# Cryptographic Hash Functions

Saravanan Vijayakumaran
sarva@ee.iitb.ac.in

Department of Electrical Engineering
Indian Institute of Technology Bombay

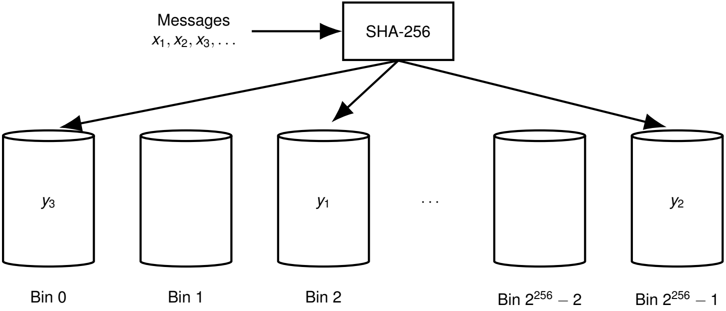
January 7, 2026

# Cryptographic Hash Functions

- Important building block in cryptography
- **Input**: Variable length bitstrings
- **Output**: Fixed length bitstrings
  - For example, output length can be 256 bits
- **Preimage resistant**: Easy to compute but difficult to invert
  - Given $H(x)$, computationally infeasible to find $x$
- **Collision resistant**
  - Computationally infeasible to find $x \neq y$ with $H(x) = H(y)$
- Applications
  - Password hashing
  - Digital signatures on arbitrary length data
  - Commitment schemes

# Cryptographic Hash Functions

- Collisions always exist as domain is larger than co-domain, but are hard to identify
- Having a large co-domain is necessary



# SHA-256

- SHA = Secure Hash Algorithm, 256-bit output length
- Accepts bit strings of length upto $2^{64} - 1$
- Announced in 2001 by NIST, US Department of Commerce
- Output calculation has two stages
  - Preprocessing
  - Hash Computation
- Preprocessing
  1. The input $M$ is padded to a length which is a multiple of 512
  2. A 256-bit state variable $H^{(0)}$ is set to

$$
\begin{aligned}
H_0^{(0)} &= \mathtt{0x6A09E667}, & H_1^{(0)} &= \mathtt{0xBB67AE85}, \\
H_2^{(0)} &= \mathtt{0x3C6EF372}, & H_3^{(0)} &= \mathtt{0xA54FF53A}, \\
H_4^{(0)} &= \mathtt{0x510E527F}, & H_5^{(0)} &= \mathtt{0x9B05688C}, \\
H_6^{(0)} &= \mathtt{0x1F83D9AB}, & H_7^{(0)} &= \mathtt{0x5BE0CD19}.
\end{aligned}
$$

# SHA-256 Input Padding

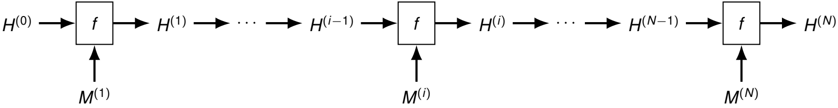
- Let input $M$ be $l$ bits long
  - Find smallest non-negative $k$ such that
$$k + l + 65 = 0 \bmod 512$$
  - Append $k + 1$ bits consisting of single 1 and $k$ zeros
  - Append 64-bit representation of $l$
- Example: $M = 101010$ with $l = 6$
  - $k = 441$
  - 64-bit representation of 6 is $000 \cdots 00110$
  - 512-bit padded message

$$\underbrace{101010}_{M}\ 1\ \underbrace{00000 \cdots 00000}_{441 \text{ zeros}}\ \underbrace{00 \cdots 00110}_{l}.$$

# SHA-256 Hash Computation

1. Padded input is split into $N$ 512-bit blocks $M^{(1)}, M^{(2)}, \ldots, M^{(N)}$
2. Given $H^{(i-1)}$, the next $H^{(i)}$ is calculated using a function $f$

$$H^{(i)} = f(M^{(i)}, H^{(i-1)}), \quad 1 \leq i \leq N.$$




3. $f$ is called a *compression function*
4. $H^{(N)}$ is the output of SHA-256 for input $M$

# SHA-256 Compression Function Building Blocks

- $U$, $V$, $W$ are 32-bit words
- $U \wedge V$, $U \vee V$, $U \oplus V$ denote bitwise AND, OR, XOR
- $U + V$ denotes integer sum modulo $2^{32}$
- $\neg U$ denotes bitwise complement
- For $1 \leq n \leq 32$, the shift right and rotate right operations

$$\text{SHR}^n(U) = \underbrace{000\cdots000}_{n \text{ zeros}} u_0 u_1 \cdots u_{30-n} u_{31-n},$$

$$\text{ROTR}^n(U) = u_{31-n+1} u_{31-n+2} \cdots u_{30} u_{31} u_0 u_1 \cdots u_{30-n} u_{31-n},$$

- Bitwise choice and majority functions

$$\text{Ch}(U, V, W) = (U \wedge V) \oplus (\neg U \wedge W),$$
$$\text{Maj}(U, V, W) = (U \wedge V) \oplus (U \wedge W) \oplus (V \wedge W),$$

- Let

$$\Sigma_0(U) = \text{ROTR}^2(U) \oplus \text{ROTR}^{13}(U) \oplus \text{ROTR}^{22}(U)$$
$$\Sigma_1(U) = \text{ROTR}^6(U) \oplus \text{ROTR}^{11}(U) \oplus \text{ROTR}^{25}(U)$$
$$\sigma_0(U) = \text{ROTR}^7(U) \oplus \text{ROTR}^{18}(U) \oplus \text{SHR}^3(U)$$
$$\sigma_1(U) = \text{ROTR}^{17}(U) \oplus \text{ROTR}^{19}(U) \oplus \text{SHR}^{10}(U)$$

# SHA-256 Compression Function Calculation

- Maintains internal state of 64 32-bit words $\{W_j \mid j = 0, 1, \ldots, 63\}$
- Also uses 64 constant 32-bit words $K_0, K_1, \ldots, K_{63}$ derived from the first 64 prime numbers $2, 3, 5, \ldots, 307, 311$
- $f(M^{(i)}, H^{(i-1)})$ proceeds as follows
  1. Internal state initialization
  $$W_j = \begin{cases} M_j^{(i)} & 0 \le j \le 15, \\ \sigma_1(W_{j-2}) + W_{j-7} + \sigma_0(W_{j-15}) + W_{j-16} & 16 \le j \le 63. \end{cases}$$
  2. Initialize eight 32-bit words
  $$(A, B, C, D, E, F, G, H) = \left(H_0^{(i-1)}, H_1^{(i-1)}, \ldots, H_6^{(i-1)}, H_7^{(i-1)}\right).$$
  3. For $j = 0, 1, \ldots, 63$, iteratively update $A, B, \ldots, H$
  $$T_1 = H + \Sigma_1(E) + \mathrm{Ch}(E, F, G) + K_j + W_j$$
  $$T_2 = \Sigma_0(A) + \mathrm{Maj}(A, B, C)$$
  $$(A, B, C, D, E, F, G, H) = (T_1 + T_2, A, B, C, D + T_1, E, F, G)$$
  4. Calculate $H^{(i)}$ from $H^{(i-1)}$
  $$(H_0^{(i)}, H_1^{(i)}, \ldots, H_7^{(i)}) = \left(A + H_0^{(i-1)}, B + H_1^{(i-1)}, \ldots, H + H_7^{(i-1)}\right).$$

# Birthday Attacks for Finding Collisions

- Birthday Problem: Given $Q$ people, what is the probability of two of them having the same birthday?
- Suppose the size of $\mathcal{Y}$ is $M$. For SHA-256, $M = 2^{256}$.
- If we calculate $H$ for $Q$ inputs, the probability of a collision is

$$1 - \left(1 - \frac{1}{M}\right)\left(1 - \frac{2}{M}\right)\cdots\left(1 - \frac{Q-1}{M}\right) \approx 1 - \exp\frac{-Q(Q-1)}{2M}$$

- For success probability $\varepsilon$, the number of “queries” is

$$Q \approx \sqrt{2M \ln \frac{1}{1-\varepsilon}}$$

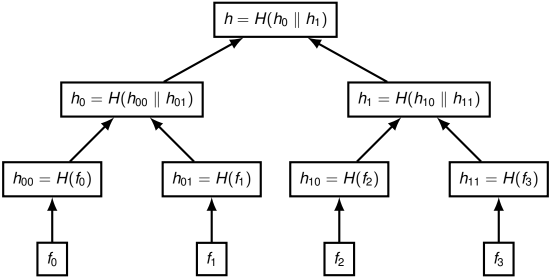
- For $\varepsilon = 0.5$, $Q \approx 1.17\sqrt{M}$
- For SHA-256, $Q \approx 2^{128}$

# Applications of Hash Functions

- Password hashing
- Digital signatures on arbitrary length data
- Virus fingerprinting
- Commitment schemes
  - A kind of digital envelope
  - Allows one party to “commit” to a message $m$ by sending a commitment $c$ to the counterparty
  - Set $c = H(m \| r)$ where $r$ is a random $n$-bit string
  - **Hiding**: $c$ reveals nothing about $m$
  - **Binding**: Infeasible for $c$ to be opened to a different message $m'$

# Merkle Trees

- Suppose a client uploads multiple files to server
- Client wants to ensure file integrity at a later retrieval




- For $N$ files, $\mathcal{O}(\log N)$ communication from server ensures integrity
- The communication is called a *Merkle proof*

# References


- Chapter 6 of *Introduction to Modern Cryptography*, J. Katz, Y. Lindell, 3rd edition
- Chapter 3 of *An Introduction to Bitcoin*, S. Vijayakumaran, `www.ee.iitb.ac.in/~sarva/bitcoin.html`