BLAKE Hash Function

BS Project (ECS 412)
(Under Dr. Shashank Singh)

Algorithm of MD5 and BLAKE-256 Hash Function

By:

Ajay Choudhury (18018)

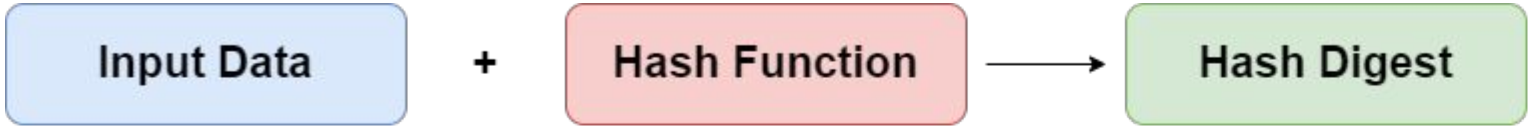
EECS Department

Hash Function

Hash functions are just functions that take arbitrary-length strings and compress them into shorter strings. Mathematically they can be represented as:

$$h : \{0,1\}^{\infty} \to \{0,1\}^{n}$$

Hashing is a process of scrambling a piece of information or data beyond recognition. It is designed to be computationally expensive and practically **irreversible**.



Features of a Cryptographic Hash Function

- **Collision Resistant**

A hash function H is said to be collision resistant if it is computationally impractical to find x and x’ such that H(x) = H(x’).

- **Preimage Resistant**

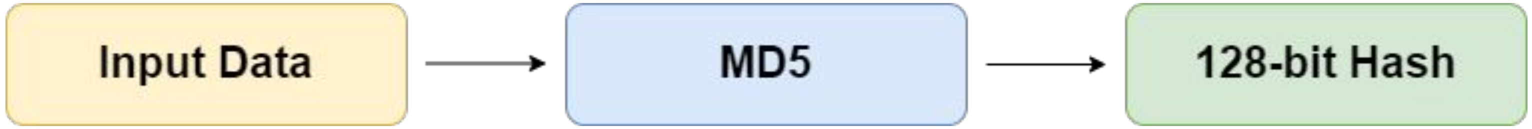
A hash function is said to be preimage resistant if it is computationally impractical for a polynomial-time algorithm to predict the message x given its hash y such that H(x) = y.

- **Second Preimage Resistant**

A hash function is said to be second preimage resistant if given x, it is not feasible for a polynomial-time algorithm to compute y such that H(x) = H(y).

MD5 (Message-Digest) Hashing Algorithm

- One-way cryptographic hash function.
- **128-bit** digest size for every single input.
- Initially designed for digital signatures.
- Designed as a **successor to MD4**, by **Ronald Rivest** in 1991.
- **MD5 is cryptographically broken**, but it is still used in some places.

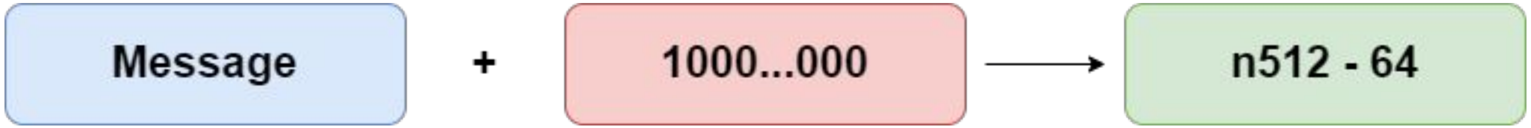


MD5 Algorithm: Step 1: Padding Bits

Some extra bits are added to the original message such that the total length of the output become **64-bit less than a multiple of 512-bit.**

Padding bits have the **first bit as 1 and rest are 0's.**

Example: If there is a string/message of length 910-bit, then a padding of 50-bit is added to the message to make it's length 960-bit = (2 x 512 - 64).

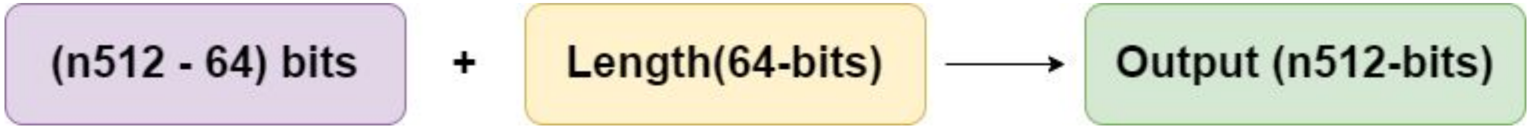


Step 2: Appending Length Bits

To make the total length of the output, a multiple of 512, we need to add a **64-bit length section** to the message in little-endian representation, which is determined by the size of the message.

Example:

- If length of message is 2^8, then length bit will have eight 1's and rest 0's,
- Similarly, for a message of length 2^{64}, 64 1's are added, and
- With messages greater than 2^{64}, modulo of 2^{64} is taken.



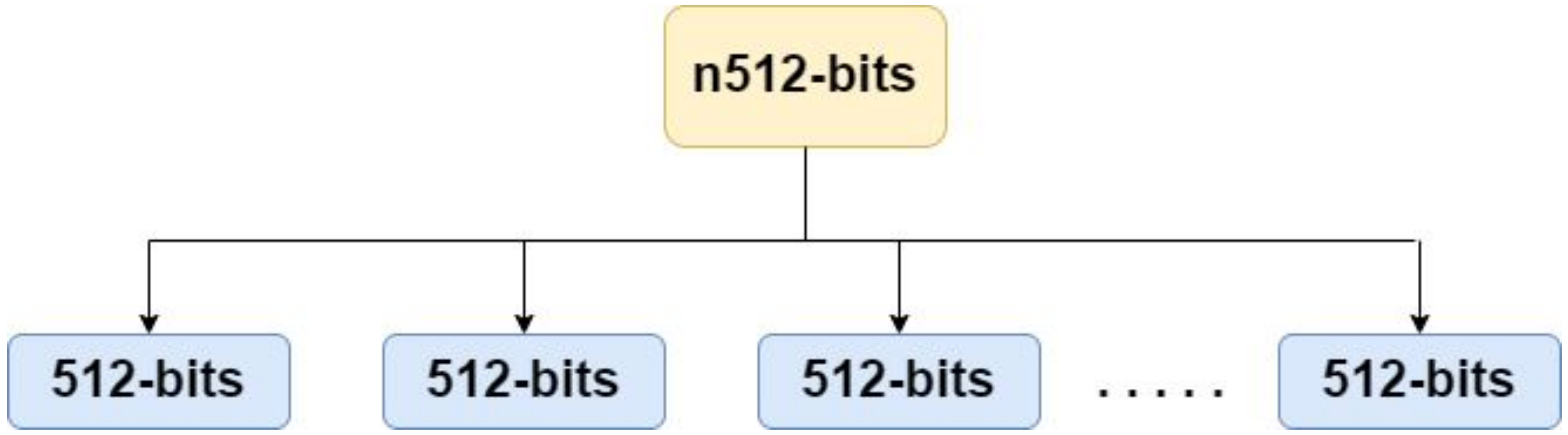
Step 3: Initializing MD Buffer

In MD5, four 32-bit buffer are used, each buffer contains hexadecimal values in it. **A, B, C and D** are the initialized buffers and they are copied into **a, b, c and d** for further operations.

A	01	23	45	67
B	89	ab	cd	ef
C	fe	dc	ba	98
D	76	54	32	10

Step 4: Dividing the Message into Blocks

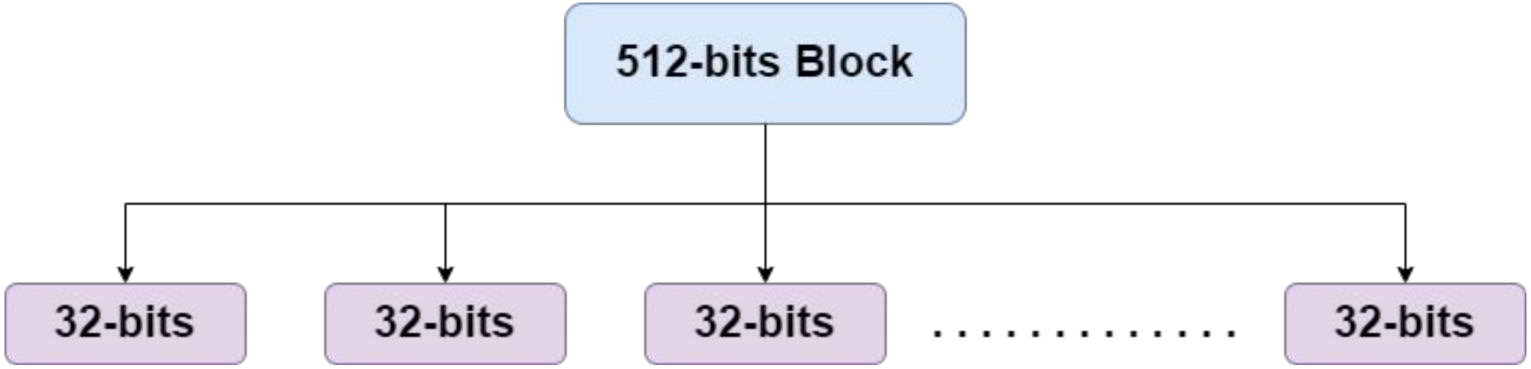
The whole output generated after appending the padding and length bits is now a multiple of 512. So the **message is now divided into n blocks of 512-bit each.**



Step 5: Dividing Blocks into Sub-blocks of 32-bit

Each block of 512-bit is now divided into **16 sub-blocks of 32-bit each** and **four rounds of operations** are performed on each sub-block.

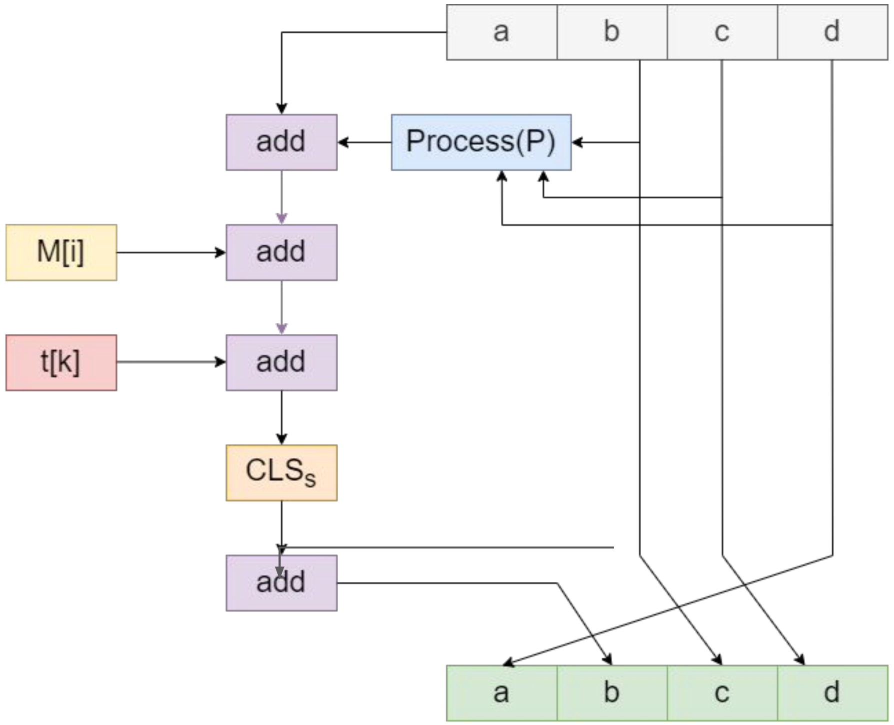
16 operations(one on each sub-block) are performed in each round i.e **64 operations in each block of 512-bit**.



Step 6: Operations on Sub-blocks

Here, the given figure represents a **single MD5 operation on a sub-block** in each 512-bits block.

Here,
a,b,c,d : Chaining variables
P : Non-linear function
M[i] : ith part of the message
t[k] : kth constant
CLS_s : Circular Left Shift by s-bits



Auxiliary Functions or Process (P)

Single MD5 operation can also be represented in equation as:

$$a = b + ((a + P(b, c, d) + M[i] + t[k]) <<< s)$$

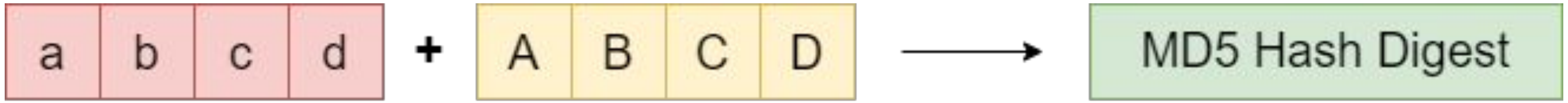
In each round, 16 operations are performed and in each round different auxiliary function or process (P) is used:

$$F(B, C, D) = (B \wedge C) \vee (\neg B \wedge D)$$
$$G(B, C, D) = (B \wedge D) \vee (C \wedge \neg D)$$
$$H(B, C, D) = B \oplus C \oplus D$$
$$I(B, C, D) = C \oplus (B \vee \neg D)$$

Step 7: Generating the Message Digest

After all the rounds have been performed, the buffers **a, b, c and d** are added with the initial buffers **A, B, C and D** and is re-assigned to **a, b, c and d**.

Now, **a, b, c and d contains the MD5 output of the message**, starting from a and ending with d.



The **resultant buffer is passed on as the starting buffer for the next 512-bit block** (if current block is not the last 512-bit block). Similar cycle goes on until the last block from which we get the **128-bit MD5 digest or hash**.

Use of Message Digest (MD5)

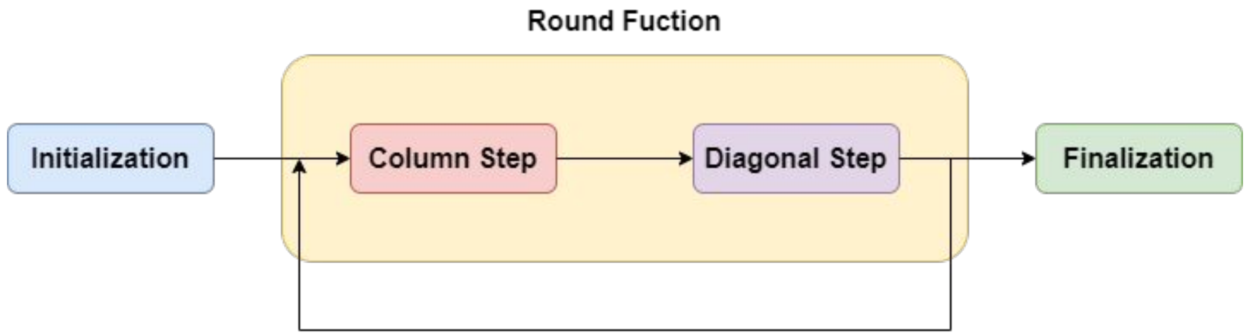
The message digest is usually attached with the message or file and gets shipped to the user. The user can check the integrity of the file or message with the help of message digest as follows:

- User generates the MD5 hash through the same process.
- Compares the locally generated message digest with the shipped digest.
- If it matches then the file is untampered, otherwise the file/message has been tampered.

MD5 hash function has now been replaced by more robust hash functions like **Secure Hash Algorithms(SHA)**, one such example is **BLAKE**.

BLAKE Hash Function

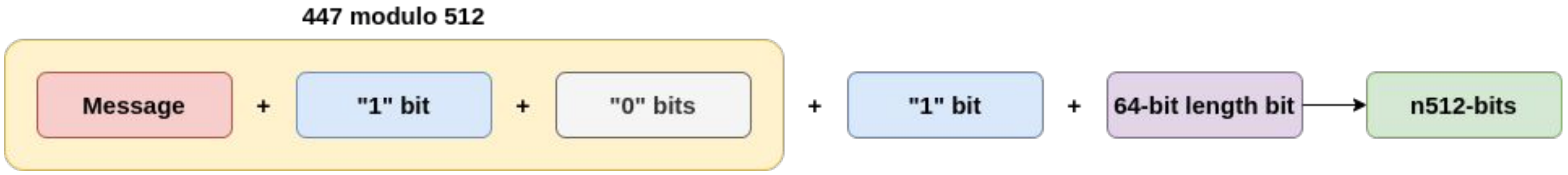
- Cryptographic hash function.
- Many variants like BLAKE-224, BLAKE-256, BLAKE-384 and BLAKE-512.
- Proposed as a standard for **SHA-III** and was selected in the final round.
- Designed by Jean-Philippe Aumasson, Luca Henzen, Willi Meier and Raphael C.-W. Phan.



Algorithm- Step 1: Padding Bits

The data to be hashed (at most 2^{64} −1 bits) is first padded such that its length reaches a multiple of 512. It is then split into 512-bit blocks. It involves two steps:

1. Append to the data a bit “1” followed by the minimal (possibly zero) number of bits “0” so that the total length is congruent to 447 modulo 512. Thus, at least one bit and at most 512 are appended.
2. Append to the data a bit “1” followed by a 64-bit unsigned little-endian representation of the original data bit length. Then, the 512-bit blocks are divided into 16 32-bit sub-blocks.



Step 2: Initialization

The whole block is now divided into blocks of 512-bits. A 512-bit (16 word) initial state is initialized and represented as a 4 x 4 array. It is initialized with **h, s, t** and word constants **c**. It also uses a table of 10 16-element permutations.

$$\begin{pmatrix} v_0 & v_1 & v_2 & v_3 \\ v_4 & v_5 & v_6 & v_7 \\ v_8 & v_9 & v_{10} & v_{11} \\ v_{12} & v_{13} & v_{14} & v_{15} \end{pmatrix} \leftarrow \begin{pmatrix} h_0 & h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 & h_7 \\ s_0 \oplus c_0 & s_1 \oplus c_1 & s_2 \oplus c_2 & s_3 \oplus c_3 \\ t_0 \oplus c_4 & t_0 \oplus c_5 & t_1 \oplus c_6 & t_1 \oplus c_7 \end{pmatrix}$$

Here,

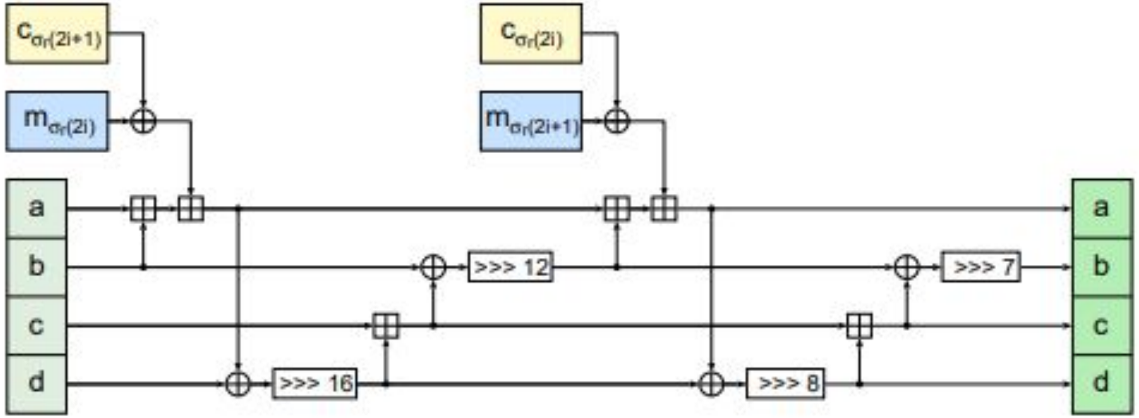
- **h:** chain value,
- **s:** salt,
- **c:** constant, and
- **t:** counter

Permutation Table

σ_0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
σ_1	14	10	4	8	9	15	13	6	1	12	0	2	11	7	5	3
σ_2	11	8	12	0	5	2	15	13	10	14	3	6	7	1	9	4
σ_3	7	9	3	1	13	12	11	14	2	6	5	10	4	0	15	8
σ_4	9	0	5	7	2	4	10	15	14	1	11	12	6	8	3	13
σ_5	2	12	6	10	0	11	8	3	4	13	7	5	15	14	1	9
σ_6	12	5	1	15	14	13	4	10	0	7	6	3	9	2	8	11
σ_7	13	11	7	14	12	1	3	9	5	0	15	4	8	6	2	10
σ_8	6	15	14	9	11	3	0	8	12	2	13	7	1	4	10	5
σ_9	10	2	8	4	7	6	1	5	15	11	9	14	3	12	13	0

Step 3: Round Function in BLAKE-256

A round function iterates 14 times on the state **v** and uses a single core function **G**. The core operation **G** operates on a 4-word column or diagonal **a, b, c, d**, which is combined with 2 words of message **m[ ]** and two constant words **c[ ]**. It is performed 8 times per full round i.e. 112 times in 14 rounds. The operation is depicted visually as:



Core Function G

The operation can also be depicted in form of equations as:

$$\begin{aligned}
a &\leftarrow a + b + (m_{\sigma_r(2i)} \oplus c_{\sigma_r(2i+1)}) \\
d &\leftarrow (d \oplus a) \ggg 16 \\
c &\leftarrow c + d \\
b &\leftarrow (b \oplus c) \ggg 12 \\
a &\leftarrow a + b + (m_{\sigma_r(2i+1)} \oplus c_{\sigma_r(2i)}) \\
d &\leftarrow (d \oplus a) \ggg 8 \\
c &\leftarrow c + d \\
b &\leftarrow (b \oplus c) \ggg 7
\end{aligned}$$

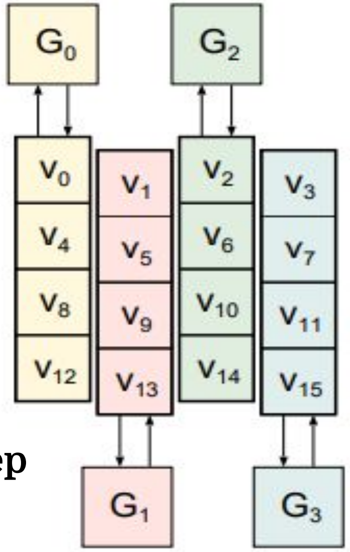
Here,

- **r** is the round number (0–13), and
- **i** varies from 0 to 7.
- **“+”** here means addition modulo 2^{32}, and
- “$\oplus$” refers to the XOR operation.

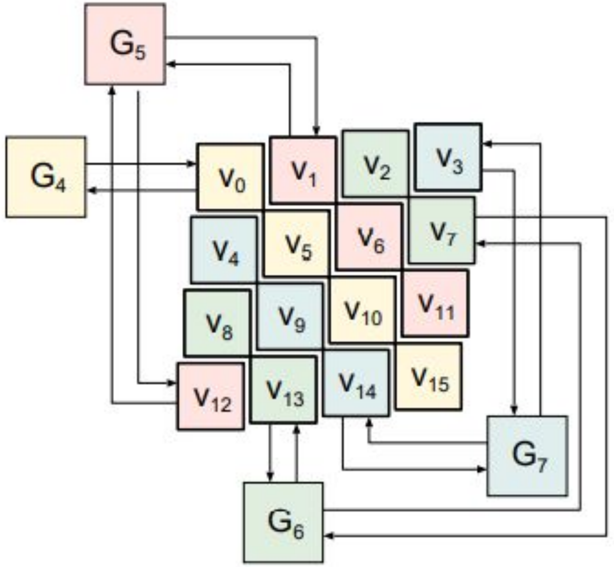
Here, (a >>> s) means right shift operation by **s-bit** on **a**. at round $r \geq 10$, the permutation used is $\boldsymbol{\sigma}_{r \bmod 10}$, for example, at the last round (r = 15), the permutation $\boldsymbol{\sigma}_{15 \bmod 10} = \boldsymbol{\sigma}_5$ is used

Column and Diagonal Step

Here, **G** is applied to the word-matrix in two ways: **column-wise** and **diagonally**. Firstly, G is applied to each column with 4 state variables and next the core function G is applied diagonally as depicted in the figure below.



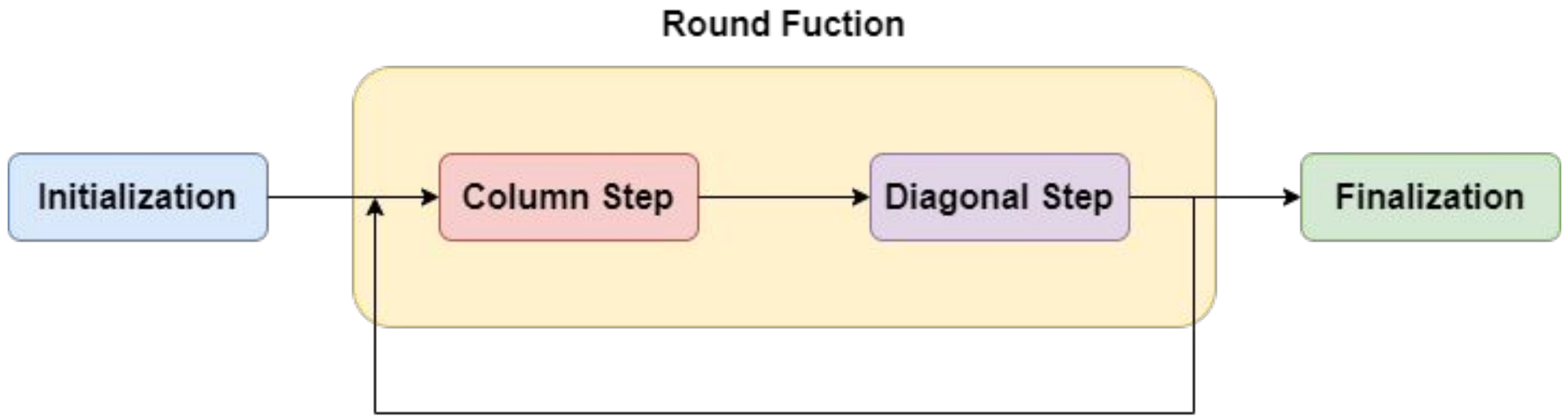
Column Step



Diagonal Step

One Round in Brief

In one round, the core function G is run for 8 times i.e. in 14 rounds it is invoked 112 times in total.



Step 4: Finalization

The new chain values are extracted from the modified state variables and XOR operations are performed on them along with salts and the initial chain values to get the final hash value. It is depicted as:

$$\begin{aligned}
h'_0 &\leftarrow h_0 \oplus s_0 \oplus v_0 \oplus v_8 \\
h'_1 &\leftarrow h_1 \oplus s_1 \oplus v_1 \oplus v_9 \\
h'_2 &\leftarrow h_2 \oplus s_2 \oplus v_2 \oplus v_{10} \\
h'_3 &\leftarrow h_3 \oplus s_3 \oplus v_3 \oplus v_{11} \\
h'_4 &\leftarrow h_4 \oplus s_0 \oplus v_4 \oplus v_{12} \\
h'_5 &\leftarrow h_5 \oplus s_1 \oplus v_5 \oplus v_{13} \\
h'_6 &\leftarrow h_6 \oplus s_2 \oplus v_6 \oplus v_{14} \\
h'_7 &\leftarrow h_7 \oplus s_3 \oplus v_7 \oplus v_{15}
\end{aligned}$$

Applications and Potential of BLAKE Hash Function

- The successor of BLAKE: BLAKE2b is faster than MD5, SHA-1, SHA-2, and SHA-3, on 64-bit x86-64 and ARM architectures.
- BLAKE2 provides better security than SHA-2 and similar to that of SHA-3.
- RAR file archive format version 5 supports an optional 256-bit BLAKE2sp file checksum instead of the default 32-bit CRC32; it was implemented in WinRAR v5+.
- BLAKE2sp is the BLAKE2 version used by 7zip file compressor signature in context menu "CRC SHA".

References

1. Rivest R. (April 1992). “Step 4. Process Message in 16-Word Blocks”. The MD5 Message-Digest Algorithm. IETF.
2. SHA-3 proposal BLAKE, version 1.3. [Source]- SHA-3 proposal BLAKE (aumasson.jp).
3. The Hash Function BLAKE- Jean-Philippe Aumasson, Willi Meier Raphael C.-W. Phan, Luca Henzen.