Cryptographic Hash Functions

Saravanan Vijayakumaran
sarva@ee.iitb.ac.in

Department of Electrical Engineering
Indian Institute of Technology Bombay

July 17, 2018

Cryptographic Hash Functions

- Important building block in cryptography
- Provide data integrity by construction of a short fingerprint or *message digest*
- Map arbitrary length inputs to fixed length outputs
  - For example, output length can be 256 bits
- Applications
  - Password hashing
  - Digital signatures on arbitrary length data
  - Commitment schemes

Properties

- Let $H : \mathcal{X} \mapsto \mathcal{Y}$ denote a cryptographic hash function
- $H(x)$ can be computed efficiently for all $x \in \mathcal{X}$
- If H is considered secure, three problems are difficult to solve
  - Preimage
    - Given $y \in \mathcal{Y}$, find $x \in \mathcal{X}$ such that $H(x) = y$
  - Second Preimage
    - Given $x \in \mathcal{X}$, find $x' \in \mathcal{X}$ such that $x' \neq x$ and $H(x) = H(x')$
  - Collision
    - Find $x, x' \in \mathcal{X}$ such that $x' \neq x$ and $H(x) = H(x')$
- If $|\mathcal{X}| \geq 2|\mathcal{Y}|$, then we have

  Collision resistance $\implies$ Second preimage resistance $\implies$ Preimage resistance

  (Proof in Section 4.2, Stinson, 3rd edition)

SHA-256

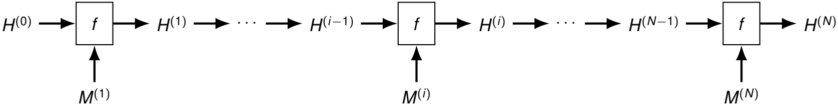
- SHA = Secure Hash Algorithm, 256-bit output length
- Accepts bit strings of length upto $2^{64} - 1$
- Announced in 2001 by NIST, US Department of Commerce
- Output calculation has two stages
  - Preprocessing
  - Hash Computation
- Preprocessing
  1. The input M is padded to a length which is a multiple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

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

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

- Bitwise choice and majority functions

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

- Let

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

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 $A, B, \ldots, H$
  $$\begin{aligned} T_1 &= H + \Sigma_1(E) + \mathrm{Ch}(E, F, G) + K_j + W_j \\ T_2 &= \Sigma_0(A) + \mathrm{Maj}(A, B, C) \\ (A, B, C, D, E, F, G, H) &= (T_1 + T_2, A, B, C, D + T_1, E, F, G) \end{aligned}$$
  4. Calculate $H^{(i)}$ from $H^{(i-1)}$
  $$(H_0^{(i)}, H_1^{(i)}, \ldots, H_7^{(i)}) = \left(A + H_0^{(i-1)}, B + H_1^{(i-1)}, \ldots, H + H_7^{(i-1)}\right).$$

References

- Chapter 5 of *Introduction to Modern Cryptography*, J. Katz, Y. Lindell, 2nd edition
- Chapter 4 of *Cryptography: Theory and Practice*, Douglas R. Stinson, 3rd edition
- Chapter 8 of *A Graduate Course in Applied Cryptography*, D. Boneh, V. Shoup, `www.cryptobook.us`
- Chapter 3 of *An Introduction to Bitcoin*, S. Vijayakumaran, `www.ee.iitb.ac.in/~sarva/bitcoin.html`