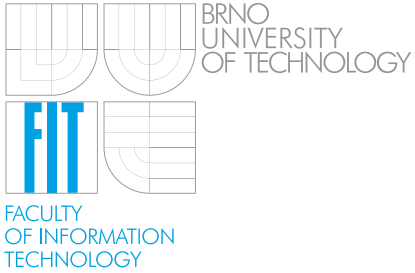
Hash functions

David Grochol

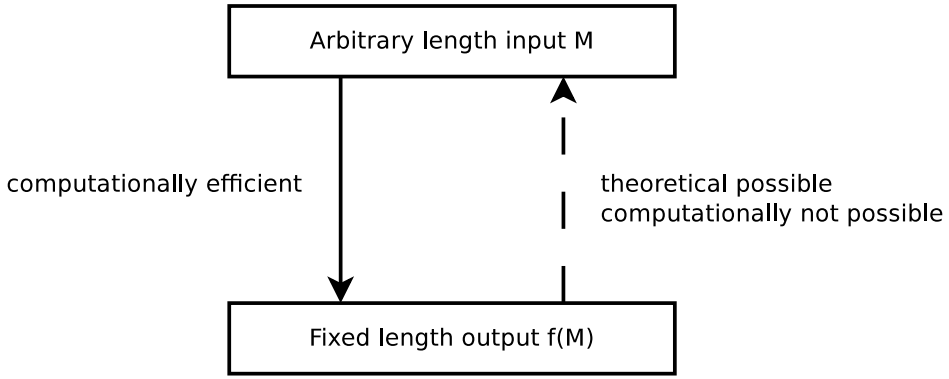
Brno University of Technology, Faculty of Information Technology
Božetěchova 2, 612 00 Brno, CZ
www.fit.vutbr.cz/∼igrochol



LTA 2014, Brno

Hash function - basic concepts

- A hash function is a computationally efficient function mapping input strings of arbitrary length to output strings of some fixed length.
- The output of the hash function is called a hash-value, hash code, hash sum, checksum or simply hash



- Simple hash function: f(M) = M mod 128

Formal Hash function

Hash function:

$$h : \{0,1\}^* \rightarrow \{0,1\}^N$$

Collision:

$$h(a) = h(b) \wedge a \neq b$$

Cryptographic hash function

Need more requirements, for example:

- collision resistance
- small difference in the input data to cause a large difference in the output
- strictly one-way
- and more (details on next pictures)

Requirements for Hash function

Given a function $h : X \longrightarrow Y$, then we say h is:

1. **practical efficient**
   if given $x \in X$ it is computationally efficient to find a value $y \in Y$ s.t. $h(x) = y$
2. **first preimage resistance (one-way)**
   If given $y \in Y$ it is computationally infeasible to find a value $x \in X$ s.t. $h(x) = y$
3. **second preimage resistance (weak collision resistance)**
   if given $x \in X$ it is computationally infeasible to find a value $x' \in X$, s.t. $x' \neq x$ and $h(x') = h(x)$
4. **collision resistance (strong collision resistance)**
   if it is computationally infeasible to find two distinct values $x', x \in X$, s.t. $h(x') = h(x)$

Requirements for Hash function (Continued)

Other requirements

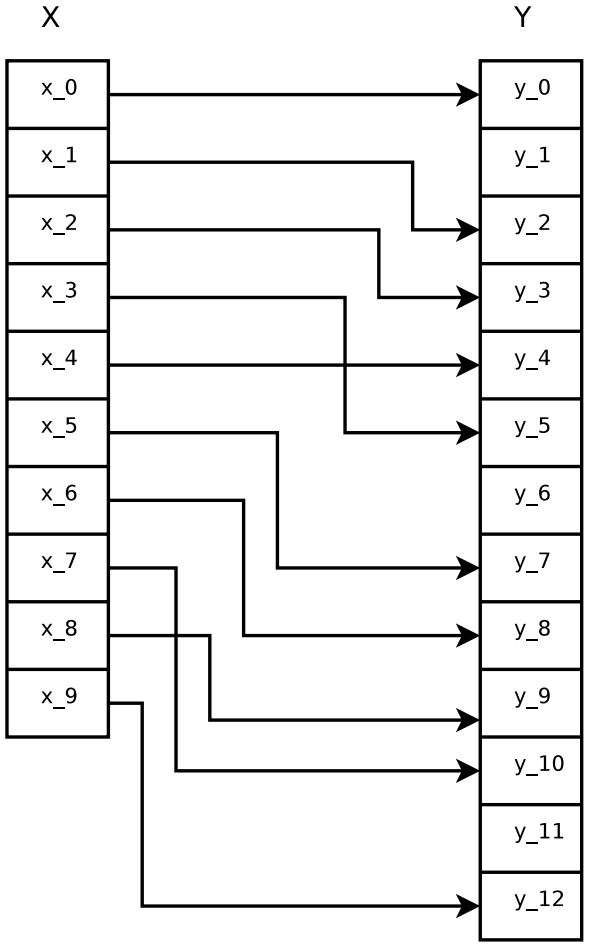
- The hash function requires a small difference in the input data to cause a large difference in the output
- For $\forall x \in X$ output value ($h(x)$) has "random" character
- **Example SHA-1:**
  "password" $\Rightarrow$
  "5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8"
  "passwort" $\Rightarrow$
  "2e2b6533a81bc15430cf65de46dc097eeb5ba70c"

Types hash function

- **One-way hash function (OWHF)**
  – Practical efficient
  – First preimage resistance
  – Second preimage resistance,
  – "random" character.
- **collision resistant hash function (CRHF)**
  – Practical efficient
  – Collision resistance
  – "random" character
- Every CRHF is OWHF.

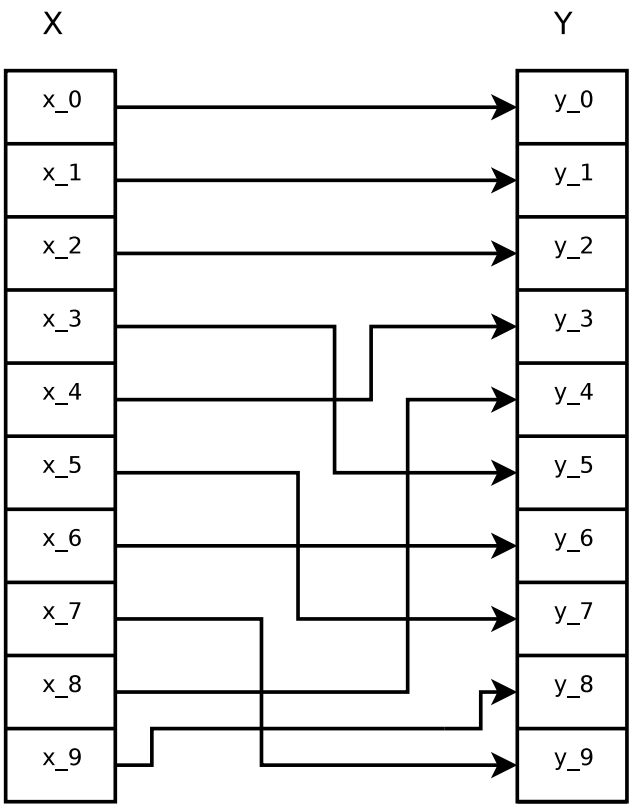
Perfect hash function

- A perfect hash function for a set X is hash function that maps distinct elements in X to a set Y, with no collision and $|X| \leq |Y|$.
- In mathematical terms, it is a total injective function.



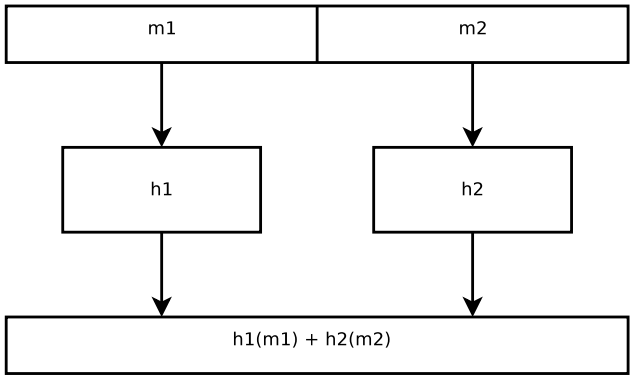
Minimal perfect hash function

- A minimal perfect hash function a set X is hash function that maps distinct element in X to Y, with no collision and $|X| = |Y|$.
- In mathematical terms, it is a total injective surjective function (bijection).

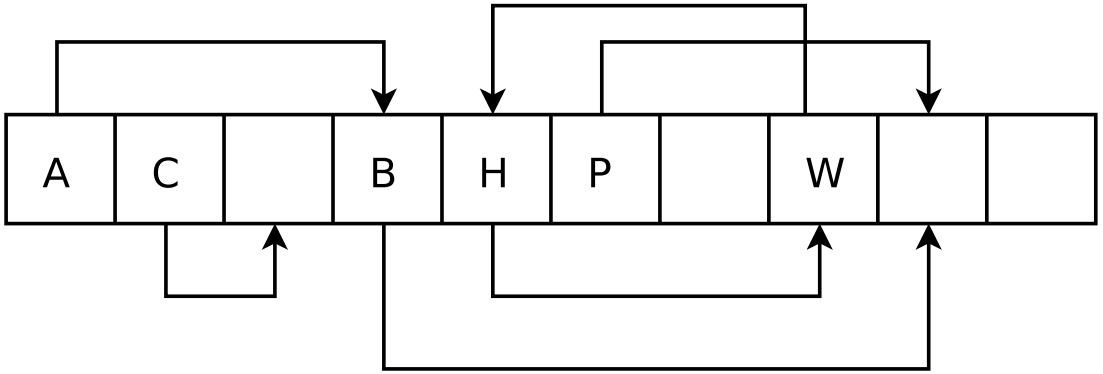


Other way hashing

- **Partitioned Hashing** – Input binary string is split to two (or more) submessage $m = m1 \cdot m2$. Output hash is concatenate $h1(m1)$ and $h2(m2)$.



- **Cuckoo hashing** – Cuckoo hashing use two different hash function. This provides two possible locations in the hash table for each value.



Evaluation of quality hash function

- **Distribution of output hash function**
  A good hash function should map a (nearly) uniform distribution of hash values. No matter what the distribution of the input values was, i.e. return hash $y \in Y$ with probability $|Y|^{-1}$
  Theoretical it is testing all possible inputs.
- **Time complexity hash function**
  Hash function should be very efficient and deterministic operation. We want to have a hash function with minimum time complexity.

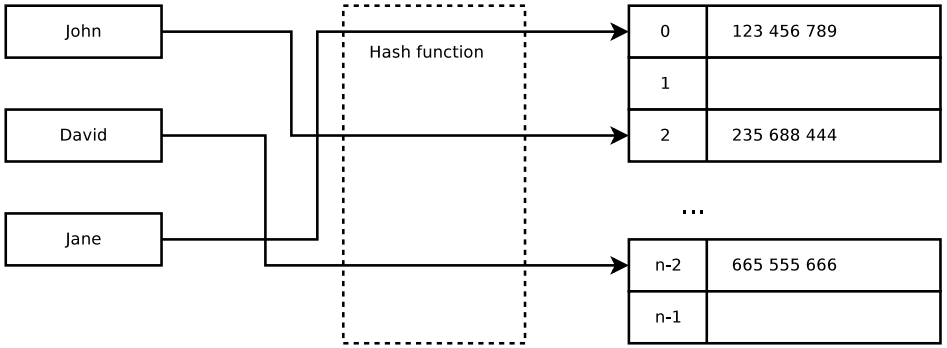
Using Hash function

Hash functions have much use, for example:

- Hash table
- Comparation files
- Password verification
- Verifying the integrity of files or messages
- File or data identifier
- Cyclic redundancy check (CRC)
- Pseudorandom generation number
- and more

Hash Table

- Hash table is a data structure used to implement an associative array, a structure that can map keys to values.



- Hash tables have many uses. For example cache, HDD map, file system, data structure in programing language.

Hash Table (Continued)

Collision resolve in hash table:

- **Separate chaining**
  Each entry contains a table (a pointer to) the list of hash the same place. When the search will go through the entire list of elements. The advantage is that the load factor may be greater than 1.
- **Open addressing**
  This method a hash collision resolving by probing or searching through alternative location in the array until an unused array slot found.
- **Cuckoo hashing**
- **Perfect hashing**

- Password verification
- Verifying the integrity of files or messages
- File or data identifier
- Cyclic redundancy check (CRC)
- Pseudorandom generation number

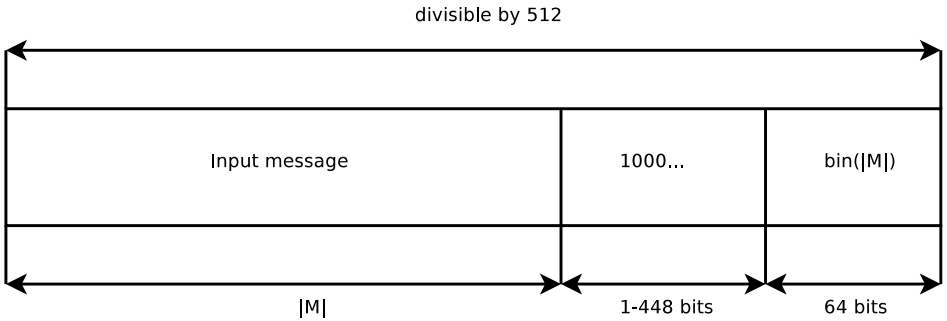
MD5

- The MD5 Message-Digest algorithm is producing 128-bit hash value, typically expressed in text format as a 32 digit hexadecimal number.
- Family of MD algorithm contains MD2, MD3, MD4, MD5 and MD6.
- MD5 was designed in 1992.
- MD5 was cryptographically broken and is still used by some applications.
- MD5 work with 512-bit block.
- Time complexity: O(n)

MD5 - algorithm

MD5 algorithm work in 5 step:

1. Append padding bits
2. Append length



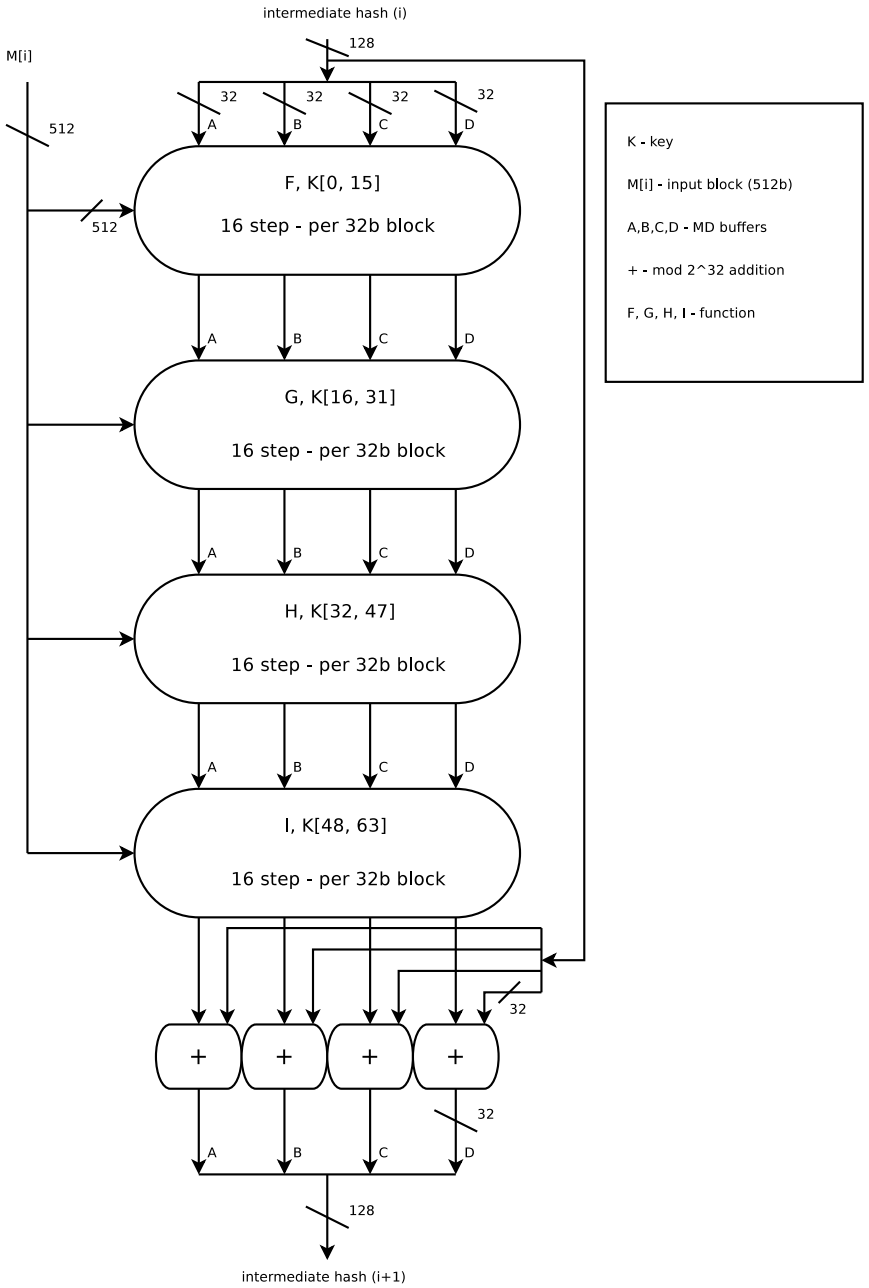
3. Initialize 4 (32-bit) MD buffers
4. Process message block per 16-word blocks (32-bit) in 4 rounds

$$F(X,Y,Z) = X \wedge Y \vee \neg X \wedge Z$$
$$G(X,Y,Z) = X \wedge Z \vee Y \wedge \neg Z$$
$$H(X,Y,Z) = X \oplus Y \oplus Z$$
$$I(X,Y,Z) = Y \oplus (X \vee \neg Z)$$

Notation:
$\wedge$ - bitwise AND
$\vee$ - bitwise OR
$\neg$ - bitwise NOR
$\oplus$ - bitwise XOR

5. Output

MD5 - all rounds



MD5 - example

Preparing input

- **Input:** "Hello world!" $a48656c6c6f20776f726c6421_{16}$
- **Complete input:** $a48656c6c6f20776f726c6421800000-$
  $-0000000000000000000000000000000000000000000068_{16}$

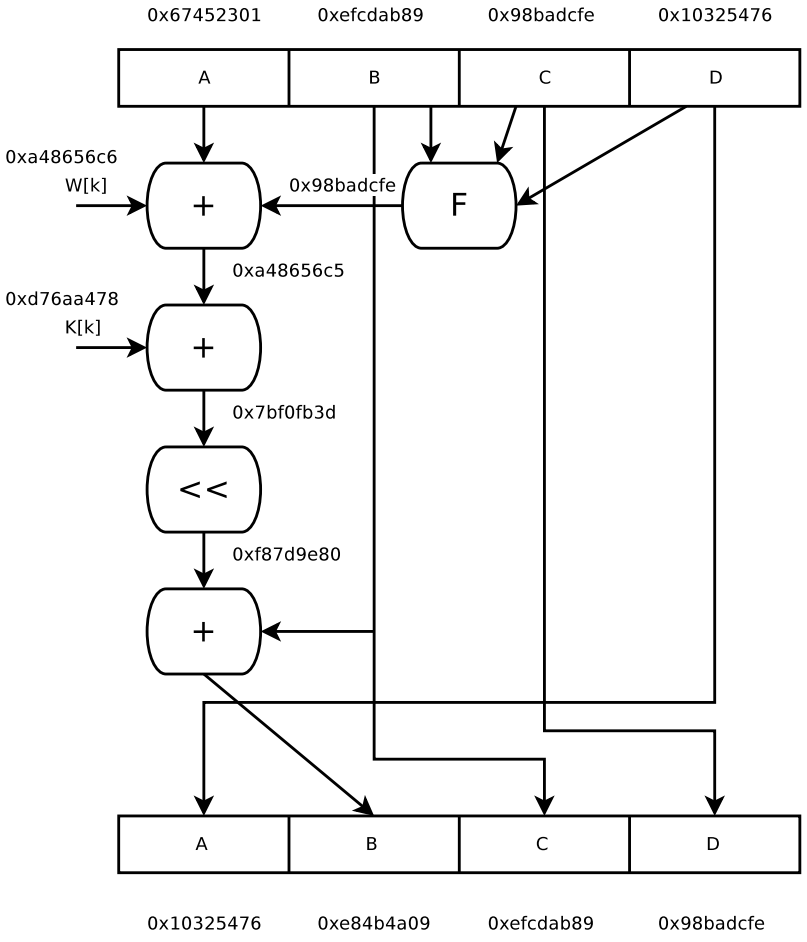
Processing (values for first round)

- **W(0):** $a48656c6_{16}$
- **K(0):** $d76aa478_{16}$
- **buffers:** A:67452301, B:$efcdab89$, C:$98badcfe$, D:10325476
- **shift left:** 7

Final utput

- **MD5("Hello world!"):**
  $86fb269d190d2c85f6e0468ceca42a20_{16}$

MD5 - example first step in first round



K - key

W[i] - word (32 bit)

A,B,C,D - MD buffers

+ - mod 2^32 addition

F - one from F, G, H, I function

<< - shift to left

SHA-2

- SHA-2 is name of 3 algorithm SHA-256, SHA-384, SHA-512.
- SHA-2 was designed in 2001 (SHA-0 (1993), SHA-1 (1995))
- SHA algorithms are based on MD5. SHA-2 works similarly as MD5, but it use different functions and cycle.
- using in TLS and SSL, PGP, SSH, S/MIME, IPsec and cryptocurrencies (Bitcoin, SHA-256)
- Time complexity: O(n)

SHA-256 - algorithm

Algorithm for one block:

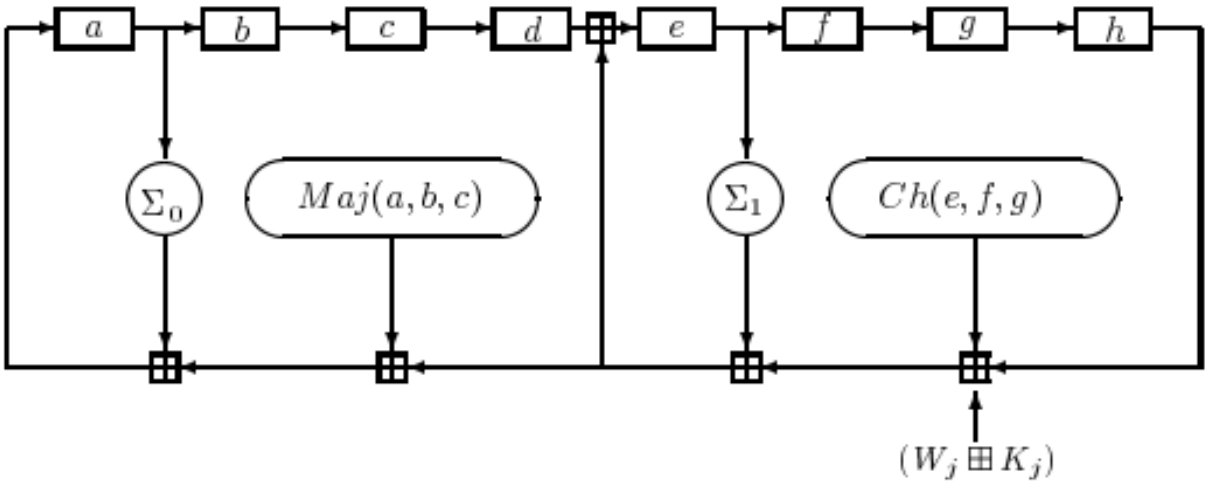
1. Initialize buffers $(a, ..., h) \leftarrow (H_1^{i-1}, ..., H_8^{i-1})$. H^0 constants are defined in SHA-256 standard.
2. Compute functions for block. Definition function:

$$Ch(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z)$$
$$Maj(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$$
$$\Sigma_0(x) = S^2(x) \oplus S^{13}(x) \oplus S^{22}(x)$$
$$\Sigma_1(x) = S^6(x) \oplus S^{11}(x) \oplus S^{25}(x)$$
$$\sigma_0(x) = S^7(x) \oplus S^{18}(x) \oplus R^3(x)$$
$$\sigma_1(x) = S^{17}(x) \oplus S^{19}(x) \oplus R^{10}(x)$$

$\oplus$	bitwise XOR
$\wedge$	bitwise AND
$\vee$	bitwise OR
$\neg$	bitwise complement
$+$	mod 2^{32} addition
R^n	right shift by n bits
S^n	right rotation by n bits

3. Compute the ith intermediate hash value H^i
$(H_1^i, ..., H_8^i) \leftarrow (a + H_1^{i-1}, ..., h + H_8^{i-1})$

SHA-256 - algorithm (Continued)



- Symbol $\boxplus$ denotes mod 2^{32} addition.
- Message is prepared as MD5.
- W_j is j word of block of input message.
- K_j is j constant defines in standard SHA-256.

Other Hash function

Non-cryptographic

- checksum, CRC16/32/64
- fletcher-4/8/16/32
- sum8/16/24/32
- Pearson hashing
- nhash (modulo)
- CityHash

Cryptographic

- BLAKE-256/512
- ECOH
- RIPEMD-128/160/320
- Keccak (SHA-3)
- Spectral Hash
- Tiger
- Whirlpool

Bibliography

Donald Ervin Knuth
The art of computer programming / Vol. 3, Sorting and serching.
Addison-Wesley, 1998.

Computer Security Resource Center
Descriptions of SHA-256, SHA-384, and SHA-512
http://csrc.nist.gov/groups/STM/cavp/documents/shs/sha256-384-512.pdf

R. Rivest
The MD5 Message-Digest Algorithm
https://www.ietf.org/rfc/rfc1321.txt
April 1992.

Bibliography

Maurer, W. D. and Lewis, T. G.
Hash Table Methods
ACM Comput. Surv., 1975.

R. Pagh and F. Rodler
Cuckoo Hashing
Journal of Algorithms, 2004.

B. Majewski,N. Zormald, G. Havas and Z. Czech
A family of perfect hashing methods
The computer Journal, 1996.

Thank you for your attention!