

## Digital Signatures

Contents:

- What is a digital signature.
- What are digital signatures used for.
- RSA digital signatures.
- Hash functions for digital signatures.

## What is a digital signature

- To simulate physical signatures.
- It uses a public key system. (Note: a public key system is not necessarily used only for encryption and decryption).
- Can only be created by the *right* person who has the private key.
- Signature can be verified by anyone who knows the public key.
- Digital signatures provide **authentication** and **non-repudiation** services.
  - Receiver knows who the originator of the message was.
  - The person who signed a document cannot repudiate it. The signer has liability of the content of a document when a signature is produced.
  - In dispute resolution, a trusted third party (TTP) needs to make a judgment. Evidence is the digital signature and the message.

## How Digital Signatures Work

- Set up: public key system with public key  $PK$  and private key  $SK$ .
- Signing algorithm:  $s := E_{SK}(m)$ , (“encryption” using the private key).
- Signature verification:
  - $m' := E_{PK}(s)$ , (“decryption” using the public key).
  - Check whether  $m' = m$ ? If so, then the signature is accepted as being valid.
- Note:  $E_K(\cdot)$  does not always represent the same algorithm, it varies depending on the key (public or private).

## How Digital Signatures Work

- Note:
  - digital signature is a transformation of the original message.
  - like physical signature, digital signature has to be associated with a particular message, or otherwise it doesn't make sense.
  - When we refer to a digital signature, by default it is assumed that the verifier is given the original message.
  - In real systems, signer has to manage to send the message to the designated verifier.
  - Digital signature does **not** provide message confidentiality!

## Signing and Encrypting

- ▶ As digital signature does not provide confidentiality, how do we sign a secret message?
  - ▶ Sign the message using sender's private key, then encrypt the message (and the signature) using the receiver's public key.
  - ▶ It is not a good idea to first encrypt the message using the receiver's public key, then sign it using the sender's private key. **Why?**
- ▶ Algorithms for digital signatures can be independent of those for message encryption.
  - ▶ Separate systems should be used for encryption and digital signatures.

## RSA Digital Signatures

**Set up:** public key:  $(n, e)$ ; private key:  $d$ .

RSA Encryption and Decryption.

- ▶ Message:  $m < n$ .
- ▶ Encryption:  $c := m^e \bmod n$ .
- ▶ Decryption:  $m := c^d \bmod n$ .

RSA Digital Signature and Verification.

- ▶ Message:  $m < n$ .
- ▶ Signing:  $s := m^d \bmod n$ .
- ▶ Verifying:  $m = s^e \bmod n$ ?

## RSA Digital Signatures

- ▶ In RSA encryption/decryption algorithms,
  - ▶ Encryption is done by someone sending message to the owner of the RSA key pairs.
  - ▶ Decryption is done by the owner of the private key who then can decrypt properly.
- ▶ In RSA signature/verification algorithms,
  - ▶ Digital signature is created by the owner of RSA key pairs.
  - ▶ Verification is done by anyone who is interested and can get the corresponding public key.
- ▶ Combine encryption and digital signature algorithms all together,
  - ▶ only the designated receiver can verify the validity of the signature.

## Limitation on Digital Signatures

Limitations:

- ▶ Since the message can be retrieved from the signature, the size of the digital signature of a message should be as large as or even larger than the message.
- ▶ Both the original message and the signature should be sent to the designated receiver.
- ▶ For a large message, it has to be split into many smaller blocks in order to run the signature algorithm. Signature verification also takes the same number of repeated applications of the verification algorithm.

Solution:

- ▶ One good solution to minimize the computation is to employ a hash function.

## Hash Functions

- ▶ A hash function is a function that takes an arbitrary length of input and outputs a fixed length *digest* (or hash code).
- ▶ A hash function  $H()$  should have the following properties:
  - One-way:** Given any message  $m$ , it is easy to compute  $H(m)$ . However the reverse is computationally infeasible.
  - Collision resistant:** Given any message  $m_1$ , it is computationally infeasible to find  $m_2 \neq m_1$  such that  $H(m_1) = H(m_2)$ .
  - Strong collision resistant:** It is computationally infeasible to find  $m_1 \neq m_2$  such that  $H(m_1) = H(m_2)$ .

## Hash Functions

- ▶ Length of hash code (the digest): acceptable minimum size = 128 (bits).
  - Birthday attack:** In a group of 23 random people, the probability that two of them share the same birthday is larger than 50%.
  - n-bit digest:** with  $2^{n/2}$  trials one can find collision, i.e. one can find  $m_1 \neq m_2$  such that  $H(m_1) = H(m_2)$ .
- ▶ For a hash code of length
  - 60:** with  $2^{30}$  trials one can find a collision.
  - 80:** with  $2^{40}$  trials one can find a collision.
  - 128:**  $2^{64}$  trials is difficult.

## Hash Functions for Digital Signatures

- ▶ Signature algorithms modified.
  - Signing:**  $s := E_{SK}(H(m))$ .
  - Verifying:** (1)  $H(m') = E_{PK}(s)$ , (2)  $m \Rightarrow H(m)$ , (3)  $H(m) = H(m')$ ?
- ▶ The advantages of employing hash function in digital signatures:
  - ▶ As hash code is of fixed length regardless of the length of the original message, signature and verification algorithms can be done only once for any message.
  - ▶ With a good hash function, digital signature is as secure as signing the whole lot of the message.

## A Simple Hash Function

- ▶ Message  $b$  is split into blocks of length  $n$ ,  
 $b = ((b_{11}, b_{12}, \dots, b_{1n}); \dots; (b_{k1}, b_{k2}, \dots, b_{kn}))$ .
- ▶ If the length of message is not multiple of  $n$ , fill up with blank spaces.
- ▶ Hash code is an  $n$ -bit message (looks like a random number).

	Bit 1	Bit 2	...	Bit n
Block 1	$b_{11}$	$b_{12}$	...	$b_{1n}$
Block 2	$b_{21}$	$b_{22}$	...	$b_{2n}$
...	...	...	...	...
Block k	$b_{k1}$	$b_{k2}$	...	$b_{kn}$
<b>Hash code</b>	$C_1$	$C_2$	...	$C_n$

- ▶ Note: this is just an example, real hash functions will not work like this.

## Real Hash Functions

- **[MD4]** Message Digest algorithm version 4 was invented by Rivest (first author of RSA) in 1990 for public criticism.
- **[MD5]** Upgraded to MD5 in 1991, and has been used in some commercial products. Collision found on 17 August, 2004.
- Modified as Secure Hash Standard (SHS) in 1992.
- Hash code length of MD4 and MD5: 128 bits (16 bytes).
- Hash code length of SHS: 160 bits (20 bytes).
- URL for MD5: <http://theory.lcs.mit.edu/~rivest/rfc1321.txt>
- SHA-1 is another popular 160-bit hash function.

## NIST Hash Function Competition

- **[SHA-0, 1 and 2]** Developed by NSA. SHA-1 most widely established and used. Security flaws identified in 2005.
- SHA-2 uses a variable digest size - 224, 256, 384 and 512. No security flaws yet found, but algorithmically similar to SHA-1. Hence, open competition underway to develop a new hash standard.
- Competition held by NIST and the new standard will be called SHA-3.
- Competition announced on November 2, 2007. Submissions were due by October 31, 2008.
- 14 candidates accepted to round 2. List published on July 24, 2009.
- Winner to be announced in 2012.

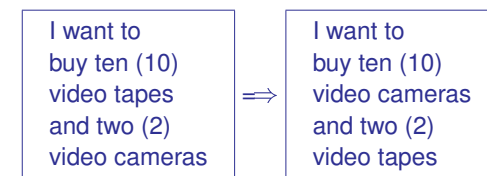
## Real Hash Functions

Why collision resistant?

- If  $\mathcal{B}$  can find  $m' \neq m$  such that  $H(m') = H(m)$ , then by obtaining  $\mathcal{A}$ 's signature on  $m$ ,  $\mathcal{B}$  can claim the signature is on message  $m'$ . WHY?
- If  $\mathcal{B}$  can create something which can pass the signature verification algorithm successfully on some message, then  $\mathcal{B}$  can claim it is someone's signature on the message.
- If one can forge a signature successfully (like the case above), then the signature algorithm is said to be insecure.

## Another Benefit of Using Hash Functions

- It provides information integrity service.
  - Without hash function, signatures may be manipulated due to message split.



- With hash function this cannot happen.

## Review Questions

### Review

- Encryption provides confidentiality services.
- Digital signature provides authentication and non-repudiation services.
- Hash functions provide message integrity services (only when integrated with encryption or digital signature).

### Questions

- What is a hash function?
- What are the features of digital signatures?
- What are the advantages of using hash functions in digital signatures?