# Digital Signature

A.K. Bhateja

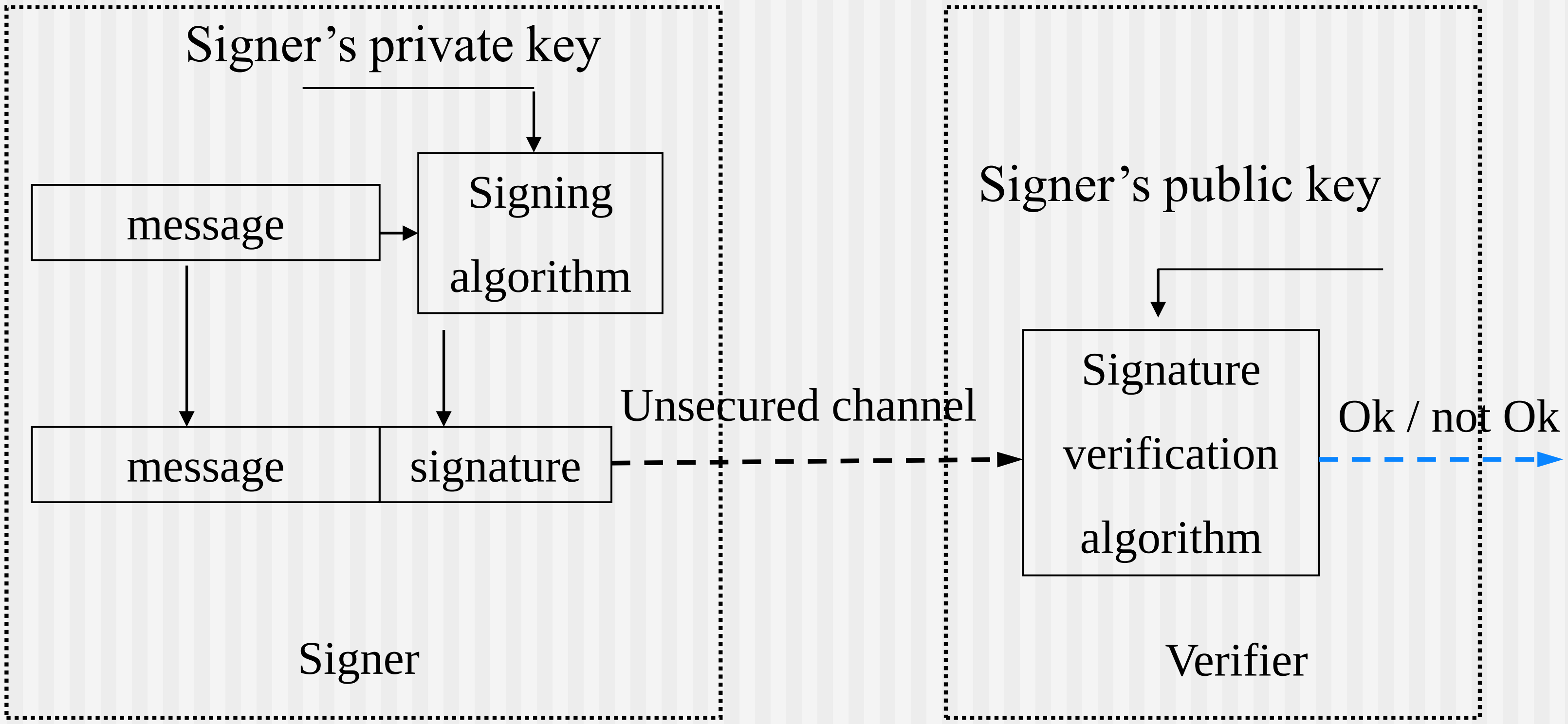
## Digital Signature

- Digital signature is one of the most fundamental and useful inventions of modern cryptography.
- A digital signature is a technique for establishing the origin of a particular message
- Purpose of a digital signature: for an entity to bind its identity to a message.
- Digital signatures cannot be separated from the message and attached to another
- Digital signatures are the public-key primitives of message authentication

## Properties of Digital Signature

- Data Integrity
  - an attacker cannot change the message. Bob receives a message *m* from Alice and wants to convince himself that the message was not modified en-route.
- Message Authentication
  - Assurance that signature has been generated only by sender who own the matching secret private key and no one else.
- Non-repudiation
  - the signer can not later repudiate having signed the message, since no one but the signer possesses his private key.

# Digital Signature Scheme



## RSA signature

- How can user Bob send Alice a signed message *m* in a public-key cryptosystem?
  - He first computes his signature *S* for the message *m* using $d_B$

    $S = \mathrm{D}(m, d_B)$
  - He then encrypts *S* using $e_A$ (for privacy), and sends the result $\mathrm{E}(S, e_A)$ to Alice.
- Alice receives the encrypted message
  - She first decrypts the ciphertext with $d_\mathrm{A}$ to obtain *S*.
  - She the encrypt S using $e_B$ to get *m*

    $m = \mathrm{E}(S, d_A)$

## Message Authentication Code (MAC)

Def: A MAC system $I = (S, V)$ is a pair of efficient algorithms, $S$ (signing algorithm) and $V$ (verification algorithm).

- $S$ is a probabilistic algorithm is used to generate tag $t \leftarrow S(k, m)$, where $k$ is a key, $m$ is a message, and the output $t$ is called a tag.
- $V$ is a deterministic algorithm that is used to verify tag $r \leftarrow V(k, m, t)$, the output $r$ us either accept or reject.
- MAC must satisfy the following correctness property: for all keys $k$ and all messages $m$,

  $P(V(k, m, S(k, m)) = \text{accept}] = 1.$

- Security in MAC means $S$ is unpredictable
  - Given tags $t_1, t_2, \ldots$, of $m_1, m_2, \ldots$ it is hard to predict tag $t$ for any $m \notin \{m_1, m_2, \ldots\}$

## Constructing MACs

- Using Pseudo Random Functions (PRF)
  - A PRF is an algorithm F that takes two inputs, a key *k* and an input data block *x*, and outputs a value *y* = F(*k*, *x*). e.g. AES, stream cipher
  - Any secure PRF is also a secure MAC
  - AES as a PRF gives a MAC for 128-bit message i.e. these MACs are good for short inputs.

## Constructing MACs

- Using Pseudo Random Functions (PRF)
  - A PRF is an algorithm F that takes two inputs, a key *k* and an input data block *x*, and outputs a value $y = F(k, x)$. e.g. AES, stream cipher
  - Any secure PRF is also a secure MAC
  - AES as a PRF gives a MAC for 128-bit message i.e. these MACs are good for short inputs.

## Hash Functions

- A hash function is a mathematical function that converts data of arbitrary length to a fixed length. Generally cryptographic hash functions have the following properties:
  - One way function
  - Compresses arbitrary long inputs into a fixed length output
  - Collision resistance

    if $m_1 \neq m_2$, then $h(m_1) \neq h(m_2)$

    It should be hard to find two different inputs (of any length) that when fed into the hash function result in the same hash (collision free).

Note that it is impossible for a hash function not to have collisions.

Ex.: Cyclic Redundancy Checks (CRC)

## Difference between MAC and Hash

- Hashes are used to guarantee the integrity of data, a MAC guarantees integrity AND authentication both.
- A hashcode is blindly generated from the message without any kind of external input. This code can be used to check if the message got any alteration during its travel.

## Cryptographic Hash Function

- $h: X \to Y, |X| > |Y|$

  $h: \{0, 1\}^* \to \{0, 1\}^n$, $h: \{0, 1\}^{b+n} \to \{0, 1\}^n$
- A classical application of hash function: to store (username, password) as (username, *h*(password)) to have the secrecy of the password.
- Some most commonly used hash functions
  - Secure Hashing Algorithm (SHA-2 and SHA-3)
  - RACE Integrity Primitives Evaluation Message Digest (RIPEMD)
  - Message Digest Algorithm 5 (MD5)
  - BLAKE2

## Efficiency of Operation

- Generally for any hash function $h$ with input $x$, computation of $h(x)$ is a fast operation.
- Computationally hash functions are much faster than a symmetric encryption.

## Security requirements

- Pre-image resistant
  - Given a hash value $y$, it is computationally hard to find a pre-image $x$, s.t. $h(x) = y$.
- Second pre-image registrant
  - Given an input $x$, it is computationally hard to find a second $x'$ such that $h(x) = h(x')$
- Collision resistant
  - Collisions should be “hard" to find

A function $h: \{0, 1\}^* \rightarrow \{0, 1\}^n$ is collision resistant if for every polynomial time algo ‘$A$’, $\exists$ a negligible function negl($n$) s.t. P($A$ successfully produces a collision) $\leq$ negl($n$)

## Probability of collision

- Let us consider a pool of $k$ messages produced randomly and $y$ be a hashcode of a message.
- Suppose the hashcode can take $N = 2^n$ different but equiprobable values.
- Pick a message $x$ at random from the pool of messages.

$$P(h(x) = y) = 1/N$$

$$P(h(x) \neq y) = 1 - 1/N$$

- If two messages $x_1$ and $x_2$ are picked, randomly, from the pool. Probability that none of the two messages has its hashcode equal to $y$ is $(1 - 1/N)^2$ .

## Probability of collision

- Probability that none of the messages in a pool of $k$ messages has its hashcode equal to $y$ is $(1 – 1/N)^k$
- The probability that at least one of the $k$ messages has its hashcode equal to $y$ is

$$1 - (1 – 1/N)^k \approx k/\mathrm{N}$$

## Questions

- How large should $k$ be so that the pool of messages contains at least one message whose hashcode equals the given value $y$ with a probability of 0.5?

$$k/N = 0.5 \Rightarrow k = 0.5\,N$$

- Consider 128 bit hash. $N = 2^{128}$. It is required to construct a pool of $2^{127}$ messages so that there is at least one message whose hashcode is $y$ with probability ½.

## Probability of collision

- Probability that, in a class of 20 students, someone else has the same birthday as yours (assuming you are one of the 20 students) is $\frac{19}{365}$
- Probability that there exists at least one pair of students in a class of 20 students with the same birthday is $\frac{(20\times19)/2}{365} = \frac{190}{365}$
- Birthday attack: given a group of 23 or more randomly chosen people, the probability that at least two of them will have the same birthday is more than 50%.

## Probability of collision

- Given a pool of *k* messages, each of which has a hashcode value from *N* possible such values, the probability that the pool will contain at least one pair of messages with the same hashcode is $1 - \frac{N!}{(N-k)!\,N^k}$

  Total number of ways, to construct a pool of *k* messages with no duplications in hashcode is

  $$N \times (N-1) \times (N-2) \times \cdots \times (N-k+1) = \frac{N!}{(N-k)!}$$

  Total number of ways to construct a pool of *k* messages without worrying about hashcode duplication is

  $$N \times N \times \cdots \times N = N^k$$

  Probability that the pool has no duplications in the hashcodes is

  $$\frac{N!}{(N-k)!\,N^k}$$

# Probability of collision

- Probability of at least one collision is

$$p = 1 - \frac{N!}{(N-k)!\,N^k} = 1 - \frac{N}{N} \times \frac{N-1}{N} \times \cdots \times \frac{N-k+1}{N}$$

$$= 1 - 1 \times \left(1 - \frac{1}{N}\right) \times \left(1 - \frac{2}{N}\right) \ldots \times \left(1 - \frac{k-1}{N}\right)$$

For small $x \geq 0$, $(1 - x) \leq e^{-x}$

$$\therefore \quad p = 1 - \left[e^{-\frac{1}{N}} \times e^{-\frac{2}{N}} \times \ldots \times e^{-\frac{k-1}{N}}\right]$$

$$= 1 - e^{-\frac{k(k-1)}{2N}} \qquad \ldots (1)$$

$$= 1 - \left(1 - \frac{k(k-1)}{2N}\right) \text{ because } e^{-x} \approx 1 - x$$

$$= \frac{k(k-1)}{2N}$$

## No. of messages required for a collision with probability 1/2

- Probability of at least one collision is

$$p = 1 - e^{-\frac{k(k-1)}{2N}} = \frac{1}{2}$$

$$\Rightarrow e^{\frac{k(k-1)}{2N}} = 2 \quad \Rightarrow \frac{k(k-1)}{2N} = \log 2$$

$$\Rightarrow k(k-1) = 2N \log 2$$

$$\Rightarrow k^2 \approx 2N \log 2 \Rightarrow k \approx \sqrt{(2 \log 2)N}$$

$$\text{Or } k \approx 1.177\sqrt{N} \quad \approx \sqrt{N}$$

- i.e. if the hashcode can take a total $N$ different values with equal probability, a pool of $\sqrt{N}$ messages will contain at least one pair of messages with the same hashcode with a probability of 1/2.

- For $n$-bit hash function i.e. $N = 2^n$.
- A pool of $2^{n/2}$ randomly generated messages wil contain at least one pair of messages with the sam hashcode with a probability of 1/2.
- For 128 bit hashcodes, $N = 2^{128}$. So a pool of $2^{6}$ randomly generated messages will have at least one pai with identical hashcodes with a probability of 1/2.
- To resist the collision attack, $N$ should be large enough.
- Currently a minimum of $n \geq 128$ is recommended