

# Hash Functions

## ***Abstract***

*This section introduces the properties and applications of hash functions in cryptography. The main characteristics of the family of Secure Hash Algorithms (SHA) are outlined. The so-called Birthday Paradox and its application to finding hash collisions are explained.*

# Outline

---

- **Hash Functions**
  - The Secure Hash Algorithm (SHA)
  - The Birthday Paradox

# Hash Functions

- A cryptographic hash function

- ◆ takes as input a message of arbitrary length (e.g., a few kB or MB)
- ◆ produces as output a message digest of fixed length (e.g., 160 bit)

- Properties of a hash function  $h(m)$

- ◆ 1. given a message  $m$ , its digest  $h(m)$  can be computed quickly

- ◆ 2.  **$h(m)$  is a one-way (or pre-image resistant) function:**

given a  $y$ , it is computationally infeasible to find any  $m'$  with  $h(m') = y$

- if  $y$  is the hash of some message  $m$ , the aim is not at finding the original  $m$ , but any such  $m'$  with  $h(m') = y$
- any hash function  $y = h(m)$  is *not invertible*, as there are infinite messages  $m$  that yield  $y = h(m)$ , but only some functions are *one-way* (in the sense above)
- example:

$2^{160}$  possible hash values  $y$  of length 160 bit

$2^{81921}$  possible messages  $m$  of length up to 10 kByte

$2^{81761}$  messages  $m$  of length up to 10 kByte for each single  $y$

# Hash Functions

- Further properties of a hash function  $h(m)$ 
  - ♦ 3.  **$h(m)$  is a strongly collision-free function:**  
it is computationally infeasible to find any two different messages  $m_1$  and  $m_2$  with  $h(m_1) = h(m_2)$  ( $m_1 \neq m_2$ )
    - since the set of possible messages  $m$  is infinite, it is expected that there are many pairs of messages  $m_1$  and  $m_2$  with  $h(m_1) = h(m_2)$ , but, for a strongly collision-free function  $h(m)$ , it is practically impossible to find any of them
  - ♦ 4.  **$h(m)$  is a weakly collision-free (a.k.a. second pre-image resistant) function:**  
given  $m$ , it is computationally infeasible to find any other  $m' \neq m$  with  $h(m') = h(m)$ 
    - weakly collision-free resistance is easier to satisfy than strongly resistance
    - strongly collision-free resistance  $\Rightarrow$  weakly collision-free resistance
    - strongly collision-free resistance  $\nRightarrow$  weakly collision-free resistance

# Applications of Hash Functions

---

- Digital signature
  - ◆ the hash of the message is signed, not the message itself
- Message integrity
  - ◆ hash used as an error detection code
- Message integrity and authentication
  - ◆ hash embedding a shared secret (symmetric key): HMAC

# Examples: Some Bad Hash Functions

---

- $y = h(x) = x \bmod n$ 
  - ◆ invertible?
  - ◆ one-way?
  - ◆ collision-free?
- $y = h(x) = \alpha^x \bmod p$ 
  - ◆ invertible?
  - ◆ one-way?
  - ◆ collision-free?

# Examples: Some Bad Hash Functions

---

- $y = h(x) = x^2 \bmod n$ , with  $n = p \cdot q$ 
  - ◆ invertible?
  - ◆ one-way?
  - ◆ collision-free?
- $y = h(x) = \text{DES}_x(\text{"000000...0"})$  (DES encrypt with key  $K=x$ , with  $x < 2^{56}$ )
  - ◆ invertible?
  - ◆ one-way?
  - ◆ collision-free?

# Outline

---

- Hash Functions
- **The Secure Hash Algorithm (SHA)**
- The Birthday Paradox



# The Secure Hash Algorithms (SHA)

- Families of hash functions developed by the NSA and given to NIST
  - ◆ SHA, or SHA-0 (retronym), originally published in 1993 (160 bits)
  - ◆ revised as SHA-1 in 1995 (160 bits)
  - ◆ two families of SHA-2 published in 2001 (256 and 512 bits, with respective truncated versions to 224 and 384 bits)
- SHA-3 selected after a public competition among non-NSA designers in 2012 (same lengths as SHA-2: 224, 256, 384, 512 bits)

	<i>Output Size [bit]</i>	<i>Block Size [bit]</i>	<i>Max Message Size [bit]</i>
SHA-0	160	512	$2^{64}-1$
SHA-1	160	512	$2^{64}-1$
SHA-2 256	256	512	$2^{64}-1$
SHA-2 512	512	1024	$2^{128}-1$
SHA-3 256	256	1088	unlimited
SHA-3 512	512	576	unlimited

# The SHA-1 Algorithm

- The message  $m$  of length  $T$  bits is
  - padded with "1000....0" to make it 64 bits shorter than the next highest multiple of 512 bits
  - padded with 64 bits encoding the message length  $L$
  - divided in  $L$  blocks of fixed length 512 bits  $m = [m_1, m_2, \dots, M_L]$

$$L = \lfloor T/512 \rfloor + 1$$

- The  $L$  blocks  $m_j$  are processed via a sequence of  $L$  subsequent rounds

$$|X_j| = 160 \quad |m_j| = 512$$

$$X_j = h'(X_{j-1}, m_j) \quad j = 1, 2, \dots, L$$

$$X_L = h(m)$$

- The last block  $X_L$  is the final hash (SHA-1)

# The SHA-1 Algorithm Details (1)

1.  $X \wedge Y$  = bitwise "and", which is bitwise multiplication mod 2, or bitwise minimum.
2.  $X \vee Y$  = bitwise "or", which is bitwise maximum.
3.  $X \oplus Y$  = bitwise addition mod 2.
4.  $\neg X$  changes 1s to 0s and 0s to 1s.
5.  $X + Y$  = addition of  $X$  and  $Y$  mod  $2^{32}$ , where  $X$  and  $Y$  are regarded as integers mod  $2^{32}$ .
6.  $X \leftarrow r$  = shift of  $X$  to the left by  $r$  positions (and the beginning wraps around to the end).

We also need the following functions:

$$f_t(B, C, D) = \begin{cases} (B \wedge C) \vee ((\neg B) \wedge D) & \text{if } 0 \leq t \leq 19 \\ B \oplus C \oplus D & \text{if } 20 \leq t \leq 39 \\ (B \wedge C) \vee (B \wedge D) \vee (C \wedge D) & \text{if } 40 \leq t \leq 59 \\ B \oplus C \oplus D & \text{if } 60 \leq t \leq 79 \end{cases}$$

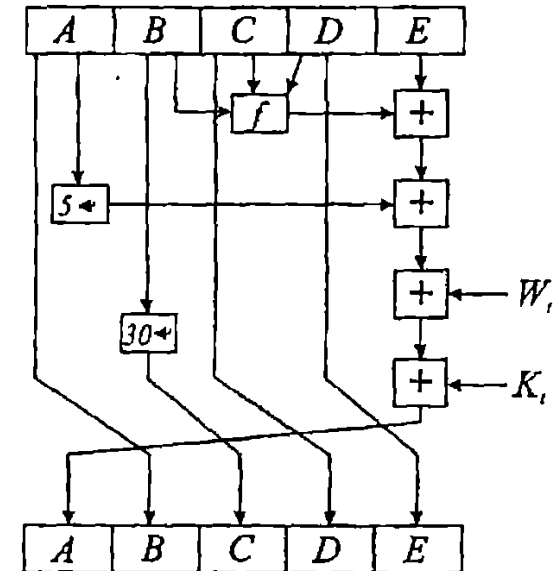
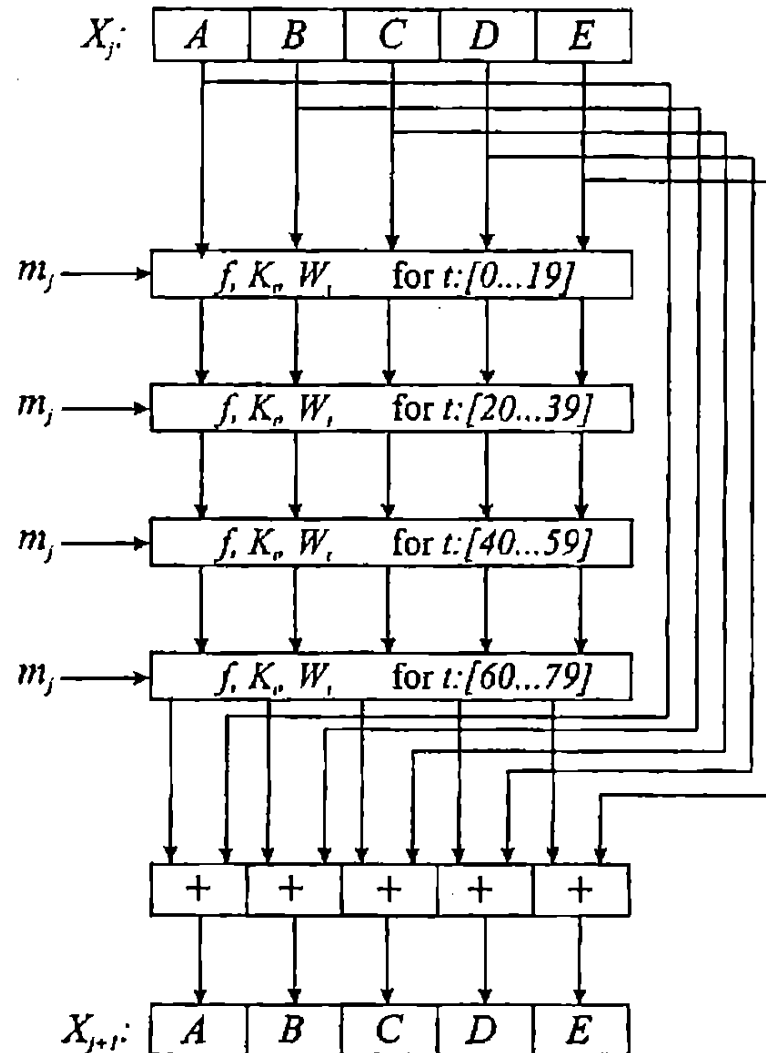
Define constants  $K_0, \dots, K_{79}$  as follows:

$$K_t = \begin{cases} 5A827999 & \text{if } 0 \leq t \leq 19 \\ 6ED9EBA1 & \text{if } 20 \leq t \leq 39 \\ BF1B8CDB & \text{if } 40 \leq t \leq 59 \\ CA62C1D6 & \text{if } 60 \leq t \leq 79 \end{cases}$$

## The SHA-1 Algorithm

1. Start with a message  $m$ . Append bits, as specified in the text, to obtain a message  $y$  of the form  $y = m_1 \| m_2 \| \dots \| m_L$ , where each  $m_i$  has 512 bits.
2. Initialize  $H_0 = 67452301$ ,  $H_1 = EFCDA889$ ,  $H_2 = 98BADCFE$ ,  $H_3 = 10325476$ ,  $H_4 = C3D2E1F0$ .
3. For  $i = 0$  to  $L - 1$ , do the following:
  - (a) Write  $m_i = W_0 \| W_1 \| \dots \| W_{15}$ , where each  $W_j$  has 32 bits.
  - (b) For  $t = 16$  to  $79$ , let  $W_t = (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) \leftarrow 1$
  - (c) Let  $A = H_0$ ,  $B = H_1$ ,  $C = H_2$ ,  $D = H_3$ ,  $E = H_4$ .
  - (d) For  $t = 0$  to  $79$ , do the following steps in succession:  
 $T = (A \leftarrow 5) + f_t(B, C, D) + E + W_t + K_t$ ,  $E = D$ ,  
 $D = C$ ,  $C = (B \leftarrow 30)$ ,  $B = A$ ,  $A = T$ .
  - (e) Let  $H_0 = H_0 + A$ ,  $H_1 = H_1 + B$ ,  $H_2 = H_2 + C$ ,  
 $H_3 = H_3 + D$ ,  $H_4 = H_4 + E$ .
4. Output  $H_0 \| H_1 \| H_2 \| H_3 \| H_4$ . This is the 160-bit hash value.

# The SHA-1 Algorithm Details (2)



# Outline

---

- Hash Functions
- The Secure Hash Algorithm (SHA)
- **The Birthday Paradox**

# The Birthday Paradox

- With  $r = 20$  persons in a room

- the probability that at least 1 has my same birthday ( $N=365$ ) is

$$P = 1 - (364/365)^{20} \cong 0.053$$

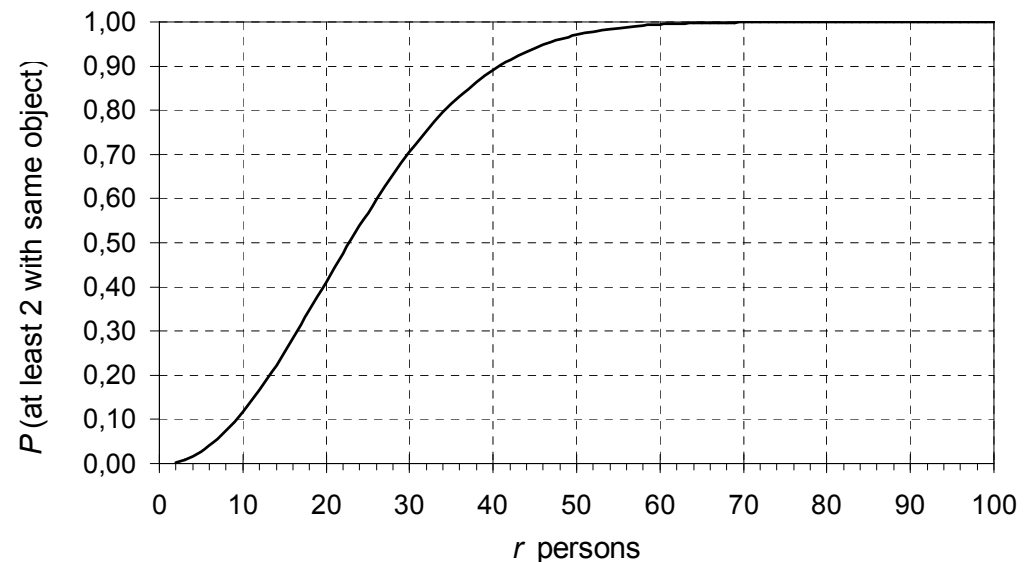
- the probability that at least 2 have the same birthday ( $N=365$ ) is

$$P = 1 - (1 - 1/365) \cdot (1 - 2/365) \cdot \dots \cdot (1 - 19/365) \cong 0.411$$

- Approximation for  $N$  objects picked by  $r$  persons ( $N \gg r$ )

$$P \cong 1 - e^{-r^2/2N}$$

➡  $r \approx \sqrt{N}$  to have  $P \approx 50\%$



- Let us consider two rooms, each with  $r$  persons. What is the probability that at least one in the first room has the same birthday ( $N=365$ ) as at least one in the second room?

- general problem: 2 sets of  $r$  persons pick 1 object among  $N$

$$P \cong 1 - e^{-r^2/N}$$

➡ again:  $r \approx \sqrt{N}$  to have  $P \approx 50\%$

- Birthday attack to digital signature

- two sets of  $r$  documents each, irrelevant variations of two documents (a legit and a bogus one)
- if  $|h|=60$  bit, there are  $N=2^{60}$  possible hashes
- if  $r \approx 2^{30}$ , the probability of having a pair of documents from the two sets with same hash is  $\sim 50\%$

➡ same hash, same digital signature!