Hash Functions

Cryptology (1)

Martin Stanek

2025

KI FMFI UK Bratislava

- hash function computes a fixed-length fingerprint/digest/hash from a message/document of (almost) arbitrary length
- $h : X \to Y$ function – deterministic, efficient (fast), without any key
- usually $X = \{0,1\}^*$, $X = \{0,1\}^{\leq 2^{64}}$, $X = \{0,1\}^{\leq 2^{128}}$, …
  $Y = \{0,1\}^{160}$ for SHA-1, $\{0,1\}^{256}$ for SHA-256 and SHA3-256, …
- various uses of hash functions:
  - digital signature schemes (digest of the message is signed)
  - padding in public-key encryption schemes
  - verifying integrity of data, MAC constructions
  - instantiation of random oracles and pseudorandom functions
  - proof of work, password storing methods, etc.

Basic requirements of hash functions (informally)

preimage resistance (one-way)
It is infeasible to compute $x \in X$ given $y \in h(X)$ such that $h(x) = y$.

second preimage resistance
It is infeasible to compute $x' \in X$ given $x \in X$ such that $x \neq x'$ & $h(x) = h(x')$.

collision resistance
It is infeasible to compute $x, x' \in X$ such that $x \neq x'$ & $h(x) = h(x')$.

Remarks:

- $|X| \gg |Y|$, otherwise the h.f. is useless $\Rightarrow$ large number of collisions
- Y is finite, h is deterministic $\Rightarrow$ ("hardcoded") collisions can be found in $O(1)$ time in theory
- formalizing the requirements is not straightforward; *hash function families*
- the informal definitions are sufficient for our needs

Properties of hash functions – discussion

- collision resistance $\Rightarrow$ second preimage resistance
  - if you can find a second preimage, then you have a collision
- collision resistance $\not\Rightarrow$ preimage resistance
  - identity: $X = Y, \forall x \in X : h(x) = x$ (Coll, $\neg$Pre)
  - let g with range $\{0,1\}^n$ be collision and preimage resistant; then

$$h(x) = \begin{cases} 0 \parallel x & \text{if } |x| = n \\ 1 \parallel g(x) & \text{otherwise} \end{cases}$$

    is collision resistant but not preimage resistant
- second preimage resistance $\not\Rightarrow$ preimage resistance
  - identity again (Sec, $\neg$Pre)
- however, in a “normal” situation …

Collision by inverting

- assumption: h can be inverted efficiently
- algorithm to find a collision:
  1. $x \leftarrow X$ (random)
  2. invert $h(x) \mapsto x'$
  3. if $x' \neq x$ … collision found
- let us estimate the probability of success
- notation:
  - $[x] = \{x' \in X \mid h(x') = h(x)\}$
  - C – set of all equivalence classes

$$\Pr_{\text{succ}} = \frac{1}{|X|}\sum_{x\in X}\frac{|[x]|-1}{|[x]|} = \frac{1}{|X|}\sum_{c\in C}\sum_{x\in c}\frac{|c|-1}{|c|}$$

$$= \frac{1}{|X|}\sum_{c\in C}(|c|-1)$$

$$= \frac{1}{|X|}\underbrace{\sum_{c\in C}|c|}_{|X|} - \frac{1}{|X|}\underbrace{\sum_{c\in C}1}_{\leq |Y|} \geq 1 - \frac{|Y|}{|X|}$$

- after k repetitions:
  $\Pr_{\text{succ}} \geq 1 - (|Y|/|X|)^k$

Generic attack for finding preimage/2nd preimage

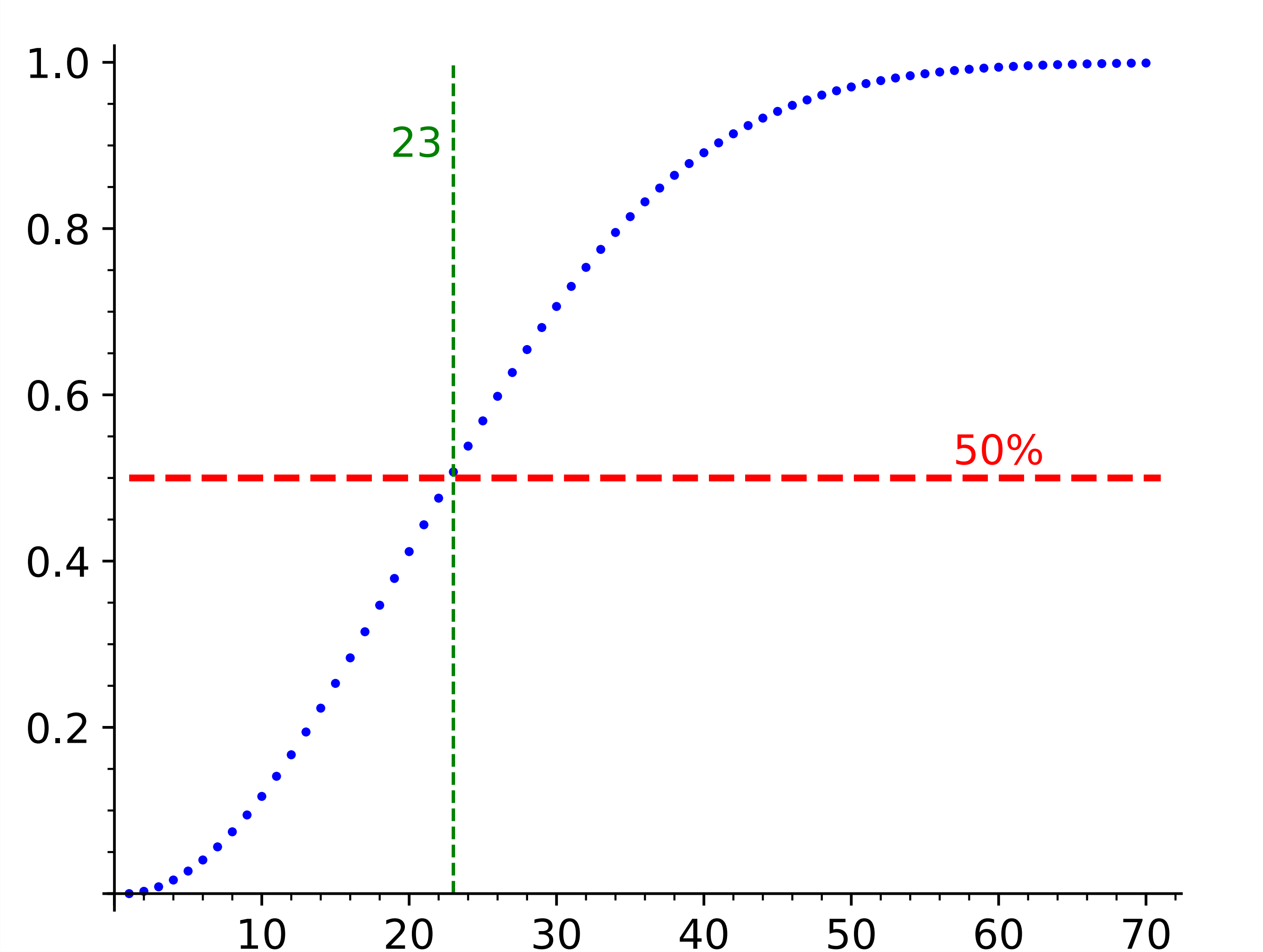
- generic attack, finding a preimage for given $y \in h(X)$:
- algorithm:
  1. choose $x \in X$ (randomly or systematically)
  2. if $h(x) = y$ then the preimage is found, otherwise repeat
- expected complexity $O(2^n)$ for $Y = \{0, 1\}^n$
- similar generic attack for finding a second preimage

- generic attack for finding collisions
- What is the probability that at least two people in a room share the same birthday?
  - assumption: uniform distribution of birthdays

$$\Pr_2 = 1 - \frac{365 \cdot 364}{365^2} \approx 0.0027$$

$$\Pr_3 = 1 - \frac{365 \cdot 364 \cdot 363}{365^3} \approx 0.0082$$

- k people: $\Pr_k = 1 - 365^{\underline{k}}/365^k$
- at least 23 people needed for probability $\geq 1/2$
- "hash function" maps people to dates; $|Y| = 365$; shared birthday = collision


1.0
0.8
0.6
0.4
0.2
23
50%
10
20
30
40
50
60
70

- choose (distinct, random) $x_1, \ldots, x_k \leftarrow X$
- compute $h(x_1), \ldots, h(x_k)$
- find collisions, for example by sorting $(h(x_i), x_i)$ and searching for collisions in adjacent elements, or by storing $(h(x_i), x_i)$ in a hash table using the hash value as a key
- linear time and memory complexity $O(k)$
  - we treat n as a constant (for $Y = \{0,1\}^n$); also assuming constant time to evaluate h
  - time: using Radixsort for sorting in $O(k)$ or using a hash table with $k \times O(1)$ operations
  - memory complexity can be improved (see later)

Birthday attack – analysis (1)

- What is the probability of success?
- trivial observations – the probability of success increases:
  - for increasing k
  - for unbalanced distribution of images
- assume the worst situation: h distributes the hash values uniformly, i.e.

$$\Pr[h(x) = y] = \frac{1}{|Y|} \quad \forall y \in Y$$

- let $y_1, \ldots, y_k$ be random, independent and uniform elements from Y; notation: $|Y| = N$
- probability that all y_i's are distinct:

$$\Pr_{\text{dist}} = \frac{N(N-1)\cdot \ldots \cdot (N-k+1)}{N^k} = \left(1 - \frac{1}{N}\right)\left(1 - \frac{2}{N}\right) \cdot \ldots \cdot \left(1 - \frac{k-1}{N}\right)$$

- probability of at least one collision:
  $\Pr_{\text{col}} = 1 - \Pr_{\text{dist}}$
- let's estimate $\Pr_{\text{col}}$:
  $\Pr_{\text{col}} = 1 - \prod_{i=1}^{k-1}\left(1 - \frac{i}{N}\right)$
  $\geq 1 - e^{-\frac{1}{N}-\frac{2}{N}-\ldots-\frac{k-1}{N}} = 1 - e^{\frac{-k(k-1)}{2N}}$

Remark:

- we use inequality $1 - x \leq e^{-x}$
- it follows from Taylor series:
  $e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \ldots$
- or draw the graphs

- solve for k, such that $\Pr_{\text{col}} \geq \varepsilon$, for a constant $\varepsilon \in (0,1)$:

  $\Pr_{\text{col}} \geq 1 - e^{-k(k-1)/(2N)} \geq \varepsilon \quad \Rightarrow \quad 2N \cdot \ln(1-\varepsilon) \geq -k^2 + k$

  solving quadratic inequality $\Rightarrow \quad k \geq \sqrt{N} \cdot \sqrt{2\ln(1-\varepsilon)^{-1}} \qquad (\star)$
- $(\star)$ a very small constant ignored at the end

Birthday attack – remarks and implications

- the complexity is $O\left(N^{1/2}\right)$ for *reasonable* ε, e.g., 50%, 66%, 99%, …

$$\varepsilon = 50\% : \quad k \approx 1.177 \cdot N^{1/2}$$
$$\varepsilon = 99\% : \quad k \approx 3.035 \cdot N^{1/2}$$
$$\varepsilon = 99.99\% : \quad k \approx 4.292 \cdot N^{1/2}$$

- for $Y = \{0,1\}^n$ we get $O\left(2^{n/2}\right)$
  - for SHA-1 $\approx 2^{80}$, for SHA-256 $\approx 2^{128}$
- generic attack,
  - any hash function can be attacked
  - recall: generic attack for symmetric encryption is brute-force, $O\left(2^{l}\right)$, where l is the length of the key

- the length of hash value should be twice the length of symmetric key used for encryption
- standardized parameters for AES and SHA-2/SHA-3 families:

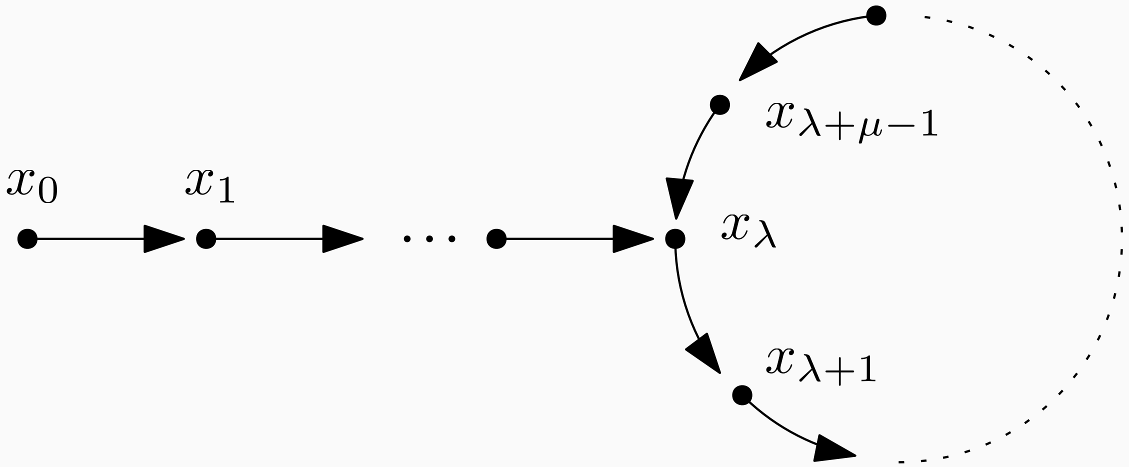
AES key length	SHA-2/SHA-3 output length
	224
128	256
192	384
256	512

“Meaningful” collisions

- prepare documents m, m' with t places that can be changed without changing the meaning of the document
  - one space vs. two spaces, synonyms etc.
- 2^t variants of each document
- hash and find a collision between these two sets
- the same asymptotic time and memory complexity of birthday attack

Improving memory complexity of the birthday attack (1)

- assumption: h as a random function on $h(X)$
- sequence: $x_0, x_1, x_2, \ldots$, where $x_i = h(x_{i-1})$ for $i \geq 1$



- expected (as $N \to \infty$): $\rho = \lambda + \mu = \sqrt{\pi N/2}$

Finding collision in constant memory:

1. $x_0 \leftarrow X$ (using $X \setminus Y$ guarantees the existence of a collision, $\lambda \geq 1$)
2. compute (x_i, x_{2i}) for $i \geq 1$: $x_i = h(x_{i-1}), x_{2i} = h\left(h\left(x_{2(i-1)}\right)\right)$
3. if $x_i = x_{2i}$ then $h^i(x_0) = h^{2i}(x_0)$, we found a point on the cycle, $\lambda \leq i$, and the collision can be computed as follows:
    3.1. compute $\left(x_j, x_{i+j}\right)$ for $j = 0, 1, \ldots, i$ starting with (x_0, x_i)
    3.2. check for situation when $x_j \neq x_{i+j}$ and $x_{j+1} = x_{i+j+1}$
    3.3. collision $h\left(x_j\right) = h\left(x_{i+j}\right)$; remark: $\mu \mid (2i - i) \Rightarrow x_\lambda = x_{i+\lambda}$

Improving memory complexity of birthday attack (3)

- only a constant number of values (e.g. x_0, and the recent pair of values (x_i, x_{2i}) or $\left(x_j, x_{i+j}\right)$) should be stored
- complexity:
  - cycle is detected (point is found) if $i \geq \lambda$ and $\mu \mid i$
  - the difference $2i - i$ increases by 1 in each iteration, i.e. the cycle is detected with $\lambda + \mu$ iterations maximum
  - complexity $O(\lambda + \mu) = O\left(\sqrt{N}\right)$
- this method does not change the asymptotic time complexity of b.a.
- no control over the colliding messages/inputs

Collision resistance in practice

- collision resistance is not easy
- MD5:
  - designed by Ron Rivest in 1991,
  - collision published in 2005
- SHA-1
  - designed by NSA, standard in 1995
  - deprecated by web browsers in 2017
  - first collision in 2017; two pdf files, see https://shattered.io/
  - attack complexity: $2^{63.1}$ SHA-1 compressions

- SHA-1 was replaced fast (use of hash function in signature schemes):

year	SHA-1	SHA-256
01/2015	66.7%	33.3%
01/2016	13.2%	86.8%
01/2017	1.5%	98.4%
01/2018	0.0%	99.8%

Constructions

Hash functions – variety of approaches

- hash functions based on hard computational problems (for example DLOG, SIS)
  - provable properties (assuming the hardness of underlying problem)
  - slow, impractical $\Rightarrow$ not used in practice
- hash functions based on block ciphers
- dedicated constructions

Hash functions based on block ciphers

- $m = m_1, m_2, \ldots, m_k$ input divided into blocks
- sequential processing of input blocks
- h_0 – initialization vector
- h_i – intermediate hash value ($1 \leq i \leq k$)
- $H(m) = h_k$ – the hash value is the output of the last iteration
- problem: small block length
  - specific block ciphers, e.g., SHACAL-2 for SHA-256
  - double block length constructions

Examples:

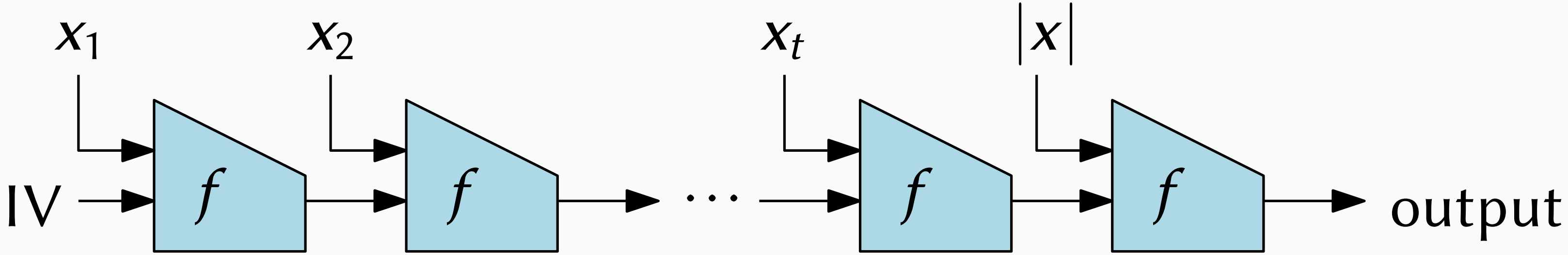
- Matyas, Meyer, Oseas:
  $h_i = E_{g(h_{i-1})}(m_i) \oplus m_i$
- Davies, Meyer:
  $h_i = E_{m_i}(h_{i-1}) \oplus h_{i-1}$
- Miyaguchi, Preneel:
  $h_i = E_{g(h_{i-1})}(m_i) \oplus h_{i-1} \oplus m_i$

Dedicated constructions

- no proofs of security based on some “hard underlying problem”
- fast, usually one of the design goals
- most common design approaches:
  - Merkle-Damgård: SHA-1, SHA-2 family
  - HAIFA: BLAKE2
  - sponge: SHA-3 (Keccak)
  - Merkle tree: BLAKE3
- usually an iterated construction (informally):
  - message padding and slicing
  - start with IV and sequentially process the slices
  - result is the output of the final iteration (sometimes additional processing)

Merkle-Damgård construction (1)

- collision resistance of compression function implies collision resistance of hash function
- fixed input length compression function $f : \{0,1\}^{n+r} \to \{0,1\}^n$
- hash function $H : \{0,1\}^{\leq l} \to \{0,1\}^n$
- input $x = x_1, x_2, \ldots, x_t$ (block length r)
  - last block padded by 10…0 (if needed)
  - additional block $x_{t+1} = |x|$; in binary, thus $l < 2^r$
- other variants of padding used in practice or proposed in the literature
- using the length in padding … *MD strengthening*
  - improves security of the construction (for example: long message attack on second preimage resistance – colliding intermediate values of a very long message and another one)



Computation:

1. $h_0 = 0^n$ (IV)
2. $h_i = f(h_{i-1} \parallel x_i)$, for $i = 1, \ldots, t + 1$
3. $H(x) = h_{t+1}$

Collision resistance of MD construction

Let $x \neq x'$ be a collision in H: $H(x) = H(x')$, i.e., $h_{t+1} = h'_{t'+1}$

– if $t \neq t'$ then $x_{t+1} \neq x'_{t'+1}$ and $f(h_t, x_{t+1}) = f\left(h'_{t'}, x'_{t'+1}\right)$ … collision in f

Let $x \neq x'$ be a collision in H: $H(x) = H(x')$, i.e., $h_{t+1} = h'_{t'+1}$

- if $t \neq t'$ then $x_{t+1} \neq x'_{t'+1}$ and $f(h_t, x_{t+1}) = f\left(h'_{t'}, x'_{t'+1}\right)$ … collision in f
- $t = t'$: $\quad x = x_1, \dots, x_{t+1}, x' = x'_1, \dots, x'_{t+1}$

  $f(h_t, x_{t+1}) = f(h'_t, x'_{t+1})$ … either collision in f or $h_t = h'_t \wedge x_{t+1} = x'_{t+1}$

  $f(h_{t-1}, x_t) = f(h'_{t-1}, x'_t)$ … either collision in f or $h_{t-1} = h'_{t-1} \wedge x_t = x'_t$

  …

  $f(\mathrm{IV}, x_1) = f(\mathrm{IV}, x'_1)$ … either collision in f or $x_1 = x'_1$
- either we get a collision in f or $x = x'$

Merkle-Damgård problems

- structural problems of MD construction
- hash is a complete information ↦ length extension attacks
  - calculating hash of an extended message without knowing the original message
- minimal intermediate state – n-bit for n-bit output
  - multicollisions: with less complexity than expected
- fixed points can be easily found in Davies-Meyer compression function
  - allow more efficient 2nd preimage attacks
- no real-world attacks for suitable parameters, but classical MD constructions are less secure than random functions (oracles)

family	function	length [bits]		
		max. input	output	block
	SHA-1	$2^{64} - 1$	160	512
SHA-2	SHA-256	$2^{64} - 1$	256	512
	SHA-384	$2^{128} - 1$	384	1024
	SHA-512	$2^{128} - 1$	512	1024
SHA-3	SHA3-256	∞	256	1088
	SHA3-384	∞	384	832
	SHA3-512	∞	512	576

SHA-2

- SHA-2 family of hash function
  - SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 and SHA-512/256
- similar design of SHA-256 (32-bit words, block size 512 bits) and SHA-512 (64-bit words, block size 1024 bits)
- other variants are truncated versions with different initialization vectors
- Merkle-Damgård construction

- input message M; $l = |M|$ ($0 \leq l < 2^{64}$ bits)
- padding and parsing:
  - padding: $M1\underbrace{00\ldots0}_{k}\underbrace{(l)_2}_{64\text{ bits}}$, where k is the smallest value such that the overall length is a multiple of 512
  - parsing into 512-bit blocks: $M^{(1)}, M^{(2)}, \ldots, M^{(N)}$
  - each block consists of 16 32-bit words: $M^{(i)} = M_0^{(i)}, M_1^{(i)}, \ldots, M_{15}^{(i)}$
- initialization vector (8 32-bit words): $H_0^{(0)}, H_1^{(0)}, \ldots, H_7^{(0)}$
- intermediate hash values: $H_0^{(i)}, H_1^{(i)}, \ldots, H_7^{(i)}$
- SHA-256 digest: $H_0^{(N)}, H_1^{(N)}, \ldots, H_7^{(N)}$

compression function (for $i = 1, \ldots, N$):

1. expanding a message block ($\mapsto W_0, \ldots, W_{63}$)

$$W_i = \begin{cases} M_t^{(i)} & \text{for } 0 \leq t \leq 15 \\ \sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16} & \text{for } 16 \leq t \leq 63 \end{cases}$$

2. $(a, b, c, d, e, f, g, h) \leftarrow \left(H_0^{(i-1)}, H_1^{(i-1)}, \ldots, H_7^{(i-1)}\right)$
3. for $t = 0, \ldots, 63$:
    1. $T_1 = h + \sum_1(e) + \mathrm{Ch}(e, f, g) + K_t + W_t$, where K_t is a round constant
    2. $T_2 = \sum_0(a) + \mathrm{Maj}(a, b, c)$
    3. $(a, b, c, d, e, f, g, h) \leftarrow (T_1 + T_2, a, b, c, d + T_1, e, f, g)$
4. $H_0^{(i)}, H_1^{(i)}, \ldots, H_7^{(i)} \leftarrow a + H_0^{(i-1)}, b + H_1^{(i-1)}, \ldots, h + H_7^{(i-1)}$

– SHACAL-2 block cipher in Davies-Meyer mode

Functions used in SHA-256

- functions operate on 32-bit words, addition is computed mod $2^{\{32\}}$
- $\mathrm{Ch}(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z)$
- $\mathrm{Maj}(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$
- $\sum_0(x) = \mathrm{ROTR}^2(x) \oplus \mathrm{ROTR}^{13}(x) \oplus \mathrm{ROTR}^{22}(x)$
- $\sum_1(x) = \mathrm{ROTR}^6(x) \oplus \mathrm{ROTR}^{11}(x) \oplus \mathrm{ROTR}^{25}(x)$
- $\sigma_0(x) = \mathrm{ROTR}^7(x) \oplus \mathrm{ROTR}^{18}(x) \oplus \mathrm{SHR}^3(x)$
- $\sigma_1(x) = \mathrm{ROTR}^{17}(x) \oplus \mathrm{ROTR}^{19}(x) \oplus \mathrm{SHR}^{10}(x)$
- ROTR – circular shift rotation to the right
- SHR – shift to the right

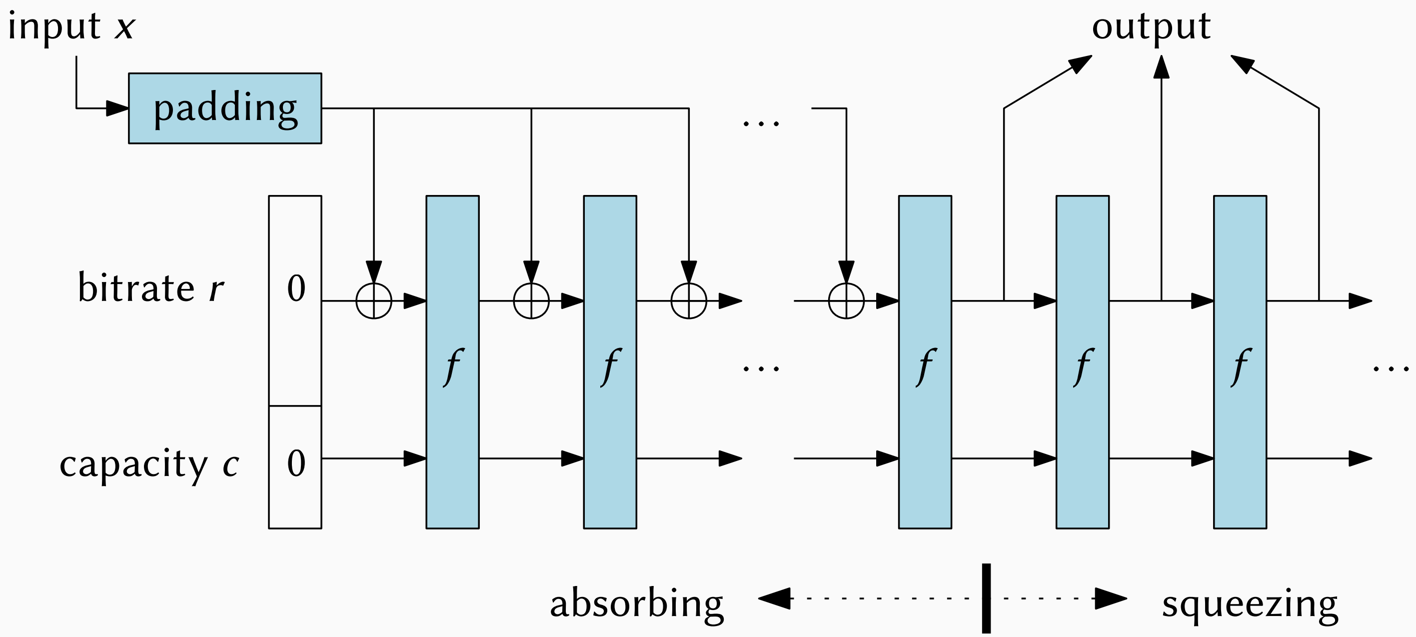
SHA-3 overview

- Keccak – winner of SHA-3 competition (2012)
- standard: NIST FIPS 202 (2015)
  - 4 hash functions with fixed-length output:
    SHA3-224, SHA3-256, SHA3-384, SHA3-512
  - 2 functions with variable-length output (XOF – extendable-output functions):
    SHAKE128, SHAKE256
- different approach than SHA-1 or SHA-2 hash functions
  - Keccak is not an MD-construction
- sponge construction
- other functions/variants proposed:
  - SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash and ParallelHash

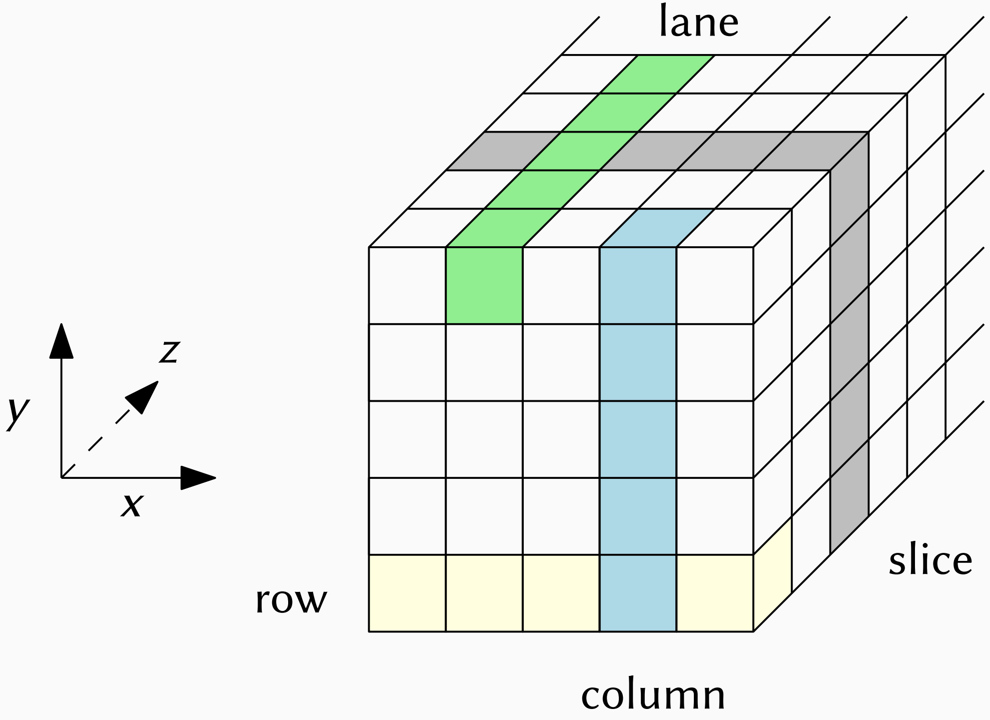
Sponge

- absorbing & squeezing
- arbitrary output length
- padding for SHA3-256: x || 01 || 10^*1

f – permutation on $\{0,1\}^{r+c}$
r – bitrate (1088 for SHA3-256)
c – capacity (512 for SHA3-256)

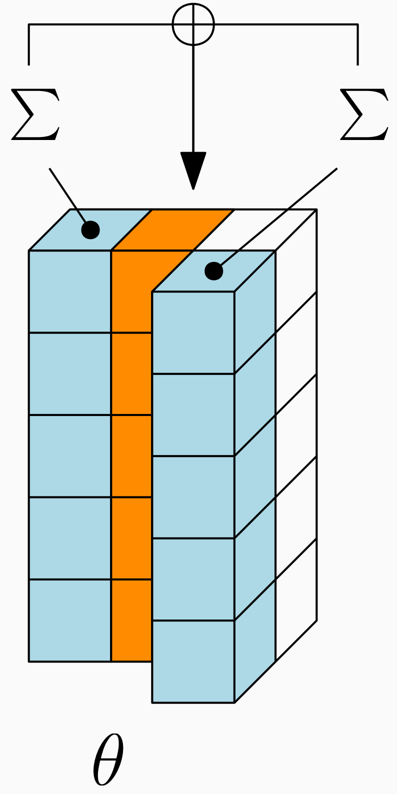


- state: $5 \times 5 \times 2^l$ bits
  - $2^l = 64$ for SHA3-256
- $12 + 2l$ rounds
  - 24 rounds for SHA3-256
- round function (θ is applied first): $R = \iota \circ \chi \circ \pi \circ \rho \circ \theta$



SHA-3 inside permutation f (2)

- θ (theta) – xor each bit of a column with parities of two neighboring columns
- ρ (rho) – rotate each lane by a constant value
- π (pi) – permute the positions of the lanes
- χ (chi) – flip bit if neighbors to the right are 0, 1
  - χ operates on rows (independently, in parallel)
- ι (iota) – xor a round specific constant to lane[0,0]
  - destroying symmetry



1. *Show how we can find fixed points in Davies-Meyer compression function, i.e., how to find* m, h *such that* $f(m, h) = h$.

2. *Discuss the security of a hash function (MD construction) that uses the following compression function, where* E *is a block cipher with 256-bit block and 256-bit key:*
   a) $h_i = E_{h_{i-1}}(m_i) \oplus h_{i-1}$
   b) $h_i = E_{m_i}(m_i) \oplus h_{i-1}$

3. *Let* $h : \{0,1\}^{2n} \to \{0,1\}^n$ *be a collision resistant hash function. Let* $f : \{0,1\}^{4n} \to \{0,1\}^n$ *is defined as follows:* $f(x) = h(h(x_1) \parallel h(x_2))$, *where* $x = x_1 \parallel x_2$ *and* $|x_1| = |x_2| = 2n$. *Prove or disprove:* f *is collision resistant.*