**: Learn the keys
      - *Question:* Which one of these two is easier for the attacker? (assuming they have the exact same amount of information for either attack)
        - *Answer*: Since getting the keys requires getting the plain text anyway, learning the plain text is easier, and plaintext is easier.
    - **Distinguishing attack**:
      - Given cipher text c, determine if it corresponds to message 0 or message 1: (see the diagram)



    - Informally: We have an adversary who
      1) chooses two messages m0 and m1 (where m0 != m1)
      2) Asks for encryption of one of the messages
      3) Need to determine which message was encrypted
    - If the adversary can correctly do this (i.e. output correct b' such that b' = b) then the encryption scheme is broken
    - But, if it can withstand this attack, then it will also withstand plaintext and key recovery attacks

- **Example encryption scheme: One-time pad with one bit (Shannon, 1940s)**
  - Outline:
    - Both the sender and receiver share a random-bit key **k**
    - Sender encrypts the single-bit message **m** by xor'ing the message with k to get the cipher text **c**
    - Receiver decrypts c by xor'ing it with k
  - We can show this is correct with the same correctness test shown above



Key k : random bit

$$k = \begin{cases} 1 \text{ w/ probability } \frac{1}{2} \\ 0 \text{ w/ probability } \frac{1}{2} \end{cases}$$

$$Enc_k(m) = m \oplus k$$

$\oplus$ xor

m is one bit.

$$Dec_k(c) = c \oplus k$$

Ensuring Correctness

$$Dec_k(Enc_k(m)) = m\ ?$$

$$k \oplus (m \oplus k) = m \oplus k \oplus k = m \oplus 0 = m$$

This scheme is correct.

- **To prove security:**
  - For all adversaries, play the "distinguishing attack" game to prove security using this encryption scheme
  - Probability of the adversary guessing correctly must be **no more than 1/2**
  - A practical note: is something that is secure against random messages actually useful?
    - **No!** In the real world, we need to protect structured data like human-readable text

Proof that the one-time pad (for one bit) satisfies the definition of security

Observe that k is a random variable:

$$k = \begin{cases} 1 & \text{with probability } \frac{1}{2} \\ 0 & \text{with probability } \frac{1}{2} \end{cases}$$

The challenger chose to encrypt m_0

$$\Pr[m_0 \oplus k = c^*] = \Pr[\mathrm{Enc}_k(m_0) = c^*] = \Pr[b = 0]$$

$$\Pr[m_0 \oplus m_0 \oplus k = c^* \oplus m_0] = \Pr[k = m_0 \oplus c^*] = \frac{1}{2}$$

Both these values are known to the adversary (in the end, they are just a bit)

No matter what the adversary does, the probability of guessing correctly (from the perspective of the adversary) is still 1/2. This means that it satisfies the definition of security.

- **Why is the one-time pad encryption scheme not ideal?**
  - If we reuse the same one time pad key to encrypt different messages, an adversary could find the key by xor'ing the two messages together
  - One-time pad is almost never used in practice: it's really inefficient