Hash Functions

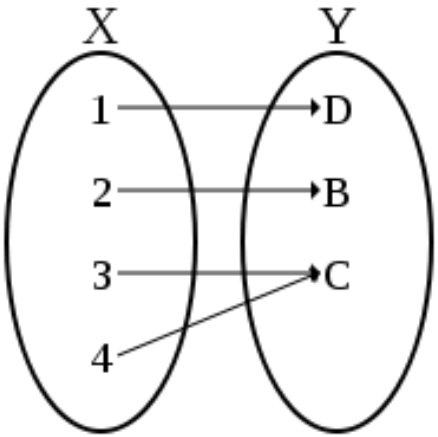
Cryptographic hash functions

Hash function: h

Input: m = message of **arbitrary length**

Output: $h(m)$ = **fixed-length** message **digest**

The possible input messages are theoretically infinite, while the set of possible message digests is finite, hence the function is surjective



Hash function properties

1. Given a message m, its digest $h(m)$ can be calculated very quickly

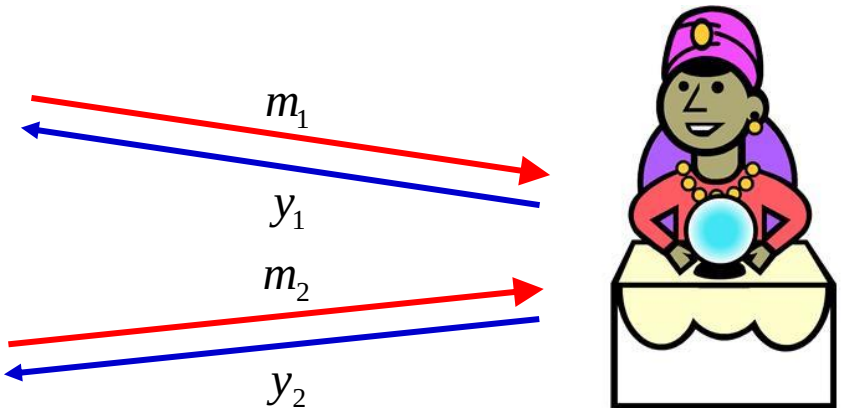
2. **Preimage resistance**: given a message digest y, it is computationally intractable to find a message m' such that $h(m') = y$. This means that $h(m)$ is a **one-way function**.

3. **Second-preimage resistance**: For a fixed x, it is computationally intractable to find $x' \neq x$ such that $h(x') = h(x)$ (***weak collision resistance***).

4. **Collision resistance**: It is computationally intractable to find two different messages, m_1 and m_2, such that $h(m_1) = h(m_2)$ (***strong collision resistance***).

Random Oracle

- Anyone can give some input to the Oracle, which produces an output of fixed length
- If the input has already been provided by someone else earlier, the Oracle always outputs the same value
- If the input was never provided, the Oracle returns a randomly chosen value



Well-known hash functions

- MD4, MD5
- HAVAL-128
- RIPEMD
- SHA-1
- SHA-2 (SHA-224, SHA-256, SHA-384, SHA-512)

For many of these weaknesses were highlighted, or easy-to-find collisions

Algorithm and variant		Output size (bits)	Internal state size (bits)	Block size (bits)	Max message size (bits)	Word size (bits)	Rounds	Operations	Collisions found	Example Performance (MiB/s)[11]
MD5 (as reference)		128	128	512	$2^{64} - 1$	32	64	and,or,xor,rot	Yes	255
SHA-0		160	160	512	$2^{64} - 1$	32	80	+,and,or,xor,rot	Yes	-
SHA-1		160	160	512	$2^{64} - 1$	32	80	+,and,or,xor,rot	Theoretical attack (2^{51})[12]	153
SHA-2	*SHA-256/224*	256/224	256	512	$2^{64} - 1$	32	64	+,and,or,xor,shr,rot	None	111
	SHA-512/384	512/384	512	1024	$2^{128} - 1$	64	80	+,and,or,xor,shr,rot	None	99

SHA-3

Similarly to what was done for AES, on 2 November 2007 NIST has announced a public competition for the standardization of a new hash function designed to take the name SHA-3 and replace SHA-1 and SHA-2

The proposals were to be submitted by 31 October 2008

NIST admitted 51 proposals for Round 1, 14 of them were passed through to Round 2, at the end of which 5 finalists were selected:

- BLAKE
- Grøstl
- JH
- Keccak
- Skein

The winner was announced in October 2012:

Keccak (Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche)

Use of hash functions for data integrity

The digest $h(m)$ is sent along with the message m: $[m, h(m)]$

The recipient receives a pair $[M, H]$ and checks if $H = h(M)$

In this case, the receiver can assume that $M = m$ and hence the received data are reliable

SHA

The Secure Hash Algorithm (SHA) was developed by the National Security Agency (NSA) and published as a NIST (National Institute of Standards and Technology) standard in 1993 (SHA-0).

The identification of a weakness led to a first review in 1995 (SHA-1, with 160-bit digest), which is still the recommended implementation.

In 2001 four new versions were added (grouped under the name SHA-2) characterized by a longer digest: SHA-224, SHA-256, SHA-348 and SHA-512.

In October 2012 the winner of the SHA-3 competition has been nominated, and it was «Keccak» by Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche.

Principle of SHA

SHA is based on an iterative procedure

The message is divided into a set of fixed length blocks:

$$m = [m_1, m_2, ..., m_L]$$

Given a compression function (h') and an initial value (X_0), the output for each message block is calculated as:

$$X_j = h'(X_{j-1}, m_j)$$

The digest of the message coincides with the last output X_L

Birthday paradox

What is the probability that in a class of 23 people there are at least two who were born on the same day?

More than **50%**!

Birthday paradox

If we **choose one person** in a group of 23, what is the probability that at least **one other person** in the group is celebrating his or her birthday on the same day?

$$1-\left(1-\frac{1}{365}\right)^{22}=5.9\%$$

Birthday paradox: given a group of 23 people, what is the probability that **any two of them** celebrate their birthday on the same day?

$$1-\left(1-\frac{1}{365}\right)\left(1-\frac{2}{365}\right)\ldots\left(1-\frac{22}{365}\right)=50.7\%$$

Birthday paradox (2)

Given a group of N elements, with N high, and r random choices (with repetition) of those elements, the probability that at least one choice is carried out twice is about:

$$1-e^{-r^2/2N}$$

If $r^2/2N = \ln 2$, that is, $r \approx 1.177\sqrt{N}$, then the probability that there is at least a repeated choice is 50%

Birthday paradox (3)

Given a group of N elements, with N great, and two groups of r random choices (with repetition) of those elements, the probability that at least one choice of first group matches one of the second is approximately equal to $1 - e^{-\lambda}$, with $\lambda = r^2/N$

The probability that there are exactly i repeated choices between the first group and the second is approximately equal to $(\lambda^i e^{-\lambda} / i!)$

If r is on the order of $\sqrt{N}$, then the probability of having at least one repeated choice is significant

For example, if $N = 365$ and $r = 30$, the probability of having at least one repeated choice is equal to 91.5%

Birthday attacks

The birthday paradox can be exploited for mounting certain attacks that are known as «**birthday attacks**».

Such an approach can be applied each time in which the attack can be modeled as a search of a match between **two lists** of entries **randomly chosen** by the attacker.

In such a case, if the attacker **fixes one entry** of the first list and looks for a match with an entry of the second list, the attack is similar to an exhaustive search (brute force) and the birthday paradox is not exploited.

Instead, if the attacker fills both lists with random entries and then looks for any two matching entries in the two lists, the success probability becomes significantly larger, and the number of required entries is **quadratically smaller** with respect to an exhaustive search.

Birthday attack against a hash function

A birthday attack can be easily applied to the search of **collisions** in a hash function

Let $f(x)$ be a hash function with L-bit digests

If an attacker fixes x_1 and looks for x_2 such that $f(x_1) = f(x_2)$ (***second preimage finding***), he or she must test 2^L values of x_2

If instead the attacker collects a list of $\simeq \sqrt{2^L} = 2^{L/2}$ possible values of x_1 and another list of $\simeq \sqrt{2^L} = 2^{L/2}$ possible values of x_2, and then searches two values x_1 and x_2 in the two lists such that $f(x_1) = f(x_2)$, he or she has success probability of about 50% (yielding a practically successful attack)

The attack complexity (and the security level) thus decreases from L bits to $L/2$ bits

Birthday attack against the discrete logarithm

If p is a large number, we can calculate the discrete logarithm $L_\alpha(\beta)$, or solve $\alpha^x \equiv \beta \pmod{p}$, with high probability, by an attack that exploits the birthday paradox

1. We construct a list containing the values of $\alpha^k \pmod{p}$ for about $\sqrt{p}$ values of k randomly chosen
2. We construct a list containing the values of $\beta\alpha^{-l} \pmod{p}$ for about $\sqrt{p}$ values of l randomly chosen

If an element of the first list coincides with one of the second list:

$$\alpha^k \equiv \beta\alpha^{-l} \Leftrightarrow \alpha^{k+l} \equiv \beta$$