网络安全技术

刘振

上海交通大学 计算机科学与工程系

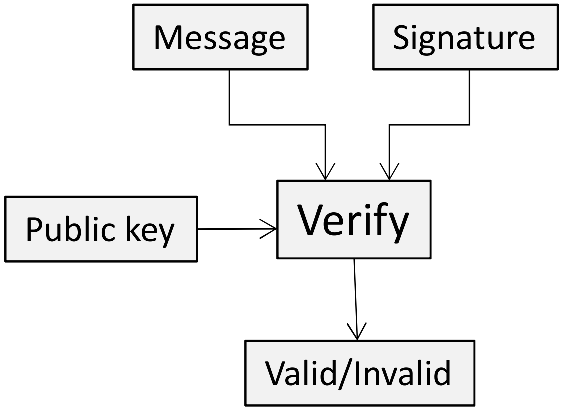
电信群楼3-509

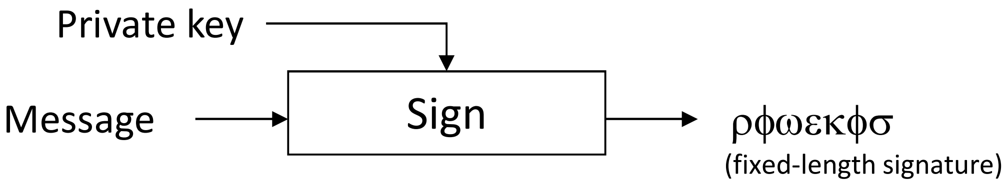
liuzhen@sjtu.edu.cn

Digital Signature
Hash Function
Message Authentication Code

Digital Signature

- There is an electronic document to be sent from Alice to Bob.
- Is there a functional equivalence to a handwritten signature?
  - Easy for Alice to sign on the document
  - But hard for anyone else to forge
  - Easy for Bob or **anyone** to verify
- Answer: digital signature
  - Sign using Alice's private key
  - Verify using Alice's public key





- Only the signer (who has a private key) can generate a valid signature
- Anyone (since the corresponding public key is published) can verify if a signature with respect to a message is valid

RSA Signature Scheme

- **Setup:**
  - $n = pq$ where p, q are large prime (say 512 bits long each)
  - $ed = 1 \bmod (p-1)(q-1)$
  - Signing (Private) Key : d
  - Verification (Public) Key : (e, n)
- **Signature Generation:**
  - $S = M^d \bmod n$

  where M is some message
- **Signature Verification:**
  - If $S^e \bmod n = M$, output valid; otherwise, output invalid

Hash Function Motivation

- Consider the RSA Signature Scheme, if $M > n$, how to sign M?
- Solution: instead of signing M directly, Alice signs a hash of M denoted by h(M)
  - Alice sends M and $S = Sign(SK_{Alice}, h(M))$ to Bob
  - Bob verifies that $Verify(PK_{Alice}, h(M), S) = valid$
- h is called a hash function
- h maps a binary string to a non-zero integer smaller than n
- h(M) is called the message digest

Hash Function

- A cryptographic hash function h(x) should provide
  - **Compression** — output length is small and fixed
  - **One-way** — given a value y it is infeasible to find an x such that h(x) = y
  - **collision resistance** — infeasible to find x and y, with x $\neq$ y such that h(x) = h(y)
- Note: As h is a compression algorithm, there should have a lot of collisions. Collision resistance require that it is hard to find any collision

Hash Function Security vs. Hash Output Length

- **If a hash function is collision resistant, then it is also one-way.**

- There is a fixed output length for every collision resistant hash function h.
- To break h against collision resistance using bruteforce attack, the adversary repeatedly chooses random value x, compute h(x) and check if the hash function is equal to any of the hash values of all previously chosen random values.
- If the output of h is N bits long, what is the expected number of times that the adversary needs to try before finding a collision?

Birthday Problem

- How many people must be in a room before probability is $\geq 1/2$ that two or more have same birthday?
  - $1 - 365/365 \cdot 364/365 \cdot \cdot \cdot (365-K+1)/365$
  - Set equal to 1/2 and solve: **K = 23**
- Surprising? A paradox? since we compare all **pairs** x and y
- **K is about sqrt(365)**
- This problem is related to collision resistance.
  - Question: suppose h's output is 80 bits long, how many values must the adversary try before having the probability of compromising collision resistance be at least 1/2?
- **Implication:** secure N bit hash requires $2^{N/2}$ work to "break" (with respect to collision resistance).

Bruteforce Attack Against the Collision-resistance of a Hash Function

- Finding collisions of a hash function using Birthday Paradox.
  1. randomly chooses K messages, m_1, m_2, …, m_k
  2. search if there is a pair of messages, say m_i and m_j such that

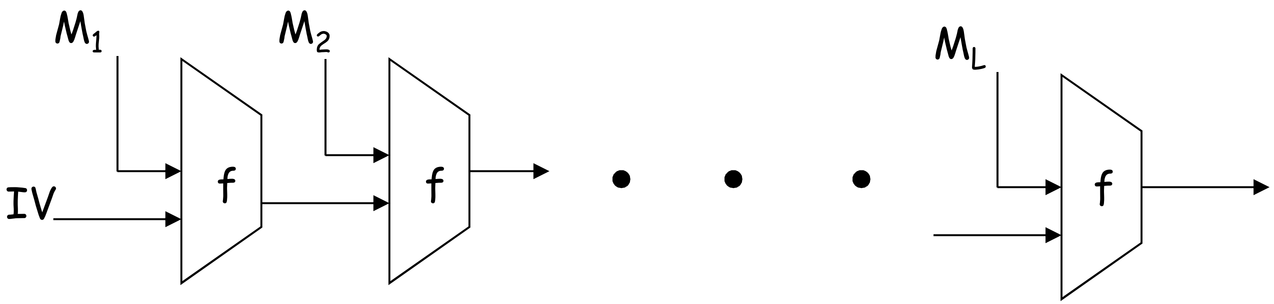
     $h(m_i) = h(m_j)$.

     If so, one collision is found.
- This birthday attack imposes a lower bound on the size of message digests.
- E.g. 10-bit message digest is very insecure, since one collision can be found with probability at least 0.5 after doing slightly over 2^5 (i.e. 32) random hashes.
- E.g. 40-bit message digest is also insecure, since a collision can be found with probability at least 0.5 after doing slightly over 2^{20} (about a million) random hashes.

General Design of Hash Algorithms

- Partition the input message into fixed-sized blocks. (e.g. 512 bits per block)
- The remaining bits of the input are padded with the value of the message length.

- The hash algorithm involves iterated use of a compression function, f.
- It is initialized by an initial value IV (i.e. a magic number).
- Note: Hash algorithms are usually designed heuristically.



Popular Crypto Hashes

- **MD5** — designed by Ronald Rivest
  - 128 bit output
  - Available at http://www.ietf.org/rfc/rfc1321
  - Note: MD5 collisions found (easily)
- **SHA-1** — A US government standard (similar to MD5)
  - 160 bit output
  - Available at http://www.itl.nist.gov/fipspubs/fip180-1.htm
  - Note: A collision found in 2017
- SHA-2 (SHA 256/384/512)
  - Based on SHA-1 with a longer hash value

Security Updates of Hash Functions

MD5

- In Aug 2004, Wang, et al. showed that it is “easy” to find collisions in MD5. They found many collisions in very short time (in minutes)
- http://eprint.iacr.org/2004/199.pdf

SHA-1

- In Feb 2005, Wang et al. showed that collisions can be found in SHA-1 with an estimated effort of 2^{69} hash computations.
  - Less than 2^{80} hash computations by birthday attack.
- http://www.schneier.com/blog/archives/2005/02/sha1_broken.html

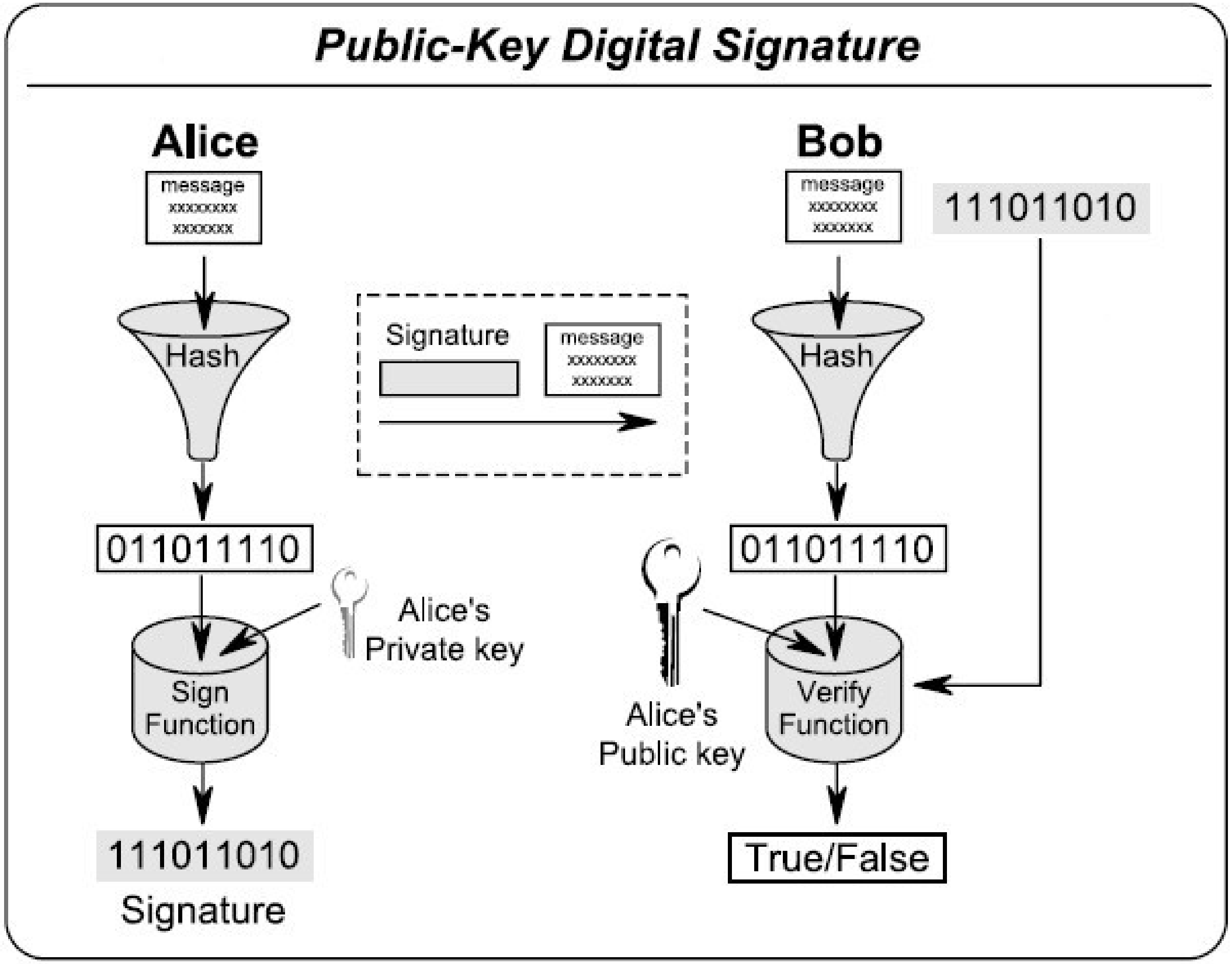
Impacts

- Hurts digital signatures
- For applications require underlying hash functions should be collision resistant, it’s time to migrate away from SHA-1.
- Start using new standards SHA-256 and SHA-512.
- http://csrc.nist.gov/CryptoToolkit/tkhash.html

Block Ciphers as Hash Functions

- Can use block ciphers as hash functions
  - Set $H_0=0$
  - compute: $H_i = AES_{M_i} [H_{i-1}]$
  - and use the final block as the hash value
  - If the length of message is not the multiple of the key size, zero-pad the last segment of message

What are the applications of cryptographic hash functions?


Public-Key Digital Signature
Alice
message
xxxxxxxx
xxxxxxx
Hash
011011110
Sign
Function
Alice's
Private key
111011010
Signature
Signature
message
xxxxxxxx
xxxxxxx
Bob
message
xxxxxxxx
xxxxxxx
111011010
Hash
011011110
Alice's
Public key
Verify
Function
True/False

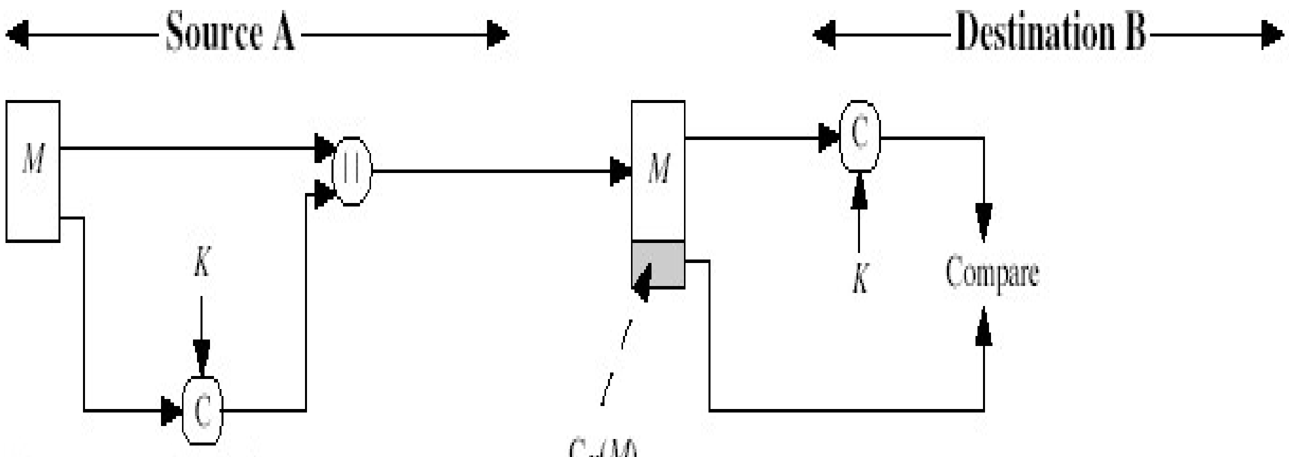
Message Authentication

Message Authentication

- make sure what is sent is what is received
- detect unauthorized modification of data
- Example: Inter-bank fund transfers
  - Confidentiality is nice, but integrity is critical
- Encryption provides confidentiality (prevents unauthorized disclosure)
- Encryption alone does not assure message authentication (a.k.a. data integrity)

MAC

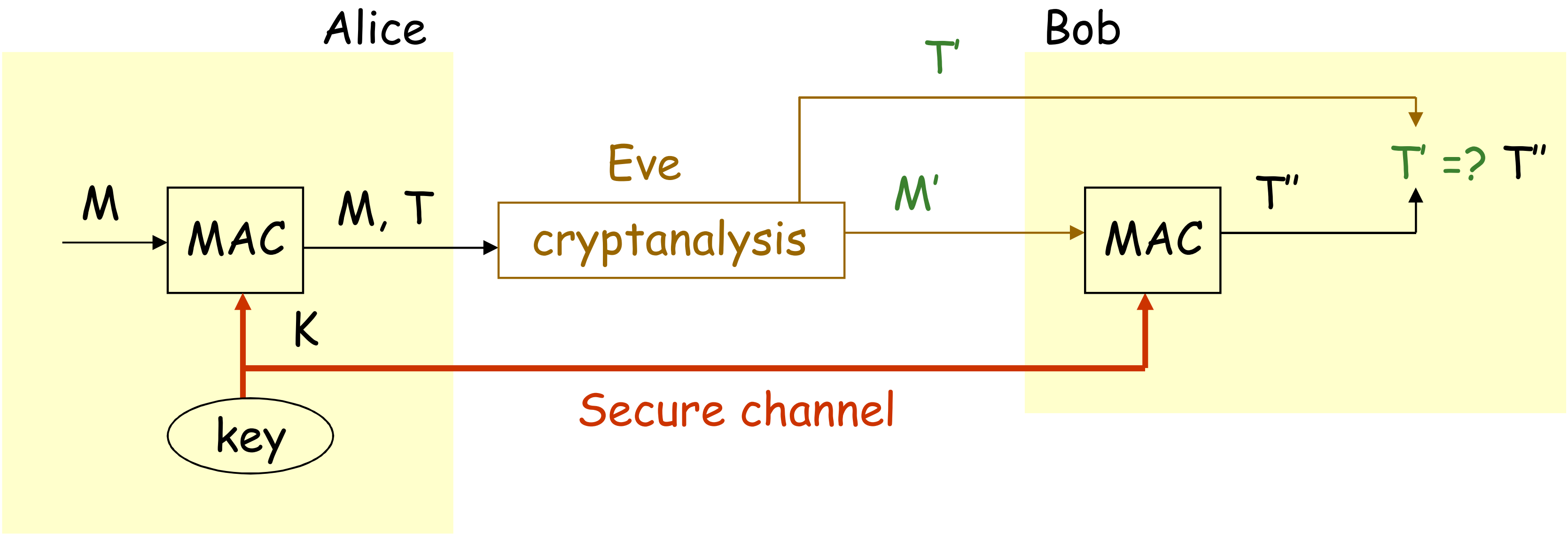
- How MAC Works
  - Sender and receiver share a secret key K
    1. Sender computes a **MAC tag** using the message and K; then sends the MAC tag along with the message
    2. Receiver computes a MAC tag using the message and K; then compares it with the MAC tag received. If they are equal, then the receiver concludes that the message is not changed
  - Note: only sender and receiver can compute and verify a MAC tag



(a) Message authentication

Message Authentication Code

- Message authentication using digital signature
  - Method: Sender signs message using a private key
  - Disadvantage: digital signature is costly
- MAC does not provide non-repudiation
  - Since both sender and receiver share the same symmetric key,
  - Use digital signature for non-repudiation



A MAC Algorithm

- MAC can be constructed from a block cipher operated in CBC mode (with IV=0).
- Suppose a plaintext has 4 plaintext blocks $P=P_0$, P_1, P_2, P_3
- Suppose K is the secret key shared between sender and receiver.

$C_0 = E(K, P_0)$,

$C_1 = E(K, C_0 \oplus P_1)$,

$C_2 = E(K, C_1 \oplus P_2)$,…

$C_{N-1} = E(K, C_{N-2} \oplus P_{N-1})$ = MAC tag

Why does a MAC work?

- Suppose Alice has 4 plaintext blocks
- Alice computes the MAC by doing the following operations:

$C_0 = E(K, P_0)$, $C_1 = E(K, C_0 \oplus P_1)$,

$C_2 = E(K, C_1 \oplus P_2)$, $C_3 = E(K, C_2 \oplus P_3) = \text{MAC tag}$

- Alice sends P_0, P_1, P_2, P_3 and MAC tag to Bob
- Suppose Trudy changes P_1 to X
- Bob computes

$C_0 = E(K, IV \oplus P_0)$, $\mathbf{C_1} = E(K, C_0 \oplus X)$,

$\mathbf{C_2} = E(K, \mathbf{C_1} \oplus P_2)$, $\mathbf{C_3} = E(K, \mathbf{C_2} \oplus P_3) = \textbf{MAC tag} \neq \text{MAC tag}$

- Hence, Trudy can't change **MAC tag** to MAC tag without key K

- **Note: The MAC algorithm above may not be secure if the messages are in variable length.**

The Insecurity of Block Cipher Based MAC Algorithm

- E.g. Given two pairs of (message, MAC tag)
  - (P', T') and (P'', T'') where

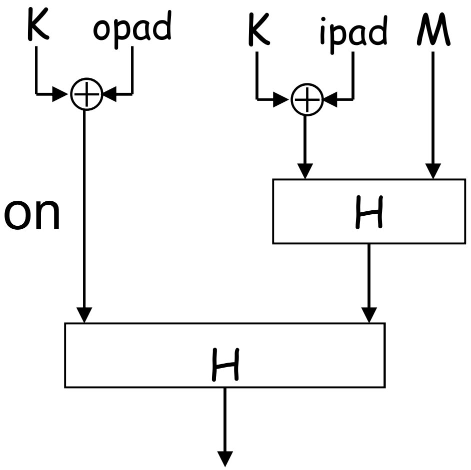
    $P' = P_1, P_2$

    $P'' = P_1$
  - Attack: anyone can forge another pair of message and MAC tag:

    (P''',T''') by setting $P''' = P_2 \oplus T''$ and $T''' = T'$.

Message Authentication - HMAC

- Message Authentication Code: $A \leftarrow C_K(M)$
  - M: message
  - A: authentication tag
  - for integrity and authenticity
- HMAC: Keyed-hashing for Message Authentication
- Used extensively in IPSec (IP Security)
  - IPSec is widely used for establishing Virtual Private Networks (VPNs)



$$HMAC_K(M) = H(\ K \oplus opad \ || \ H((K \oplus ipad) \ || \ M)\ )$$

Let B be the block length of hash, in bytes (B = 64 for MD5 and SHA-1)
ipad = 0x36 repeated B times
opad = 0x5C repeated B times

Summary

- **Signature**
  - RSA Signature
- **Hash**
  - Definitions
  - Find Collusion
- **MAC**
  - Difference from Signature