[CS304] Introduction to Cryptography and Network Security

Course Instructor: Dr. Dibyendu Roy — Winter 2023-2024
Scribed by: Manas Jitendrakumar Ingle (202151086) — [Week 7 & 8]

Week 7&8

1 Second Pre-image:

$$\begin{aligned}
&h : A \to B \quad |B| = M \\
&x_0 \subseteq A \quad h(x) \\
&|X_0| = Q \quad x, h(x) \\
&h(x_i) \quad x_i \in X_0 \quad x \neq x' \text{ such that } h(x') = h(x) \\
&X_0 \subseteq A \setminus \{x\}
\end{aligned}$$

Note: Complexity here is same as that of pre-image: $O(M)$

2 Collision Finding Algorithm:

Theorem 1 *For any $X_0 \subseteq X$ with $|X_0| = Q$, the success probability of Collision Finding is*

$$\epsilon = 1 - \left(\frac{M-1}{M}\right)\left(\frac{M-2}{M}\right)\cdots\left(\frac{M-Q+1}{M}\right).$$

Let $X_0 = \{x_1, \ldots, x_Q\}$. For $1 \leq i \leq Q$, let E_i denote the event

$$\text{"}h(x_i) \notin \{h(x_1), \ldots, h(x_{i-1})\}.\text{"}$$

We observe trivially that $\Pr[E_1] = 1$. Using induction, it follows from Finding the preimage that

$$\Pr[E_i | E_1 \wedge E_2 \wedge \cdots \wedge E_{i-1}] = \frac{M-i+1}{M},$$

for $2 \leq i \leq Q$. Therefore, we have that

$$\Pr[E_1 \wedge E_2 \wedge \cdots \wedge E_Q] = \left(\frac{M-1}{M}\right)\left(\frac{M-2}{M}\right)\cdots\left(\frac{M-Q+1}{M}\right).$$

The probability that there is at least one collision is $1 - \Pr[E_1 \wedge E_2 \wedge \cdots \wedge E_Q]$, so the desired result follows. The above theorem shows that the probability of finding no collisions is

$$\left(1 - \frac{1}{M}\right)\left(1 - \frac{2}{M}\right)\cdots\left(1 - \frac{Q-1}{M}\right) = \prod_{i=1}^{Q-1}\left(1 - \frac{i}{M}\right).$$

If x is a small real number, then $1 - x \approx e^{-x}$. This estimate is derived by taking the first two terms of the series expansion

$$e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \cdots$$

Using this estimate, the probability of finding no collisions is approximately

$$\prod_{i=1}^{Q-1}\left(1-\frac{i}{M}\right) \approx \prod_{i=1}^{Q-1} e^{-\frac{i}{M}}$$

$$= e^{-\sum_{i=1}^{Q-1}\frac{i}{M}}$$

$$= e^{-\frac{Q(Q-1)}{2M}}.$$

Consequently, we can estimate the probability of finding at least one collision to be

$$1 - e^{-\frac{Q(Q-1)}{2M}}.$$

If we denote this probability by ϵ, then we can solve for Q as a function of M and ϵ:

$$e^{-\frac{Q(Q-1)}{2M}} \approx 1-\epsilon$$

$$-\frac{Q(Q-1)}{2M} \approx \ln(1-\epsilon)$$

$$Q^2 - Q \approx 2M \ln \frac{1}{1-\epsilon}.$$

If we ignore the term $-Q$, then we estimate that

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

If we take $\epsilon = .5$, then our estimate is

$$Q \approx 1.17\sqrt{M}.$$

So this says that hashing just over $\sqrt{M}$ random elements of X yields a collision with a probability of 50%. Note that a different choice of ϵ leads to a different constant factor, but Q will still be proportional to $\sqrt{M}$.
The algorithm has a complexity of $O(\sqrt{M})$.

3 Compression Function:

Construct $H : \{0,1\}^* \longrightarrow \{0,1\}^m$ from h . Security of H will completely depend on security of h.
Given $x \epsilon \{0,1\}^*$
$|x| : length\ of\ x$
$|x| \geq m+t+1$

From x constuct y by using a public function s.t
$|y| \equiv 0 (mod t)$

$$y = \begin{Bmatrix} x & if & |x| \equiv O(mod t) \\ x||O^d & if & |x|+d \equiv O(mod t) \end{Bmatrix}$$

Select $IV \epsilon \{0,1\}^m$ $(IV\ is\ public)$

$y = y_1||y_2||...||y_r\ s.t\ |y_i| = t, 1 \leq i \leq r$

$$H \begin{cases} Z_0 & = & IV \\ Z_1 & = & h(Z_0||y_1) \\ Z_1 & = & h(Z_1||y_2) \\ . \\ . \\ . \\ Z_r & = & h(Z_r - 1||y_r) \end{cases}$$

$Z_r = H(x)$. This is known as iterative hash function.

Given (M, Z_r) is it possible to find (M_2, Z) without computing H on (M_2, Z).

4 Length Extension Attack

Let x' be any bitstring of length t, and consider the message $x\|x'$. The tag for this message, $h_K(x\|x')$, is computed to be $h_K(x\|x') = \text{compress}(h_K(x)\|x')$. Since $h_K(x)$ and x' are both known, it is a simple matter for an adversary to compute $h_K(x\|x')$, even though K is secret. This is called a length extension attack.

5 Merkle-Damgard

Suppose *compress* $: 0,1^{m+t} \to 0,1^m$ is a collision resistant compression function, where $t \geq 1$. So *compress* takes $m + t$ input bits and produces m output bits. We will use *compress* to construct a collision resistant hash function $h : X \to 0,1^m$, where $X = 0,1^i$ for $i \geq m + t + 1$. Thus, the hash function h takes any finite bitstring of length at least $m + t + 1$ and creates a message digest that is a bitstring of length m. We first consider the situation where $t \geq 2$ (the case $t = 1$ will be handled a bit later).

We will treat elements of $x \in X$ as bit strings. Suppose $|x| = n \geq m + t + 1$. We can express x as the concatenation

$$x = x_1\|x_2\| \cdots \|x_k\|x_{k+1}$$

where $0 \leq d \leq t - 2$. Hence, we have that

$$n = |x| = |x_1| + |x_2| + \cdots + |x_k| + |x_{k+1}| = mk + d + 1.$$

$$y(x) = y_1\|y_2\| \cdots \|y_{k+1}$$

6 Secure Hash Algorithm(SHA):

SHA - 160
SHA - 256
SHA - 512
$\Longrightarrow$ SHA-I PAD(x)

1. $|x| \lneq 2^{64} - 1$
2. $d = (447 - |x|)\text{mod}512$
3. $l = \text{binary}(|x|)$
4. $y = x||1||O^d||l$
5. $|x| + d = 447(\text{mod}512)$

$X \wedge Y$: bitwise and
$X \vee Y$: bitwise OR
$X \oplus Y$: bitwise XOR
$X \rceil Y$: bitwise comple
$X + Y$:addition mod 32
ROTL(x): circular shift on x by S position

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

6.1 SHA-I (x):

$Y = SHA - I - PAD(x)$

$y = M_1||M_2|| \dots ||M_n$ $\qquad |M_L| = 512$

$H_0 = 67452301$
$H_1 = EFCDAB89$
$H_2 = 98BADCFE$
$H_3 = 10325476$
$H_4 = C3D2E1F0$

$$K_t = \begin{cases} A827999 & 0 \leq t \leq 19 \\ 6ED9EBA1 & 20 \leq t \leq 39 \\ 8F1BBCDC & 40 \leq t \leq 59 \\ CA62C1D6 & 60 \leq t \leq 79 \end{cases}$$

6.1.1 Algorithm:

for i = 0 to N
$M_i = w_0||w_1|| \dots w_{15}$ $\qquad |w_i| - = 32$ bit

for t = 16 to 79
$w_t = ROTL'(w_{t-3} \oplus wt - 8 \oplus w_{t-14} \oplus w_{t-16})$

$A = H_0$ $\qquad C = H_2$ $\qquad E = H_4$
$B = H_1$ $\qquad D = H_3$

for t = 0 to 79
temp = $ROTL^5(A) + f_t(B, C, D) + E + w_t + k_t$

$E = D$ $\quad$ $D = C$ $\quad$ $C = ROTL^{30}(B)$
$B = A$ $\quad$ $A =$temp

$H_0 = H_0 + A$
$H_1 = H_1 + B$
$H_2 = H_2 + C$
$H_3 = H_3 + D$
$H_4 = H_4 + E$

return $(H_0||H_1||H_2||H_3||H_4)\}$160 bits

6.1.2 Message Authentication Code(MAC):

Alice = k and Bob = k

$C = \text{Enc}(M, K) \longrightarrow \tilde{c}$

$\text{MAC} = \text{Hash}(M, K) \xrightarrow{MAC} M\tilde{A}C$

$Dec(\tilde{C}, K) = \tilde{M}$
$\text{Hash}(\tilde{M}, K) = \text{MAC}_1$
if $\text{MAC}_1 = M\tilde{A}C$
accept $\tilde{M}$
as $\tilde{M} \cong M$

6.1.3 HMAC:

i pad $= 3636\ldots36 \rightarrow 512$ bits
o pad $= 5C5C\ldots5C \rightarrow 512$ bits
K $\rightarrow$ secret key

$HMAC_k(x) = H[(k\oplus \text{ o pad})||H((k\oplus \text{ i pad) }||x)]$

6.1.4 CBC - MAC (x,k):

$x = x_1||\ldots||x_n$ $\quad$ $|x_i| =$ bits of AES
$IV = 00\ldots0$

$y_0 = IV$
for $i = 1$ to N

$y_i = Enc((y_{i-1} \oplus x_i), k)$
return (y_n)

6.1.5 SHA - 256 bits:

message size: $< 2^{64}$ bits
block size: 512
word size: 32

6.2 Functions used in SHA - 256:

$\rightarrow ROTR^N(x) = (x >> n) \vee (x << w - n)$
$\rightarrow SHR^n(x) = x >> n$
$\rightarrow Cn(x, y, z) = (x \wedge y) \oplus (\rceil x \wedge z)$
if k = 1 we get y
x = 0 we get z
$\rightarrow$ Maj$(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$

6.3 Sigma Functions:

$\sum_0^{256}(x) = ROTR^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x)$

$\sum_1^{256}(x) = ROTR^6(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x)$

$\sigma_0^{256}(x) = SHR^7(x) \oplus SHRR^{18}(x) \oplus SHR^3(x)$

$\sigma_1^{256}(x) = SHR^{17}(x) \oplus SHRR^{19}(x) \oplus SHR^{10}(x)$
$\rightarrow$ 64 - constants of 32 bits.

$\rightarrow$ Cube roots of prime number. Find the decimal part.
First 32-bits of fractional part of the cube roots of first 64 prime number.

$\rightarrow$ 8 H constants of 32 - bits.
Take first 32 - bits of fractional part of $\surd$ of first 8 prime numbers.

$\Rightarrow l + 1 + k = 448$ mod 512
Message $\rightarrow$ 512 bit divide into 32 bit size block ,i.e, 16 such block m_i.

6.3.1 Algorithm:

SHA - 256

for i = 1 to N
$$w_t = \begin{cases} M_t^{\{i\}} & 0 \leq t \leq 15 \\ \sigma_1^{256} w_{t-2} + w_{t-7} + \sigma_1^{256} w_{t-15} + w_{t-16} & 16 \leq t \leq 63 \end{cases}$$
$a = H_0^{i-1}$

for t = 0 to 63

$T_1 = h + \sum_0^{256}(e) + ch(e, f, g) + k_t^{256} + w_t$

$$T_2 = \sum_0^{256}(a) + \mathrm{Maj}(a, b, c)$$

$$h = g \qquad g = f \qquad f = e$$
$$e = a + T_1 \qquad d = c \qquad c = b \qquad b = a \qquad a = T_1 + T_2$$

$H^(i)_0 = a + H_0^{i-1}$
$H_1^i = b + H^{i-1}1$
$\ldots$
$H_7^i = h + H_7^{i-1}$

Finally
$\underbrace{H_0^N||H_1^N||\ldots||H_7^N}_{256-bitmessage}$