COMPSCI 4CR3 - Applied Cryptography

Jake Doliskani

McMaster University

Hash Functions

This lecture

- Why we need hash functions
- Important properties of hash functions
- The birthday paradox
- An overview of different families of hash functions
- SHA-2

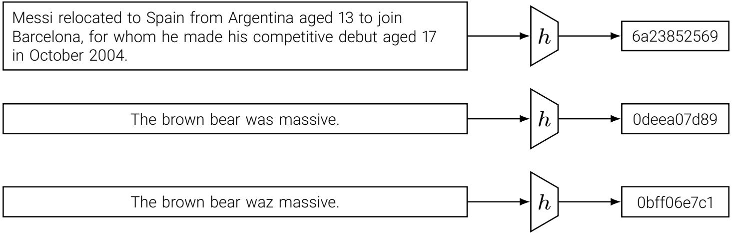
The need for hash functions

We want to be able to represent arbitrarily large data using fixed-size values.

- Extremely important computationally.
  - Used for hash tables to index large strings
  - Used in many other places, e.g., lossy compression, fingerprints, checksums, error-correcting codes, etc.
- Important for security
  - We will see later in the course how hash functions make digital signatures secure.

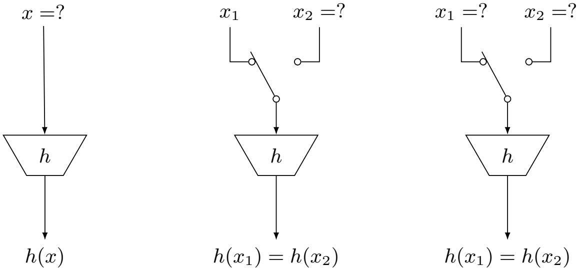
General requirements

1. Should be fast, even for large inputs
2. Should have fixed output size
3. The output should be highly sensitive to all input bits



Cryptographic requirements

1. Preimage resistance (or one-wayness)
2. Second preimage resistance (or weak collision resistance)
3. Collision resistance (or strong collision resistance)



Preimage resistance

Second preimage resistance

Collision resistance

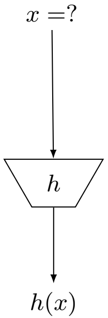
Preimage resistance

A hash function

$$h : \{0,1\}^* \to Y$$

is preimage resistant if given $y \in Y$ in the image of h, it is hard to find $x \in \{0,1\}^*$ such that $h(x) = y$.

Recall: A task T is hard if there is no polynomial time algorithm that can perform T.



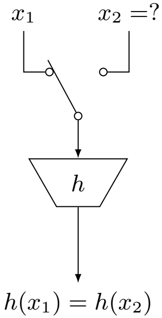
Second preimage resistance

A hash function

$$h : \{0,1\}^* \to Y$$

is second preimage resistant if given $x_1 \in \{0,1\}^*$, it is hard to find $x_2 \in \{0,1\}^*$ such that $x_1 \neq x_2$ and $h(x_1) = h(x_2)$.

Second preimage resistance implies preimage resistance.



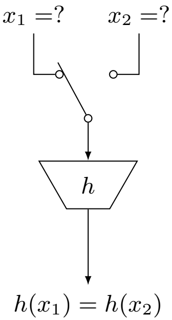
Collision resistance

A hash function

$$h : \{0,1\}^* \to Y$$

is collision resistant if it is hard to find any two $x_1, x_2 \in \{0,1\}^*$ such that $x_1 \neq x_2$ and $h(x_1) = h(x_2)$.

Collision resistance implies second preimage resistance.



Finding collisions

- Suppose the hash function h has an output length of 80 bits.
  - Brute-force: we can try 2^{80} messages to find a collision
  - This can be done by changing spaces, tabs, etc, in actual messages:

```
message1:  Transfer $10 into Trudy's account
message2:  Transfer $10,000 into Trudy,s account
```

- We can do better!
  - The Birthday Attack uses only $\approx 2^{40}$ messages.

The birthday attack

How many people are needed at a party to have a good chance of the same birthday?

$$P(\text{no collision among } t \text{ people}) = \left(1 - \frac{1}{365}\right)\left(1 - \frac{2}{365}\right)\cdots\left(1 - \frac{t-1}{365}\right)$$

For 23 people:

$$\begin{aligned} P(\text{at least one collision}) &= 1 - P(\text{no collision}) \\ &= 1 - \left(1 - \frac{1}{365}\right)\cdots\left(1 - \frac{22}{365}\right) \\ &= 0.507 \end{aligned}$$

The birthday attack

For a hash function with an n-bit output:

$$P(\text{no collision among } t \text{ messages}) = \left(1 - \frac{1}{2^n}\right)\left(1 - \frac{2}{2^n}\right)\cdots\left(1 - \frac{t-1}{2^n}\right) = \prod_{i=1}^{t-1}\left(1 - \frac{i}{2^n}\right)$$

Since $e^{-x} = 1 - x$ for small values of x, we have

$$P(\text{no collision among } t \text{ messages}) \approx \prod_{i=1}^{t-1} e^{-i/2^n} \approx e^{-\frac{t(t-1)}{2^{n+1}}}$$

Set $\lambda = P(\text{at least one collision})$. Then

$$\lambda \approx 1 - e^{-\frac{t(t-1)}{2^{n+1}}}$$

The birthday attack

If we solve for t in

$$\lambda \approx 1 - e^{-\frac{t(t-1)}{2^{n+1}}}$$

we get

$$t \approx 2^{(n+1)/2}\sqrt{-\ln(1-\lambda)}$$

To find a collision (with probability at least 50%) in a hash function with an 80-bit output, we need to try

$$t = 2^{81/2}\sqrt{-\ln(1-0.5)} \approx 2^{40.2}$$

messages.

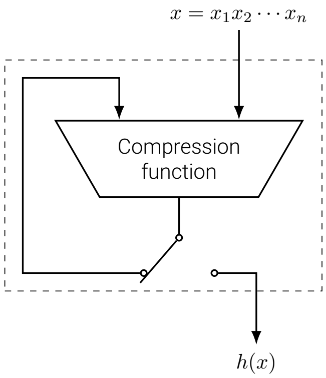
How to build hash functions

- Dedicated hash functions
  - Algorithms that are specifically designed to serve as hash functions
- Hash functions from Block cipher
  - It is also possible to use block ciphers such as AES to construct hash functions

- Remember, hash functions should be able to hash arbitrary-length messages
  - Use the Merkle–Damgård construction

The Merkle–Damgård construction

1. Segment the message into blocks of equal size
   - Do padding if needed
2. Process each block sequentially using a compression functions
3. Return the output of the last iteration



Example: Secure Hash Algorithms 2 (SHA-2)

Dedicated Hash Functions

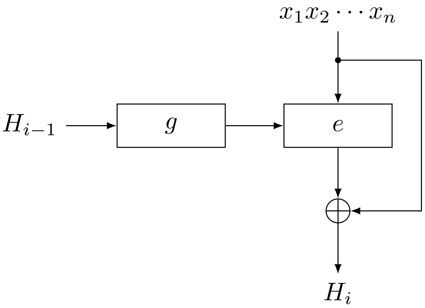
- Hash functions that are custom designed
- Many of these hash functions have been designed in the past 4 decades
- The most popular ones are based on the MD5 family
- MD5 was proposed by Rivest in 1991

- Due to early signs of potential weaknesses in MD5, NIST published SHA
- In 2004, collision-finding attacks against MD5 and SHA-0 where announced by Xiaoyun Wang
- On 23 February 2017, the CWI and Google found collisions in SHA-1

Algorithm	Output	Input	No. of rounds
SHA-224	224	512	64
SHA-256	256	512	64
SHA-384	384	1024	80
SHA-512	512	1024	80

Hash Functions from Block Ciphers

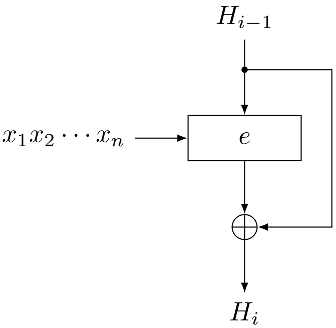
1. Segment the message into blocks of equal size
   - Do padding if needed
2. Feed the blocks into the cipher
3. XOR the output of the cipher with the message block
4. Use the resulting block as the key for the next iteration



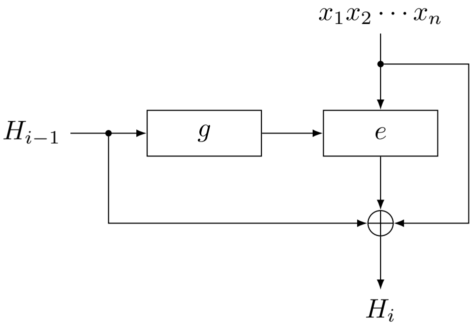
This is called the Matyas–Meyer–Oseas hash function. The equation of the hash function is

$$H_i = e(g(H_{i-1}), x_i) \oplus x_i$$

Other variations



Davies-Meyer



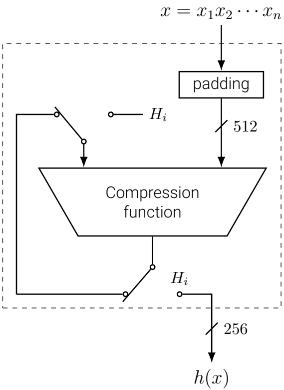
Miyaguchi-Preneel

SHA-2

1. Apply padding so that the message length is a multiple of 512.
2. Divide the message into 512-bit blocks $M_1, M_2, \ldots, M_n$.
3. Sequentially compute

$$H_i = H_{i-1} + C(M_i, H_{i-1})$$

   where C is the is the compression function. The addition is always $\bmod 2^{32}$.
4. Return H_n



SHA-2: constants

$H_0^{(1)}, H_0^{(2)}, \ldots, H_0^{(8)}$

```
6a09e667 bb67ae85 3c6ef372 a54ff53a 510e527f 9b05688c 1f83d9ab 5be0cd19
```

$K_0, K_1, \ldots, K_{63}$

```
428a2f98 71374491 b5c0fbcf e9b5dba5 3956c25b 59f111f1 923f82a4 ab1c5ed5
d807aa98 12835b01 243185be 550c7dc3 72be5d74 80deb1fe 9bdc06a7 c19bf174
e49b69c1 efbe4786 0fc19dc6 240ca1cc 2de92c6f 4a7484aa 5cb0a9dc 76f988da
983e5152 a831c66d b00327c8 bf597fc7 c6e00bf3 d5a79147 06ca6351 14292967
27b70a85 2e1b2138 4d2c6dfc 53380d13 650a7354 766a0abb 81c2c92e 92722c85
a2bfe8a1 a81a664b c24b8b70 c76c51a3 d192e819 d6990624 f40e3585 106aa070
19a4c116 1e376c08 2748774c 34b0bcb5 391c0cb3 4ed8aa4a 5b9cca4f 682e6ff3
748f82ee 78a5636f 84c87814 8cc70208 90befffa a4506ceb bef9a3f7 c67178f2
```

SHA-2: preprocessing

Padding

Suppose the message has length ℓ.

1. Append a "`1`" and k zero bits, where k is the smallest integer such that $\ell + k + 1 \equiv 448 \bmod 512$.
2. Append a 64-bit integer with value equal to ℓ.
3. Example: "`abc`" has length $\ell = 24$. After padding we get

$$\texttt{01100001 01100010 01100011 } \underbrace{\texttt{100...0}}_{423} \; \underbrace{\texttt{00...011000}}_{64}$$

Dividing

Divide the message into 512-bit blocks $M_1, M_2, \ldots, M_n$

- The first 32 bits of M_i is denoted by $M_i^{(0)}$, the second 32 bits by $M_i^{(1)}$ and so on.

SHA-2: functions

$$
\begin{aligned}
\mathrm{Ch}(x,y,z) &= (x \wedge y) \oplus (x \wedge \neg z) \\
\mathrm{Ma}(x,y,z) &= (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \\
\Sigma_0(x) &= (x \gg_R 2) \oplus (x \gg_R 13) \oplus (x \gg_R 22) \\
\Sigma_1(x) &= (x \gg_R 6) \oplus (x \gg_R 11) \oplus (x \gg_R 25) \\
\sigma_0(x) &= (x \gg_R 7) \oplus (x \gg_R 18) \oplus (x \gg_S 3) \\
\sigma_1(x) &= (x \gg_R 17) \oplus (x \gg_R 19) \oplus (x \gg_S 10)
\end{aligned}
$$

- Here, $\gg_R$ is right rotation and $\gg_S$ is right shift.
- The inputs x, y, z are 32-bit words.

SHA-2: message schedule

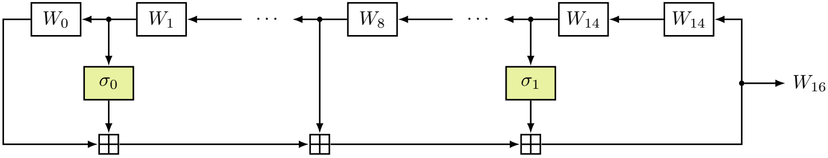
The expanded message blocks $W_0, W_1, ..., W_{63}$ are computed as follows

for $j = 0$ to 15

$$W_j = M_i^{(j)}$$

for $j = 16$ to 63

$$W_j = \sigma_1(W_{j-2}) + W_{j-7} + \sigma_0(W_{j-15}) + W_{j-16}$$

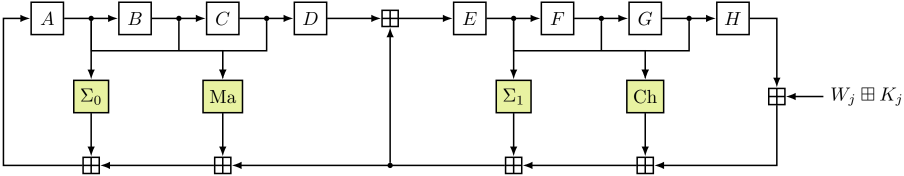


SHA-2: compression function

Input: A, B, C, D, E, F, G, H

for $j = 0$ to 63

$$T_1 = H + \Sigma_1(E) + \mathrm{Ch}(E, F, G) + K_j + W_j$$
$$T_2 = \Sigma_0(A) + \mathrm{Ma}(A, B, C)$$
$$A, B, C, D, E, F, G, H = T_1 + T_2, A, B, C, D + T_1, E, F, G$$



SHA-2

1. Pad and divide to get 512-bit blocks $M_1, M_2, \ldots, M_n$
2. for $i = 1$ to n
   2.1 $A, B, \cdots, H = H_{i-1}^{(0)}, H_{i-1}^{(1)}, \ldots, H_{i-1}^{(7)}$
   2.2 $W_0, W_1 \ldots, W_{63} = \text{schedule}(M_i)$
   2.3 $A, B, \cdots, H = \text{compress}(A, B, C, D, E, F, G, H, W)$
   2.4 $H_i^{(0)}, H_i^{(1)}, \ldots, H_i^{(7)} = H_{i-1}^{(0)} + A, H_{i-1}^{(1)} + B, \ldots, H_{i-1}^{(7)} + H$
3. Return H_n