Prof. Dr. Rudolf Mathar, Dr. Michael Reyer, Jose Leon, Qinwei He

Exercise 4
- Proposed Solution -

Friday, November 17, 2017

Solution of Problem 1

a) Having the following expression:

$$h: \{0,1\}^* \to \{0,1\}^*,\ k \mapsto \left(\left\lfloor 10000\left((k)_{10}(1+\sqrt{5})/2 - \left\lfloor (k)_{10}(1+\sqrt{5})/2 \right\rfloor\right)\right\rfloor\right)_2.$$

We want to obtain the upper bound in terms of bit length. Therefore, we will analyze the expression:

$$\alpha = \left((k)_{10}(1+\sqrt{5})/2 - \left\lfloor (k)_{10}(1+\sqrt{5})/2 \right\rfloor\right) < 1$$

but it can be arbitrary close to 1

Hence now the expression is simpler and we can obtain the upper bound:

$$10000\ (\alpha) < 10000 \leq 9999$$

Now applying the logarithm, we obtain the bit length:

$$log_2(9999) \approx 13.288 \leq 14$$

b) We search for a collision:

$$k = 1 \longrightarrow (1+\sqrt{5})/2 = 1.6180$$
$$\longrightarrow (k)_{10}(1+\sqrt{5})/2 - \left\lfloor (k)_{10}(1+\sqrt{5})/2 \right\rfloor = 0.6180$$

Therefore, we need to search for a value x, s.t:

$$x(1+\sqrt{5})/2 = a + 0.6180 + b$$

with a$\in \mathbb{Z}$, b < 0.0001
We create a while loop to obtain the value for the collision:
$x = 2$

while $(0.618 > x((1+\sqrt{5})/2) - \left\lfloor x(1+\sqrt{5})/2 \right\rfloor > 0.618 + 0.0001)$ **do**
  $x = x + 1$
end while

Obtaining a value of $k = 10947$, where

$$(h(1))_{10} = 6180$$
$$(h(10947))_{10} = 6180$$

since the values are equal, we obtain a collision.

Solution of Problem 2

Given: two hash functions with output length of 64 bits and 128 bits.

a) How many messages have to be created, such that the probability of a collision exceeds 0.86?

Birthday paradox: k objects, n bins, $p_{k,n}$, the probability of "no collision", is bounded by

$$
\begin{aligned}
& p_{k,n} \leq \exp\left(-\frac{k(k-1)}{2n}\right) \\
\Rightarrow \quad & 1 - p_{k,n} \geq 1 - \exp\left(-\frac{k(k-1)}{2n}\right) \geq p \\
\Leftrightarrow \quad & \exp\left(-\frac{k(k-1)}{2n}\right) \leq 1 - p \\
\Leftrightarrow \quad & k^2 - k + 2n \log_e(1-p) \\
& = \left(k - \frac{1}{2} + \frac{1}{2}\sqrt{1 - 8n\log_e(1-p)}\right) \cdot \left(k - \frac{1}{2} - \frac{1}{2}\sqrt{1 - 8n\log_e(1-p)}\right) \geq 0
\end{aligned}
$$

With $n = 2^{64} \approx 1.844 \cdot 10^{19}$ and $p = 0.86$, we get $k_{64} \approx 8.517 \cdot 10^9$, and with $n = 2^{128} \approx 3.403 \cdot 10^{38}$, we get $k_{128} \approx 3.658 \cdot 10^{19}$, where k_{64} and k_{128} denote the number of messages needed to get a collision with probability of $p = 0.86$.

b) The following solution is an example and other solutions are possible. The main aspect of this exercise is to show the growth in resources for generating collisions the longer the hash function is.

Hardware resource	64 bit hash function	128 bit hash function
hash function executions	$k_{64} = 8.517 \cdot 10^9$	$k_{128} = 3.658 \cdot 10^{19}$
memory size	$k_{64} \cdot 64$ bits ≈ 63.5GiB	$k_{128} \cdot 128$ bits $= 5.45 \cdot 10^{11}$GiB
comparisons	$0 + 1 + 2 + \ldots + (k_{64} - 1)$ $= \sum_{i=0}^{k_{64}-1} i = \frac{1}{2} k_{64}(k_{64} - 1)$ $\approx 3.63 \cdot 10^{19}$	$\frac{1}{2} k_{128}(k_{128} - 1) \approx 6.69 \cdot 10^{38}$

Solution of Problem 3

a) see script

b) The modified hash function $h^{'}$ is not preimage resistant, since for any hash value y of the form $0 \,\|\, m$, a preimage is m. Therefore, we can find a preimage for at least one half of all possible hash values.

Next, we prove that $h^{'}$ inherits a second-preimage and collision resistant from h. We show that if we can find a collision or a second preimage for $h^{'}$, then we can easily do so for h. Suppose:

$$\exists x_0 \neq x_1 : h^{'}(x_0) = h^{'}(x_1).$$

Two cases:

- **a)** First bit of $h^{'}(x_0)$ is 0. Impossible as implies that $x_0 = x_1$
- **b)** First bit of $h^{'}(x_0)$ is 1. Then $h^{'}(x_0) = h^{'}(x_1)$ a contradiction, as h is collision resistant.

c) One can verify that for instance

$$h = (10101010, 00001111) = h(00001111, 10101010).$$

Notice that is easy to find other collision pairs like the previous one.

d) Correct signatures will be accepted, since:

$$s^h = g^{zh} = g^{\alpha} = X$$

e) It is possible to sign an arbitrary message without knowing α. We have from the equation in the previous subtask that:

$$s = X^{h^{-1}}$$

Given a message, anyone can hash it to get h, compute h^{-1} by the Extended Euclidean Algorithm and then compute the signature using the user's public key.