10.1 Security of Hash Functions

Basic requirements

1. $m \in M$, $h(m)$ is easy to compute
2. $Y \in Y$ $(h(m))$ it is difficult to find m with $h(m) = Y$
   one-way function or preimage resistant
3. $m \in M$, it is difficult to find m' s.t. $h(m) = h(m')$
   second preimage resistant
4. ~~Its~~ It is difficult to find m, m' s.t. $h(m) = h(m')$
   (strongly) collision free

Ex 10.1

a) $h(m) = m \mod n = Y$
   fulfills 1, but not 2. take $m = Y$ $\quad m' = m + k \cdot n \quad k \in \mathbb{Z}$
   It's not second preimage resistant (and not collision free)

b) $h(m) = m^2 - 1 \mod p$, p prime
   Not preimage resistant, computing square roots mod p is easy
   $m, m' = -m, m' = m + k \cdot p \Rightarrow$ it's not 2nd preimage resistant

c) $h(m) = m^2 \mod n$, $n = p \cdot q$
   Preimage resistant, if p, q are unknown ($QRSP(a, n) \Leftrightarrow FAC(n)$)
   2nd preimage resistant: see b)

Ex 10.2 | The discrete log hash function

Select q prime s.t. $p = 2q+1$ is also prime // Recall: Sophie-Germain primes - Prop 7.5

Choose two PE $a, b \mod p$

Let $m = x_0 + x_1 \cdot q \quad 0 \leq x_0, x_1 \leq q-1 \Rightarrow 0 \leq m \leq q^2$

Define $h(m) = a^{x_0} b^{x_1} \pmod p$

h maps integers of maximum size q^2 to integers of size p; approx. half as many bits. Further, h is too slow for practical application.

Then, $h(m)$ is strongly collision free.

Proof: If some $m \neq m'$ with $h(m) = h(m')$ is known, then

$k = \log_a(b)$ can be determined $\pmod p$.

Note that: $\exists k : a^k \equiv b \pmod p$ since a is PE mod p

Write $m = x_0 + x_1 \cdot q$ and $m' = x_0' + x_1' \cdot q$

Assume: $h(m) \equiv h(m') \Rightarrow a^{x_0} b^{x_1} \equiv a^{x_0'} b^{x_1'} \pmod p$

$$\Leftrightarrow a^{x_0} (a^k)^{x_1} \equiv a^{x_0'} (a^k)^{x_1'} \pmod p$$

$$\Leftrightarrow a^{k(x_1 - x_1') - (x_0' - x_0)} \equiv 1 \pmod p$$

Since a is PE mod p and Fermat

$$k(x_1 - x_1') - (x_0' - x_0) \equiv 0 \pmod{p-1}$$

$$\Leftrightarrow k(x_1 - x_1') \equiv (x_0' - x_0) \pmod{p-1}$$

It holds that $x_1 - x_1' \not\equiv 0 \pmod{p-1}$, otherwise $m = m'$

Now k can be efficiently calculated, it is easy, if $(x_1 - x_1')^{-1} \pmod{p-1}$ exists

If the output of a hash function consists of n bits, then the probability of guessing a document with a given hash value is approximately 2^{-n}, a usually very small number.
However, the probability of constructing a match is much higher.
This is due to the so called "birthday paradox".

Prop 10.3 / k objects are randomly put into n bins.
Let $P_{k,n}$ denote the probability that no bins contain two or more objects (there is no collision). Then

$$P_{k,n} = \frac{n(n-1)\cdot(n-2)\cdot\ldots\cdot(n-k+1)}{n^k} \le \exp\left(-\frac{(k-1)k}{2n}\right)$$

Proof:

$$P_{k,n} = \frac{\#\text{ collision-free assignments}}{\#\text{ all assignments}} = \left(1-\frac{0}{n}\right)\left(1-\frac{1}{n}\right)\cdot\ldots\cdot\left(1-\frac{k-1}{n}\right)$$

$$= \prod_{i=0}^{k-1} \exp\left(\ln\left(1-\frac{i}{n}\right)\right) = \exp\left(\sum_{i=0}^{k-1} \ln\left(1-\frac{i}{n}\right)\right)$$

$$\overset{(*)}{\le} \exp\left(-\sum_{i=0}^{k-1}\frac{i}{n}\right) = \exp\left(-\frac{k(k-1)}{2n}\right)$$

$(*)$ $\ln(x) \le x-1$ $\quad x>0 \quad x=1-y \Leftrightarrow x-1=-y$
$\Rightarrow \ln(1-y) \le -y$ $\quad y<1$ and exp is monotonically increasing

Let $n=365$ (days), $k=23$ people. Assume that birthdays are equally distributed. It holds:
The prob. that at least 2 people have birthday on the same day $\ge 1/2$
Since $P_{23,365} \le \exp\left(-\frac{22\cdot 23}{2\cdot 365}\right) \approx 0.4999998$
In general $P_{k,n} \le 1/2$, if $k \ge \sqrt{2n\ln(2)}+1 \approx 1.17\sqrt{n}+1$
Since $k-1 \ge \sqrt{2n\ln(2)} \Rightarrow \frac{(k-1)^2}{2n} \ge \ln(2)$

$$\Rightarrow P_{k,n} \le e^{-\frac{k(k-1)}{2n}} \le e^{\frac{-(k-1)^2}{2n}} \le 1/2$$

Applied to hash functions: If $\approx 1.17\sqrt{n}$ hash values are generated then with prob greater than 1/2 there is a collision.
To avoid this choose $n \geq 2^{128}$

<u>Prop 10.4</u>/ (Generalized birthday paradox)
k blue and k red balls are randomly put into n bins.
If $k \sim \sqrt{\lambda n}$, then the prob. that at least one bin contains a red and a blue ball is $\approx 1 - e^{-\lambda}$.

Concrete attack against hash functions with hash length $n = 64$ bits
B generates slight variations at 35 places in the original document.

<u>Ex.</u>: The bank A $\left\{\begin{array}{l}\text{will}\\ \text{promises to}\end{array}\right.$ $\left\{\begin{array}{l}\text{give}\\ \text{let}\end{array}\right.$ B the amount of

20 million
100 $\left\{\begin{array}{l}\text{US}\\ \text{American}\end{array}\right.$ \$ $\left\{\begin{array}{l}\text{before}\\ \text{until}\end{array}\right.$ May 2019 for $\left\{\begin{array}{l}\text{use}\\ \text{invest}\end{array}\right.$ in.

B does the same with a fraudulent document m'.
See differences in red above.

Now, B has generated 2^{35} correct messages with corresponding hash value and 2^{35} fraudulent messages and hash values.
The prob. of having a collision between both groups is given by Prop. 10.4:

$$n = 2^{64}, \quad k = 2^{35}. \quad \lambda = \frac{k^2}{n} = 2^6 = 64 \Rightarrow p = 1 - e^{-\lambda} \approx 1$$

Let m_i and m_j' be the document with $h(m_i) = h(m_j')$

A sign $h(m_i)$, but $(m_j', h(m_i))$ is a valid pair

Note:

This attack needs storing $2 \cdot 2^{35}$ hash values, ≈ 550 GB

Finding a collision can be done with complexity $O(n \log(n))$ by first sorting one group and then comparing each value of the other group with the sorted one.

Defense: Defense against this type of attack: Before signing the hash of a document slightly change it in at least one place, e.g. adding blanks, colons, ...