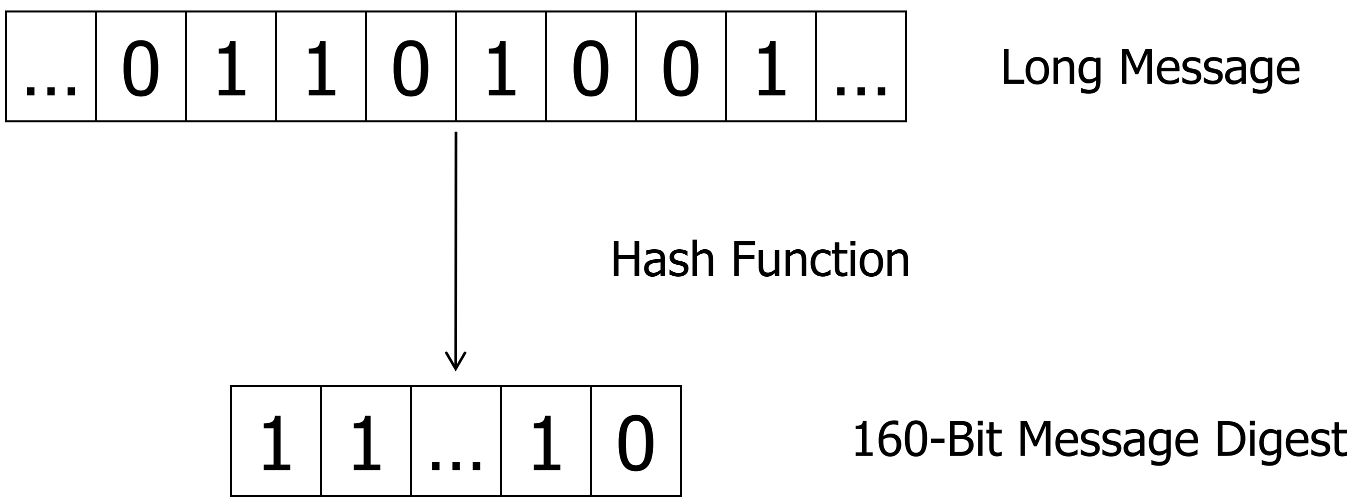
Hash Functions

Hash Functions

A hash function h takes as input a message of arbitrary length and produces as output a message digest of fixed length.



Hash Functions

Certain properties should be satisfied:

1. Given a message m, the message digest h(m) can be calculated very quickly.
2. Given a y, it is computationally infeasible to find an m′ with h(m′)=y (in other words, h is a one-way, or preimage resistant, function).
3. It is computationally infeasible to find messages m_1 and m_2 with $h(m_1) = h(m_2)$ (in this case, the function h is said to be strongly collision-free, or collision resistant).

Hash Functions

Remarks:

- A hash function h is weakly collision-free (or second preimage resistant):
  For given x, it is computational infeasible to find
  $x' \neq x$ with $h(x') = h(x)$

- A hash h is strongly collision-free
  => h is weakly collision-free
  => h is one-way .

Hash Functions

(Example) Discrete log hash function

Let p and $q=(p-1)/2$ be primes. Let α, β be two primitive roots for p. Then, there is a such that

$$\alpha^a \equiv \beta \pmod{p}.$$

The hash h maps integers mod q^2 to integers mod p.

Let $m = x_0+x_1q$ with $0 \le x_0, x_1 \le q-1$.

Define $h(m) = \alpha^{x_0}\beta^{x_1} \pmod{p}$

Hash Functions

The following shows that the function h is probably strongly collision-free.

(Proposition) If we know messages $m \neq m'$ with $h(m)=h(m')$, then we can determine the discrete logarithm $a=\log_\alpha \beta$. (The discrete log problem is assumed hard.)

<Proof> $m = x_0+x_1q$, $m' = x'_0+x'_1q$

$h(m) = h(m') \rightarrow \alpha^{x_0}\beta^{x_1} \equiv \alpha^{x'_0}\beta^{x'_1} \pmod{p}$

$\alpha^{a(x_1-x'_1)-(x'_0-x_0)} \equiv 1 \pmod{p}$

$a(x_1-x'_1) \equiv x'_0-x_0 \pmod{p-1}$

Hash Functions

Let d = gcd(x_1-$x'_{1,}$ p-1). There are exactly d solutions for a. But the only factors of p-1 are 1, 2, q, p-1.

Since $0 \le x_1, x'_1 \le q-1$, it follows that

$-(q-1) \le x_1 - x'_1 \le q-1$. Therefore if x_1-x'_1 is not zero, then d is not q or p-1, so d=1 or 2. Therefore there are at most two possibilities for a.

On the other hand, if if x_1-x'_1 is zero then m=m', contrary to our assumption. #

Simple Hash

- Simple hash
  - Discrete log hash is too slow.
  - Start with a message m of arbitrary length L. We may break it into n-bit blocks.
  - We shall denote these n-bit blocks as $m=[m_1, m_2, m_3, \ldots, m_k]$, and the last block m_k is padded with zeros to ensure that it has n bits.
  - $h(m) = m_1 \oplus m_2 \oplus m_3 \oplus \ldots \oplus m_k$

But it is easy to find two messages that hash to the same value. (so it is not collision resistant)

The Secure Hash Algorithm

- **MD4 proposed by Rivest in 1990**
- **MD5 modified in 1992**
- **SHA proposed as a standard by NIST in 1993, and was adopted as FIPS 180**
- **SHA-1 minor variation, published in 1995 as FIPS 180-1**
- **FIPS 180-2, adopted in 2002, includes SHA1, SHA-256, SHA-384, and SHA-512**
- **A collision for SHA was found by Joux in 2004**
- **Collisions for MD5 and several other popular hash functions were presented in 2004, 2005, by Wang, Feng, Lai and Yu.**

The Secure Hash Algorithm

- SHA-1(Secure Hash Algorithm)
  - iterated hash function
  - 160-bit message digest
  - word-oriented (32 bit) operation on bitstrings
  - Padding scheme extends the input x by at most one extra 512-bit block
  - The compression function maps 160+512 bits to 160 bits
  - Make each input affect as many output bits as possible

The Secure Hash Algorithm

- SHA-1-PAD(x)
  - **comment:** $|x| \leq 2^{64} - 1$
  - $d \leftarrow (447-|x|) \bmod 512$
  - $l \leftarrow$ the binary representation of $|x|$, where $||l|| = 64$
  - $y \leftarrow x \,||\, 1 \,||\, 0^d \,||\, l$ ($|y|$ is multiple of 512)

The Secure Hash Algorithm

- Operations used in SHA-1
  - $X \wedge Y$ bitwise "and" of X and Y
  - $X \vee Y$ bitwise "or" of X and Y
  - $X \oplus Y$ bitwise "xor" of X and Y
  - $\neg X$ bitwise complement of X
  - $X + Y$ integer addition modulo 2^{32}
  - $ROTL^s(X)$ circular left shift of X by s position ($0 \leq s \leq 31$)

In textbook, $X \hookleftarrow s$, instead.

The Secure Hash Algorithm

- $f_t(B,C,D) =$
  - $(B \wedge C) \vee ((\neg B) \wedge D)$ if $0 \leq t \leq 19$
  - $B \oplus C \oplus D$ if $20 \leq t \leq 39$
  - $(B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$ if $40 \leq t \leq 59$
  - $B \oplus C \oplus D$ if $60 \leq t \leq 79$

The Secure Hash Algorithm

- K_t =
  - 5A827999 if $0 \le t \le 19$
  - 6ED9EBA1 if $20 \le t \le 39$
  - 8F1BBCDC if $40 \le t \le 59$
  - CA62C1D6 if $60 \le t \le 79$

The Secure Hash Algorithm

- Algorithm SHA-1(x)
  - **extern** SHA-1-PAD
  - **global** $K_0, \ldots, K_{79}$
  - $y \leftarrow$ SHA-1-PAD(x) denote $y = M_1 \,||\, M_2 \,||..||\, M_n$, where each M_i is a 512 block
  - $H_0 \leftarrow 67452301$, $H_1 \leftarrow EFCDAB89$, $H_2 \leftarrow 98BADCFE$, $H_3 \leftarrow 10325476$, $H_4 \leftarrow C3D2E1F0$

The Secure Hash Algorithm

- **for** $i \leftarrow 1$ **to** n
  - denote $M_i = W_0 \,||\, W_1 \,||..||\, W_{15}$, where each W_i is a word
  - **for** $t \leftarrow 16$ **to** 79

    do $W_t \leftarrow ROTL^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16})$
  - $A \leftarrow H_0$, $,B \leftarrow H_1$, $C \leftarrow H_2$, $D \leftarrow H_3$, $E \leftarrow H_4$
  - **for** $t \leftarrow 0$ **to** 79

    $temp \leftarrow ROTL^5(A) + f_t(B,C,D) + E + W_t + K_t$

    $E \leftarrow D$, $D \leftarrow C$, $C \leftarrow ROTL^{30}(B)$, $B \leftarrow A$, $A \leftarrow temp$
  - $H_0 \leftarrow H_0 + A$, $H_1 \leftarrow H_1 + B$, $H_2 \leftarrow H_2 + C$, $H_3 \leftarrow H_3 + D$, $H_4 \leftarrow H_4 + E$
- **Return** $(H_0 \,||\, H_1 \,||\, H_2 \,||\, H_3 \,||\, H_4)$

Birthday Attacks

- Birthday paradox
  - In a group of 23 randomly chosen people, at least two will share a birthday with probability at least 50%. If there are 30, the probability is around 70%.

  - Finding two people with the same birthday is the same thing as finding a collision for this particular hash function.

Birthday Attacks

- The probability that all 23 people have different birthdays is

$$1 \times (1 - \frac{1}{365})(1 - \frac{2}{365}) \ldots (1 - \frac{22}{365}) = 0.493$$

  Therefore, the probability of at least two having the same birthday is 1- 0.493=0.507

- More generally, suppose we have N objects, where N is large. There are r people, and each chooses an object. Then

$$P(\text{there is a match}) \approx 1 - e^{-r^2/2N}$$

Birthday Attacks

- Choosing $r^2/2N = \ln 2$, we find that if $r \approx 1.177\sqrt{N}$, then the probability is 50% that at least two people choose the same object.

- If there are N possibilities and we have a list of length $\sqrt{N}$, then there is a good chance of a match.
- If we want to increase the chance of a match, we can make a list of length of a constant times $\sqrt{N}$.

Birthday Attacks

(Example) We have 40 license plates, each ending in a 3-digit number. What is the probability that two of the license plates end in the same 3 digits?

(Solution) N=1000, r=40

1. Approximation:

$$1-e^{-40^2/2\cdot 1000}=0.551$$

2. The exact answer:

$$1-(1-\frac{1}{1000})(1-\frac{2}{1000})\dots(1-\frac{39}{1000})=0.546$$

Birthday Attacks

- What is the probability that none of these 40 license plates ends in the same 3 digits as yours?

$$(1-\frac{1}{1000})^{40}=0.961$$

- The reason the birthday paradox works is that we are not just looking for matches between one fixed plate and the other plates. We are looking for matches between any two plates in the set, so there are more opportunities for matches.

Birthday Attacks

- The birthday attack can be used to find collisions for hash functions if the output of the hash function is not sufficiently large.
- Suppose h is an n-bit hash function. Then there are $N = 2^n$ possible outputs. We have the situation of list of length $r \approx \sqrt{N}$ "people" with N possible "birthdays," so there is a good chance of having two values with the same hash value.
- If the hash function outputs 128-bit values, then the lists have length around $2^{64} \approx 10^{19}$, which is too large, both in time and in memory.

Birthday Attacks

- Suppose there are N objects and there are two groups of r people. Each person from each group selects an object. What is the probability that someone from the first group choose the same object as someone from the second group?

$$P(\text{there is a match between two groups})$$

$$= 1 - e^{-r^2/N}$$

- Eg. If we take N=365 and r=30, then

$$P(\text{there is a match between two groups})$$

$$= 1 - e^{-30^2/365} = 0.915$$

Birthday Attacks

- A birthday attack on discrete logarithm
  - We want to solve $\alpha^x \equiv \beta \pmod p$.
  - Make two lists, both of length around $\sqrt{p}$

    1st list: $\alpha^k \pmod p$ for random k.

    2nd list: $\beta\alpha^{-h} \pmod p$ for random h.
  - There is a good chance that there is a match

    $\alpha^k \equiv \beta\alpha^{-h} \pmod p$, hence $x=k+h$.

Compared with BSGS:

BSGS algorithm is deterministic while the birthday attack algorithm is probabilistic.