Cryptography, winter 2014/15

Hash functions

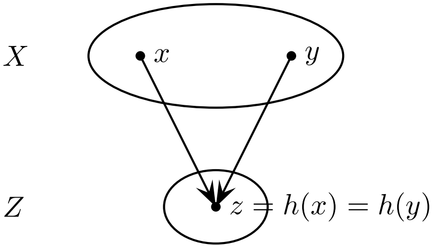
Dr. Daniel Loebenberger

In cryptography, hash functions are used, among other areas, in connection with signature schemes, where only the (short) hash values of (long) messages are signed. In the standard SHA-256, this looks as follows:

length	arbitrary		256 bits		512 bits
data	message x	$\longrightarrow$	hash value $h(x)$	$\longrightarrow$	signature $\mathrm{sig}(h(x))$

Definition

Let $h\colon X \longrightarrow Z$ be a mapping between two finite sets X and Z. If $x \neq y$ are messages in X with $h(x) = h(y)$, then x and y *collide*, and (x, y) is a *collision*.


X
x
y
Z
$z = h(x) = h(y)$

We consider three types of "attackers" $\mathcal{A}$ on a hash function h.

(i) A *collision finder* $\mathcal{A}$ takes no input and outputs either a collision (x, y), or "failure".

(ii) A *second-preimage finder* $\mathcal{A}$ takes an input x and outputs either some $y \in X$ that collides with x, or "failure".

(iii) An *inverter* $\mathcal{A}$ takes an input $z \in Z$ and outputs either some $x \in X$ with $h(x) = z$, or "failure".

In any of these situation, we define the *success probability* $\sigma_{\mathcal{A}}$ of $\mathcal{A}$ as

$$\sigma_{\mathcal{A}} = \text{prob}\left(\mathcal{A} \text{ does not return "failure"}\right).$$

We consider three types of “attackers” $\mathcal{A}$ on a hash function h.

(i) A *collision finder* $\mathcal{A}$ takes no input and outputs either a collision (x, y), or “failure”.

(ii) A *second-preimage finder* $\mathcal{A}$ takes an input x and outputs either some $y \in X$ that collides with x, or “failure”.

(iii) An *inverter* $\mathcal{A}$ takes an input $z \in Z$ and outputs either some $x \in X$ with $h(x) = z$, or “failure”.

In any of these situation, we define the *success probability* $\sigma_{\mathcal{A}}$ of $\mathcal{A}$ as

$$\sigma_{\mathcal{A}} = \text{prob}\left(\mathcal{A} \text{ does not return “failure”}\right).$$

The probability is taken over the internal random choices of $\mathcal{A}$ and uniformly random choices $x \xleftarrow{\text{⚄⚅}} X$ in (ii), and $z \xleftarrow{\text{⚄⚅}} Z$ in (iii).

Definition

A function $\epsilon\colon \mathbb{N} \longrightarrow \mathbb{R}_{\geq 0}$ is called *negligible* if for all $e \in \mathbb{N}$ there exists N so that

$$\epsilon(n) \leq \frac{1}{n^e} \text{ for all } n \geq N.$$

If ϵ is not negligible, it is called *nonnegligible*.

Definition

Let $h = \{h_n\}$ be a family of hash functions. We call h

collision resistant,

second-preimage resistant,

inversion resistant (or *one-way*),

if for all probabilistic polynomial-time

collision finders $\mathcal{A}$,

second-preimage finders $\mathcal{A}$,

inverters $\mathcal{A}$,

respectively, for h the success probability $\sigma_{\mathcal{A}}$ is negligible as function of n.

Corollary

For a family h of hash functions where the fraction $\#Z_n/\#X_n$ is negligible in n, we have

h collision resistant $\Rightarrow$ h second-preimage resistant $\Rightarrow$ h one-way.

Birthday paradox

How many randomly chosen people have to be in a room to have a probability of at least 50% that two of them have the same birthday, assuming each birthday occurs with equal probability?

Birthday paradox

How many randomly chosen people have to be in a room to have a probability of at least 50% that two of them have the same birthday, assuming each birthday occurs with equal probability?

Surprising answer:

23 people are sufficient!

Theorem:

We consider random choices, with replacement, among m labeled items. The expected number of choices until a collision occurs is $O(\sqrt{m})$.

For $e \in \mathbb{Z}$, we have

$$g^e \text{ generates } G \Longleftrightarrow \gcd(e,d) = 1.$$

Now we let $z \in G$ be also a generator, say $z = g^e$ with random $e \in \mathbb{Z}_d^\times$, and consider the function

$$h_z \colon \begin{array}{rcl} \mathbb{Z}_d \times \mathbb{Z}_d & \longrightarrow & G, \\ (a,b) & \longmapsto & g^a z^b \end{array}.$$

We set $i = \lceil k/(\ell - n - 1) \rceil$, so that x fits into i blocks of $\ell - n - 1$ bits each, and $d = i(\ell - n - 1) - k < \ell - n - 1$ is the excess. We now pad x by d zeros, so that the length of this new word $\bar{x}$ is the multiple $i(\ell - n - 1)$ of $\ell - n - 1$. We split it as follows:

$$\bar{x} = \underbrace{\overbrace{\boxed{}}^{\ell-n-1}}_{x_0}\ \underbrace{\overbrace{\boxed{}}^{\ell-n-1}}_{x_1}\ \cdots\ \underbrace{\overbrace{\boxed{}}^{\ell-n-1}}_{x_{i-2}}\ \underbrace{\boxed{\overbrace{}^{\ell-n-1-d}\,\overbrace{\ 0\ }^{d}}}_{x_{i-1}}\ \underbrace{\overbrace{\boxed{}}^{\ell-n-1}}_{x_i}$$

where each x_j has $\ell - n - 1$ bits. We attach another word x_i of $\ell - n - 1$ bits containing the binary representation of d, filled with leading zeroes.

$$\underbrace{\overbrace{\boxed{\;0\;}}^{n}\overbrace{\boxed{\;x_0\;}}^{\ell-n-1}\overbrace{\boxed{\;0\;}}^{1}}_{y_0}$$

$$\downarrow$$

$$\underbrace{\boxed{h(y_0)}\boxed{\;x_1\;}\boxed{\;1\;}}_{y_1}$$

$$\downarrow$$

$$\underbrace{\boxed{h(y_1)}\boxed{\;x_2\;}\boxed{\;1\;}}_{y_2}$$

$$\downarrow$$

$$\vdots$$

$$\downarrow$$

$$\underbrace{\boxed{h(y_{i-2})}\boxed{\;x_{i-1}\;}\boxed{\;1\;}}_{y_{i-1}}$$

$$\downarrow$$

$$\underbrace{\boxed{h(y_{i-1})}\boxed{\;x_i\;}\boxed{\;1\;}}_{y_i}$$

$$\downarrow$$

$$\boxed{h(y_i)}$$

Theorem

A collision for h^* yields a collision for h.

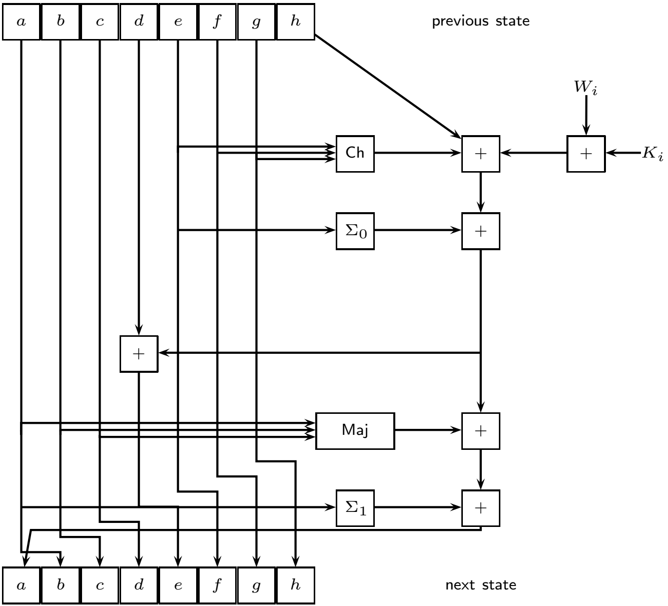


Figure: The ith round of SHA-256.

function	semantic
$+$	The sum modulo 2^{32} of its two input integers
Ch(e, f, g)	*choice*: if e then f else g; equivalenty (e and f) or (not e and g)
$\Sigma_0(a)$	$\mathsf{ROTR}^2(a) \oplus \mathsf{ROTR}^{13}(a) \oplus \mathsf{ROTR}^{22}(a)$
Maj(a, b, c)	*majority*: (a and b) or (a and c) or (b and c)
$\Sigma_1(e)$	$\mathsf{ROTR}^6(e) \oplus \mathsf{ROTR}^{11}(e) \oplus \mathsf{ROTR}^{25}(e)$

Figure: The internal SHA-256 functions. All except $+$ are executed bitwise.

Special-purpose Bitcoin mining hardware computes blocks M of a special structure such that

$$\text{SHA-256}(\text{SHA-256}(M))$$

is smaller than some pre-defined constant B.

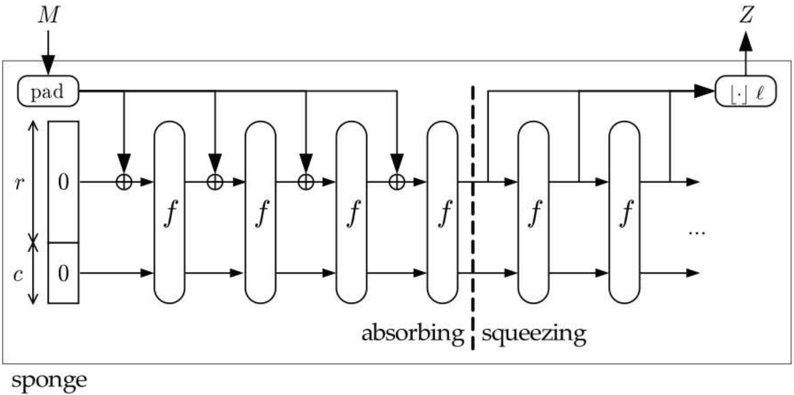


Figure: The SHA-3 sponge construction.

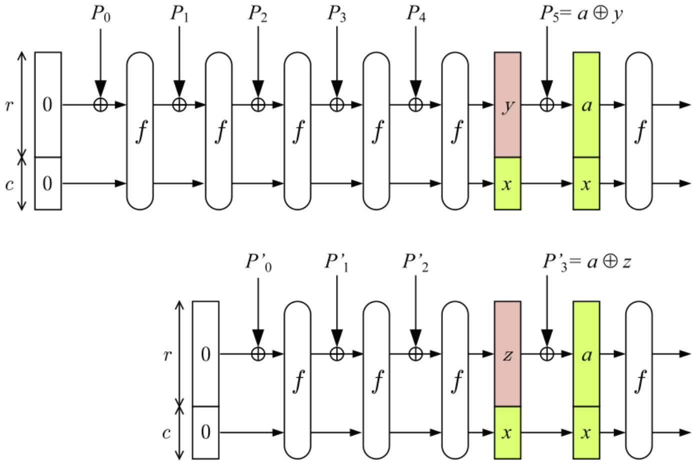


Figure: Collisions in the sponge construction.

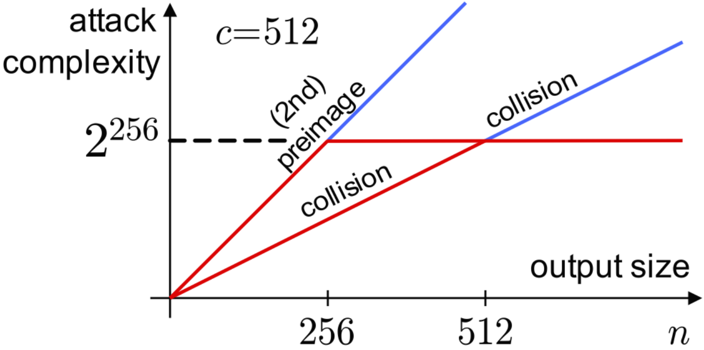


Figure: The sponge claim.

One round of the SHA-3 f function consists of five steps

$$R = \iota \circ \chi \circ \pi \circ \rho \circ \theta,$$

where ι is addition by some round specific constant.

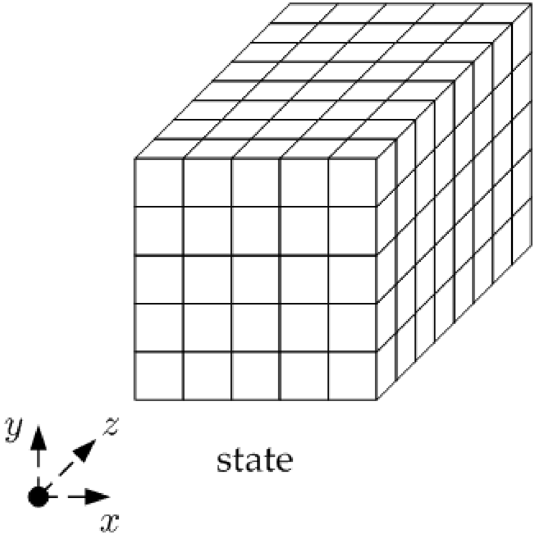


Figure: A state of the SHA-3 f function.

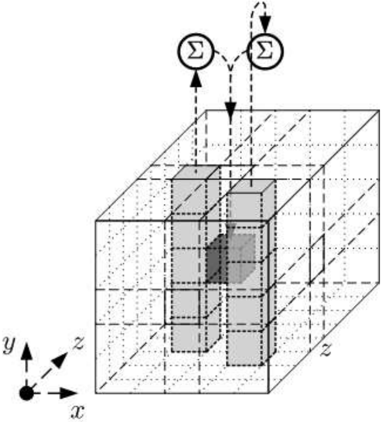


Figure: The step θ in the SHA-3 f function

Compute

$$a[x][y][z] \leftarrow a[x][y][z] + \sum_{y'=0}^{4} a[x-1][y'][z] + \sum_{y'=0}^{4} a[x+1][y'][z-1].$$

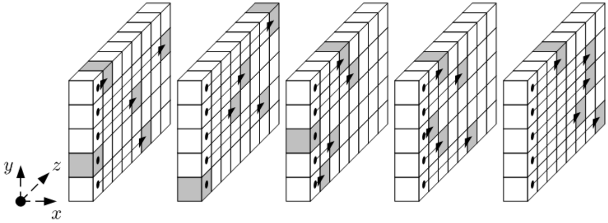


Figure: The step ρ in the SHA-3 f function

Compute

$$a[x][y][z] \leftarrow a[x][y][z - (t+1)(t+2)/2]$$

for some suitably selected $0 \leq t < 24$.

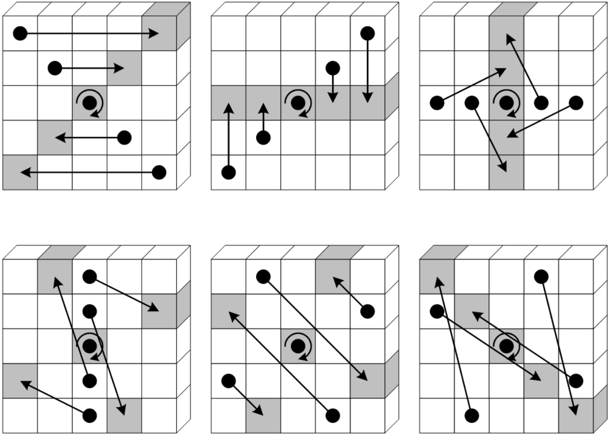


Figure: The step π in the SHA-3 f function

Compute

$$a[x][y] \leftarrow a[x'][y']$$

for some suitably selected x', y'.

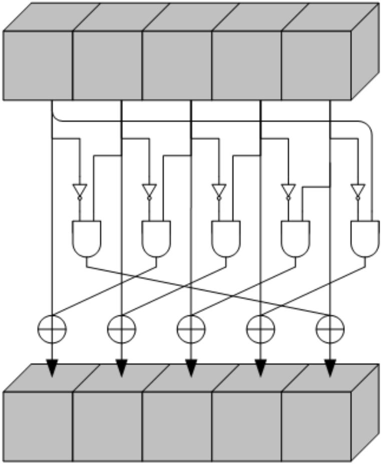


Figure: The step χ in the SHA-3 f function

Compute

$$a[x] \leftarrow a[x] + (a[x+1] + 1)a[x+2]$$