**RIPEMD family of algorithms**

**Introduction:**

The RIPEMD family of algorithms are a series of cryptographic hash functions developed by a group of researchers from Belgium and Germany in the late 1990s. The acronym "RIPEMD" stands for "RACE Integrity Primitives Evaluation Message Digest".

Hash functions are functions that map bitstrings of arbitrary finite length into strings of fixed length. Given h and an input x, computing h(x) must be easy. A one-way hash function must satisfy the following properties:

* Preimage resistance: it is computationally infeasible to find any input which hashes to any pre-specified output
* Second preimage resistance: it is computationally infeasible to find any second input which has the same output as any specified input.
* Collision resistance: it is computationally infeasible to find a collision, i.e. two distinct inputs that hash to the same result.

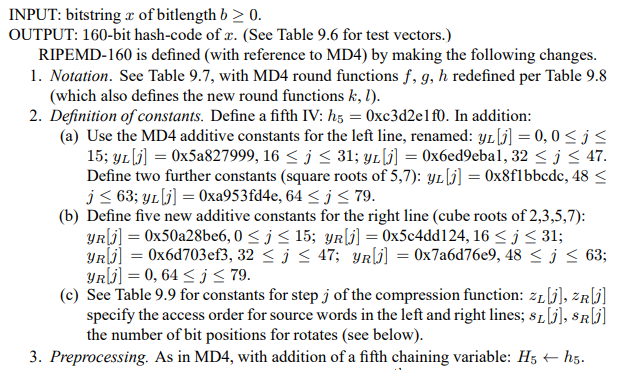
There are several versions of RIPEMD, including RIPEMD-128, RIPEMD-160, RIPEMD-256, and RIPEMD-320. Each version produces a fixed-size hash value, or message digest, of the input data, which can be of any length. RIPEMD-128 and RIPEMD-160 are designed to provide similar security to the widely-used MD5 and SHA-1 hash functions, respectively, but with higher resistance to collision attacks. RIPEMD-256 and RIPEMD-320 offer even stronger security guarantees, and are intended for use in applications where high security is required.

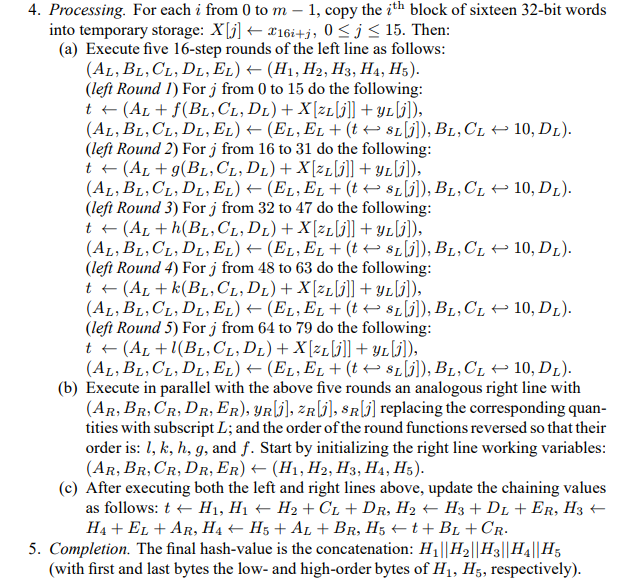
**RIPEMD-160**

RIPEMD-160 is a 160-bit cryptographic hash function, designed by Hans Dobbertin, Antoon Bosselaers, and Bart Preneel. It is intended to be used as a secure replacement for the 128-bit hash functions MD4, MD5, and RIPEMD. MD4 and MD5 were developed by Ron Rivest for RSA Data Security, while RIPEMD was developed in the framework of the EU project RIPE (RACE Integrity Primitives Evaluation, 1988-1992). This algorithm was designed due to the following reasons-

* A 128-bit hash result does not offer sufficient protection anymore. A brute force collision search attack on a 128-bit hash result requires 264 or about 2.1019 evaluations of the function. In 1994 Paul van Oorschot and Mike Wiener showed that this brute-force job can be done in less than a month with a $10 million investment. This cost is expected to halve every 18 months.
* In the first half of 1995 collisions for two out of three rounds of a version of RIPEMD was found. Using similar techniques used for RIPEMD collisions for all three rounds of MD-4 were produced. The attack on MD4 required only a few seconds on a PC, and still leaves some freedom as to the choice of the message, clearly ruling out MD4 as a collision resistant hash function. Shortly in 1996, collision for the compression function of MD-5 were also discovered. Although not yet extended to collisions for MD5 itself, this attack casts serious doubts on the strength of MD5 as a collision resistant hash function.

**Algorithm:**

****

****

**Working:**

**Input:**

Message: The message is the input data to the hash function, expressed as a sequence of bits. The message is divided into blocks of 512 bits.

Message Length: The length of the message in bits must be provided to the algorithm, so that it can properly pad the last block of the message.

Padding: The algorithm pads the message with a single '1' bit followed by a variable number of '0' bits, so that the length of the final block is exactly 512 bits. The final 64 bits of the padded message are used to represent the length of the original message

**Initialization**

Message padding - Message padding in RIPEMD-160 is a process that ensures that the length of the input x 512 bits, which is the block size of the hash function. Padding is necessary because RIPEMD-160 processes input messages in 512-bit blocks, and if the message length is not a multiple of 512 bits, then the last block of the message must be padded with zeros to make it a full block.

The message padding scheme used in RIPEMD-160 is similar to the one used in the MD5 hash function. It works as follows:

1. Append a single 1 bit to the end of the message. This bit serves as a delimiter between the message and the padding.
2. Append a sequence of zeros, so that the total length of the message plus padding is congruent to 448 modulo 512. This ensures that there is room for a 64-bit length value to be appended in the next step.
3. Append a 64-bit length value, representing the length of the original message in bits. The length value is represented as two 32-bit words, with the most significant word first.

Generation of the artificial message blocks - For each of the 80 rounds of the compression function, the message schedule is updated by computing a new value for each of the remaining words in the schedule, using the following formula:

W[i] = W[i-3] W[i-8] W[i-14] W[i-16], for 16 ≤ i ≤ 79.

This formula combines the four most recent words in the message schedule to produce a new value for each word.

Initializing Chaining variables - The initialization of the chaining variables in RIPEMD-160 is the process of setting the initial values of the five 32-bit chaining variables used in the compression function. The chaining variables are updated during each round of the compression function, and their final values are used as the output of the hash function.

The initial values of the chaining variables in RIPEMD-160 are fixed values that are defined in the specification of the hash function. These values are as follows:

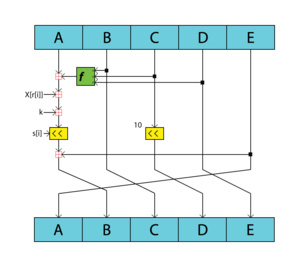
* A = 0x67452301
* B = 0xefcdab89
* C = 0x98badcfe
* D = 0x10325476

1. E = 0xc3d2e1f0

**Message block processing**

1. The compression function of RIPEMD-160 has 80 rounds. These rounds are performed on the 160-bit chaining variables in a sequential manner, with each round processing 5 words of the input block and updating the values of the chaining variables. The 80 rounds are divided into 5 rounds of 16 steps each, with each set of 16 steps using a different non-linear function to update the intermediate state of the hash computation.

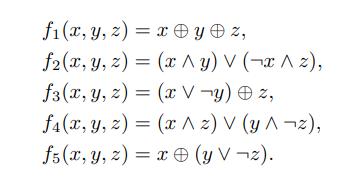
Each round uses a different additive constant, which is added to one of the words in the message block. The output of the compression function is the final value of the 160-bit chaining variables after all 80 rounds have been completed. The chaining variables are then used as the input for the next block of the message to be hashed, or as the final hash output in the case of the last block.



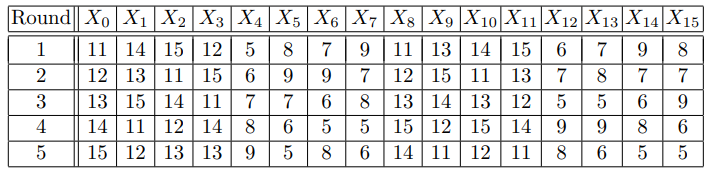
1. The operation in the compression function can be defined as-

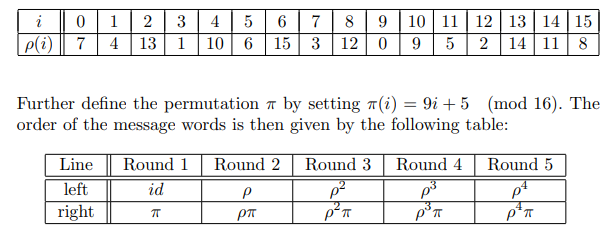
Where, << denotes left shift.

1. The Boolean function F(X,Y,Z) also changes for each round



1. The shift that occurs in the compression function also changes for each iteration according to the table below

****

1. Ordering of the message blocks

**Output:**

The final value of the chaining variables obtained after 80 operations performed on it is used as the output of the hash function. The 160-bit output is typically represented as a sequence of 40 hexadecimal digits, with each digit representing 4 bits of the output. The resulting hash value is deterministic, meaning that the same input will always produce the same output. Any changes to the input message will result in a completely different hash value.

**Applications:**

1. Cryptography: RIPEMD-160 is commonly used for digital signatures, message authentication codes (MACs), and other cryptographic protocols. Its secure hash function properties make it an ideal choice for these applications.
2. Password Storage: RIPEMD-160 is often used to securely store user passwords in databases. Passwords are hashed using RIPEMD-160 and stored in the database, making it difficult for attackers to retrieve the original passwords.
3. Data Integrity: RIPEMD-160 can be used to ensure the integrity of data during transmission or storage. By computing a hash value of the data using RIPEMD-160, any modifications or tampering with the data can be detected.
4. Digital Forensics: RIPEMD-160 can be used to verify the integrity of digital evidence, including forensic images of hard drives or other storage devices. By computing a hash value of the original data and comparing it to the hash value of the forensic image, investigators can determine whether the image has been tampered with or modified.

**Conclusion:**

In conclusion, the RIPEMD family of hash functions was developed as a response to the known weaknesses of the MD4 hash function. The first RIPEMD algorithm was released in 1996, and since then, several improved versions have been developed, including RIPEMD-128, RIPEMD-160, RIPEMD-256, and RIPEMD-320. RIPEMD-160 is the most commonly used hash function in the RIPEMD family.

While no hash function can be completely immune to attacks, the RIPEMD family of hash functions is considered to be highly secure and reliable, with a proven track record of resisting cryptanalytic attacks. Overall, the RIPEMD family of hash functions represents a significant advance in the field of cryptography and continues to be an important area of research and development for securing digital information.

**References:**

* <https://en.wikipedia.org/wiki/RIPEMD>
* <https://homes.esat.kuleuven.be/~bosselae/ripemd160/pdf/AB-9601/AB-9601.pdf>
* <https://cacr.uwaterloo.ca/hac/about/chap9.pdf>
* <https://homes.esat.kuleuven.be/~bosselae/ripemd160.html>
* <https://www.geeksforgeeks.org/ripemd-hash-function/>