**Dr. Michael Reyer**

# Tutorial 4
# - Proposed Solution -

Friday, November 16, 2018

## Solution of Problem 1

**a)** The basic requirements for a cryptographic hash function are given as.

- Given $m \in \mathcal{M}$, $h(m)$ is easy to compute.
- Given $y \in \mathcal{Y}$, it is infeasible to find $m \in \mathcal{M}$, such that $h(m) = y$. If this property holds, we call a hash function *preimage resistant (urbildresistent, Einwegfunktion).*
- Given $m \in \mathcal{M}$, it is infeasible to find $m' \neq m$, such that $h(m) = h(m')$. In this case, $h$ is called *second preimage resistant (schwach kollisionsresistent).*
- It is infeasible to find $m \neq m' \in \mathcal{M}$ with $h(m) = h(m')$. In this case, we call $h$ *(strongly) collision free (stark kollisionsresistent).*

**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 the properties 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:

- First bit of $h'(x_0)$ is 0. Impossible as it implies that $x_0 = x_1$.
- First bit of $h'(x_0)$ is 1. Then $h(x_0) = h(x_1)$ a contradiction, as $h$ is a hash function.

## Solution of Problem 2

Recall Example 10.2: Select $q$ prime, such that $p = 2q + 1$ is also prime (Sophie-Germain-primes). Choose $a, b$ as primitive elements modulo $p$. A message $m = x_0 + x_1 \cdot q$, with $0 \leq x_0, x_1 \leq q - 1$ is then hashed as

$$h(m) = a^{x_0} b^{x_1} \mod p\,.$$

This function is slow but collision free.

*Claim.* If $m \neq m'$ and $h(m) = h(m')$, then $k = \log_a(b) \mod p$ can be determined.

In other words, we show that if $m \neq m'$ with $h(m) = h'(m)$ are known, the discrete logarithm $k = \log_a(b) \mod p$ can be determined, which is known to be computationally infeasible, i.e., it is infeasible to find $m \neq m'$ with $h(m) = h'(m)$.

*Proof.* (proof by contradiction) Let $m = x_0 + x_1 \cdot q$, $m' = x_0' + x_1' \cdot q$.

$$\begin{aligned} & & h(m) &= h'(m) & \\ &\Leftrightarrow & a^{x_0} b^{x_1} &\equiv a^{x_0'} b^{x_1'} & \pmod{p} \\ &\Leftrightarrow & a^{x_0} a^{kx_1} &\equiv a^{x_0'} a^{kx_1'} & \pmod{p} \\ &\Leftrightarrow & a^{k(x_1 - x_1') - (x_0' - x_0)} &\equiv 1 & \pmod{p} \end{aligned}$$

Since $a$ is a primitive element modulo $p$,

$$\begin{aligned} & & k(x_1 - x_1') - (x_0' - x_0) &\equiv 0 & \pmod{p-1} \\ &\Leftrightarrow & k(x_1 - x_1') &\equiv x_0' - x_0 & \pmod{p-1}\,. \end{aligned} \qquad (\star)$$

As $m \neq m'$, it holds that $x_1 - x_1' \not\equiv 0 \pmod{p-1}$. Show that $k = \log_a(b) \mod p$ can be efficiently computed. Assume $1 \leq k, k' \leq p-1$ fulfill $(\star)$. Then,

$$\begin{gathered} k(x_1 - x_1') \equiv x_0' - x_0 \pmod{p-1} \;\wedge\; k'(x_1 - x_1') \equiv x_0' - x_0 \pmod{p-1} \\ \Rightarrow (k - k')(x_1 - x_1') \equiv 0 \pmod{p-1}\,. \end{gathered}$$

It holds $-(p-2) \leq k - k' \leq p-2$ and $x_1 \neq x_1'$ and $-(q-1) \leq x_1 - x_1' \leq q-1$. Let $d = \gcd(x_1 - x_1', p-1)$, then, with $(\star)$, $d \mid x_0' - x_0$.

(i) $d = 1$: $k - k' \equiv 0 \pmod{p-1} \Leftrightarrow k = k' \pmod{p-1}$ has one solution for $1 \leq k, k' \leq p-1$:

$$k = k_0 = (x_1 - x_1')^{-1}(x_0' - x_0) \mod p-1$$

(ii) $d > 1$: With $(\star)$

$$k\left(\frac{x_1 - x_1'}{d}\right) \equiv \frac{x_0' - x_0}{d} \quad (\mathrm{mod}\ \frac{p-1}{d}) \qquad (\star\star)$$

It holds $\gcd\left(\frac{x_1 - x_1'}{d}, \frac{p-1}{d}\right) = 1$. With $(i)$, it follows that $(\star\star)$ has exactly one solution $k_0$, which can be determined by using the Extended Euclidean algorithm we get

$$k = k_0 = \left(\frac{x_1 - x_1'}{d}\right)^{-1} \frac{x_0' - x_0}{d} \mod \frac{p-1}{d}$$

Recall $p - 1 = 2q \Rightarrow d \in \{1, 2, q, 2q\} \Rightarrow d \in \{1, 2\}$ as $(x_1 - x_1') \leq q - 1$. Check, if $a^{k_0} \underbrace{\left[\text{or } a^{k_0 + \frac{p-1}{2}}\right]}_{d=2 \text{ only}} \equiv b \pmod{p}$.

□

## Solution of Problem 3

Two hash functions with output lengths of 64 bits and 128 bits are given.

**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

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

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 $\approx 63.5$GiB | $k_{128} \cdot 128$ bits $= 5.45 \cdot 10^{11}$GiB |
| comparisons | $0 + 1 + 2 + \ldots + (k_{64} - 1)$ $= \sum_{i=0}^{k_{64}-1} i = \frac{1}{2}k_{64}(k_{64} - 1)$ $\approx 3.63 \cdot 10^{19}$ | $\frac{1}{2}k_{128}(k_{128} - 1) \approx 6.69 \cdot 10^{38}$ |