10.1 Security of Hash Functions

Rep

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

Ex 10.1

a) $h(m) = m \bmod n = 7$
fulfils 1., not 2. $m' = 7 + k \cdot n$, $k \in \mathbb{Z}$
not 3.4
$m \equiv m' \bmod n$

b) $h(m) = m^2 - 1 \bmod p$, p prime
Not preimage resistant, computing square roots mod p is easy
(Not 2nd preimage resistant $m' = (m + kp)$

c) $h(m) = m^2 \bmod n$, $n = p \cdot q$ p, q prime
Preimage resistant $(QRSP(a, n) \Leftrightarrow FAC(n))$
Not 2nd preimage resistant : $m' = m + kp$ $m' = -m$

Ex 10.2 | The discrete log hash function

Select q prime s.t. $p = 2q + 1$ is also prime

Choose two PE a, b mod p

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

Define $h(m) = a^{x_0} b^{x_1} \pmod{p}$. Then $h(m)$ is strongly collision free.

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

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: $a^{x_0} b^{x_1} \equiv a^{x_0'} b^{x_1'} \pmod{p}$

Rewrite: $a^{x_0} a^{k x_1} \equiv a^{x_0'} a^{k x_1'} \pmod{p}$

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

Since a is PE mod p

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

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

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

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

Ex

If the output of a hash function consists of n bits, then the probability of guessing a document with a given hash 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 prob. that no bin contains two or more objects (there is no collision). Then

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

Proof:

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

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

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

$(*)$: $\ln(x) \le x-1,\ x \ge 0$ $\qquad y = 1-x$

$\Rightarrow \ln(1-y) \le -y,\ y \le 1$

Let $n = 365$ (days), $k = 23$ people. Assume that birthdays are uniformly distributed. It holds:

The prob. that at least 2 people have birthday on the same day $\ge 1/2$

Since $p_{23,365} \le \exp\left(-\frac{23\cdot 22}{2\cdot 365}\right) = 0.4999998$

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 $\simeq 1.17\sqrt{n}$ hash values are generated then with prob $\simeq 1/2$ there is a collision.
To avoid this choose $n \geq 2^{128}$ ($n = 2^{160}$ in the DSA)

$\boxed{\text{Prop 10.4}}$ (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 $\simeq 1 - e^{-\lambda}$

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

$$\text{The bank A} \begin{Bmatrix} \text{will} \\ \text{promises to} \end{Bmatrix} \begin{Bmatrix} \text{give} \\ \text{let} \end{Bmatrix} \text{ B the amount of}$$

$$\begin{matrix} \text{20 million} \\ 100 \end{matrix} \begin{Bmatrix} \text{US} \\ \text{American} \end{Bmatrix} \$ \begin{Bmatrix} \text{before} \\ \text{until} \end{Bmatrix} \text{ May 2018 for } \begin{Bmatrix} \text{use} \\ \text{investment} \end{Bmatrix} \text{ in} \ldots$$

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

Now, B has generated 2^{35} correct messages with corresponding hash values and 2^{35} fraudulent messages and hash values.
The prob of having a collision between both groups is given by Prop 10.4:
$n = 2^{64}$, $k = 2^{35}$ $\lambda = \frac{k^2}{n} = 2^6 = 64 \Rightarrow p = 1 - e^{-\lambda} \simeq 1$

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

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

Note:

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

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 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, ...