Logotipo, nombre de la empresa

Descripción generada automáticamente

**Algoritmos y Estructuras de Datos II**

**Grupo 1**

**Tarea Extraclase 4**

**Profesor:**

**Luis Diego Noguera Mena**

**Integrantes:**

**Johnny Agüero Sandí**

**Ignacio Morales Chang**

**10/6/2021**

Contents

[Introduction 3](#_Toc74207013)

[Problem Analysis 3](#_Toc74207014)

[SHA1 3](#_Toc74207015)

[References 5](#_Toc74207016)

# 

# Introduction

A hash function is a mathematical function that converts a numerical input value of arbitrary length into a compressed numerical value of fixed length. These values returned are called message digest. A hash function is desired to have the following properties:

* Pre-image resistance: it should be computationally hard to reverse a hash function.
* Second pre-image resistance: given an input and its hash, it should be hard to find a different input with the same hash.
* Collision resistance: it should be hard to find to find two different inputs of any length that result in the same hash, also referred as collision free hash function.

Some popular hash functions: Message Digest, Secure Hash Function, RIPEMD, and Whirlpool. In this document one of the Message Digest and one of the Secure Hash Function will be described.

Hash functions have two direct applications bases on its cryptographic properties: password storage and Data integrity check.

# Problem Analysis

In this evaluation, the main objective is that the students comprehend what are hash functions and some of their appliances. To resolve this, investigation must be done by the students. One of the functions to be found of hash functions is data integrity check. Data integrity check is important in the modern world since many important files are shared in the digital form. Data integrity check protects the validity and accuracy of data. It also guarantees the searchability a traceability of your data to its original source.

# SHA1

SHA1 is a Secure Hash Algorithm for computing a condensed representation of a message or data file. A 160-bit output, called message digest, is produced by SHA-1 when a message of any length < 2^64 bits is input, then, the message digest can be input to a signature algorithm that generates or verifies the signature for the message. Doing this is more efficient when compared to signing the message instead of the message digest, this is due to the much smaller size of the message digest. This same algorithm must be used by the verifier of a digital signature as was used by the creator of the digital signature. Any change to the message in transit will most likely result in a different message digest, resulting on failing to verify the signature.

Definitions:

* Hex Digit: Representation of a 4-bit string composed of elements of the set {0, 1, …, 0, A, B, …, F}.
* To convert a word (32-bit string), each 4-bit string is converted to its hex equivalent.
* An integer between 0 and 2^32-1 inclusive may be represented as a word. Example: 291 = 2^8 + 2^5 + 2^1 + 2^0 = 256 + 32 + 2 + 1 ---> hex = 00000123. If, for example, the integer z, where 0 <= z < 2^64 then z = (2^32)x + y being z < 2^32 and 0 <= y < 2^32. “z” can be represented as the pair of words (X, Y).
* Block = 512-bit string. It may be represented by a sequence of 16 words.

Message Padding:

SHA-1 is used to compute a message digest for a message or data file that is provided as input. This input should be considered as a bit string, where the length is the number of bits in the message. When the length is a multiple of 8, it can be represented as its hex equivalent. The purpose of number padding is to make the total length of a padded message a multiple of 512. A “1” followed by m “0”s followed by a 64-bit integer are appended to the end of the message to produce a padded message of length 512\*n. The 64-bit integer is the length of the original message. Now, the padded message will be processed by SHA-1 as n 512-bit blocks.

Functions used:

f(t;B,C,D) = (B AND C) OR ((NOT B) AND D) ( 0 <= t <= 19)

f(t;B,C,D) = B XOR C XOR D (20 <= t <= 39)

f(t;B,C,D) = (B AND C) OR (B AND D) OR (C AND D) (40 <= t <= 59)

f(t;B,C,D) = B XOR C XOR D (60 <= t <= 79).

Constants used:

K(t) = 5A827999 ( 0 <= t <= 19)

K(t) = 6ED9EBA1 (20 <= t <= 39)

K(t) = 8F1BBCDC (40 <= t <= 59)

K(t) = CA62C1D6 (60 <= t <= 79).

There are different methods to yield the same message digest, we will be covering one of them. In this method the message digest is computed using the message padded. The compution is described using two buffers, each of five 32-bit words and a sequence of eighty 32-bit words. The words on the first 5-word buffer are labeled A, B, C, D, E whereas the words of the second 5-word buffer are labeled H0, H1, H2, H3, H4. The words of the 80-word sequence are labeled W(0), W(1), …, W(79).

The 16-word block M(1), M(2), …, M(n) defined when message padding, are processed in order. Before processing anything, the H’s are initialized as:

H0 = 67452301

H1 = EFCDAB89

H2 = 98BADCFE

H3 = 10325476

H4 = C3D2E1F0.

Now M(1), M(2), ... , M(n) are processed. To process M(i), we proceed as follows:

1. Divide M(i) into 16 words W(0), W(1), ... , W(15), where W(0) is the left-most word.
2. For t = 16 to 79 let

W(t) = S^1(W(t-3) XOR W(t-8) XOR W(t-14) XOR W(t-16)).

1. Let A = H0, B = H1, C = H2, D = H3, E = H4.
2. For t = 0 to 79 do

TEMP = S^5(A) + f(t;B,C,D) + E + W(t) + K(t);

E = D; D = C; C = S^30(B); B = A; A = TEMP;

1. Let H0 = H0 + A, H1 = H1 + B, H2 = H2 + C, H3 = H3 + D, H4 = H4 + E.

After being processed, the message digest is the 160-bit string represented by:

H0 H1 H2 H3 H4

# References

Tutorials Point. (s.f). Cryptography Hash functions. Recuperado de: <https://www.tutorialspoint.com/cryptography/cryptography_hash_functions.htm>