Collision-resistant hash function based on composition of functions

René Ndoundam, Juvet Karnel Sadie

CARI'10

Collision-resistant hash function based on composition of functions

René Ndoundam [a,b], **Juvet Karnel Sadié** [a,b]

[a]Université de Yaoundé I, UMI 209, UMMISCO, B.P. 337 Yaoundé, Cameroun
[b]Université de Yaoundé I, LIRIMA, Equipe GRIMCAPE, Faculté des Sciences, Département d'Informatique, B.P. 812 Yaoundé, Cameroun
E.mail : ndoundam@gmail.com, ndoundam@yahoo.com , karnel12@yahoo.fr

RÉSUMÉ. Une fonction de hachage cryptographique est une procédure déterministe qui compresse un ensemble de données numériques de taille arbitraire en une chaîne de bits de taille fixe. Il existe plusieurs fonctions de hachage : MD5, HAVAL, SHA... Il a été reporté que ces fonctions de hachage ne sont pas sécurisées. Notre travail a consisté à la construction d'une nouvelle fonction de hachage basée sur une composition de fonctions. Cette construction utilise la NP-completude des tables de contingence de dimension 3 et une relaxation de la contrainte selon laquelle une fonction de hachage doit être aussi une fonction de compression.

ABSTRACT. A cryptographic hash function is a deterministic procedure that compresses an arbitrary block of numerical data and returns a fixed-size bit string. There exists many hash functions: MD5, HAVAL, SHA, ... It was reported that these hash functions are no longer secure. Our work is focused on the construction of a new hash function based on composition of functions. The construction used the NP-completeness of Three-dimensional contingency tables and the relaxation of the constraint that a hash function should also be a compression function.

MOTS-CLÉS : NP-complet, fonction à un sens, Matrice des zéros et des uns, table de contingence de dimension 3, fonction de hachage résistante aux collisions.

KEYWORDS : NP-complete, One-way function, Matrix of zeros and ones, Three-dimensional contingency table, Collision-resistant hash function.

1. Introduction

A cryptographic hash function is a deterministic procedure that compresses an arbitrary block of data and returns fixed-size bit string, the hash value (message digest or digest). An accidental or intentional change of the data will almost certainly change the hash value. Hash functions are used to verify the integrity of data or data signature.
Let us suppose that $h : X \to Y$ is a hash function without key. The function h is secured if the following three problems are difficult to solve.

Problem 1 : First Preimage attack
Instance : a function $h : X \to Y$ and an image $y \in Y$
Query : $x \in X$ such that $h(x) = y$

We suppose that a possible hash y is given, we want to know if there exists x such that $h(x) = y$. If we can solve *First Preimage attack*, then (x, y) is a valid pair. A hash function for which *First Preimage attack* can't be solved efficiently is sometimes called *Preimage resistant*.

Problem 2 : Second Preimage attack
Instance : a function $h : X \to Y$ and an element $x_1 \in X$
Query : $x_2 \in X$ such that $x_1 \neq x_2$ and $h(x_1) = h(x_2)$

A message x_1 is given, we want to find a message x_2 such that $x_2 \neq x_1$ and $h(x_1) = h(x_2)$. If this is possible, then $(x_2, h(x_1))$ is a valid pair. A function for which *Second preimage attack* can't be solve efficiently is sometimes called *Second preimage resistant*.

Problem 3 : Collision attack
Instance : a function $h : X \to Y$
Query : $x_1, x_2 \in X$ such that $x_1 \neq x_2$ and $h(x_1) = h(x_2)$
We want to known if it is possible to find two distinct messages x_1 and x_2 such that $h(x_1) = h(x_2)$. A function for which Collision attack can't be solve efficiently is sometimes called *Collision resistant*.

There exists many hash functions : MD4, MD5, SHA-0, SHA-1, RIPEMD, HAVAL. It was reported that such widely hash functions are no longer secured [7, 8, 10, 11, 12, 13, 14]. Thus, new hash functions should be studied. The existing hash functions such as MD4, MD5, SHA-0, SHA-1, RIPEMD, HAVAL... want to achieve two goals at the same time :

a°**)** For any input x, they return a hash of x (of fixed length, this length depends on the hash function choosed)

b°**)** Preimage resistant, Second Preimage resistant and Collision Resistant.

Our contribution is to separate the two goals defined in points a°) and b°). Our hash function H_3 is defined as follows :

– $H_3 = H_2 o H_1$,

– H_2 is a classical hash function such as MD5, SHA-0, SHA-1, RIPEMD, HAVAL, ...

– Given a y, find x such that $H_1(x) = y$ is NP-Complete,

– Find x_1, x_2 such that $x_1 \neq x_2$ and $H_1(x_1) = H_1(x_2)$ is NP-Complete,

– For any input x, the length of $H_1(x)$ is not fixed. This is the main difference with the classical hash functions.

The paper is organized as follows : in Section 2, some preliminaries are presented. Section 3 is devoted to the design of our hash function. Concluding remarks are stated in Section 4.

2. Preliminaries

Let's define some preliminaries useful for the next section.

2.1. Two-dimensional

Data security in two dimension have been studied by many authors [2, 3, 9, 17]. Let m and n be two positive integers, and let $R = (r_1, r_2, \ldots, r_m)$ and $S = (s_1, s_2, \ldots, s_n)$ be non-negative integral vectors. Denoted by $\mathfrak{A}(R, S)$ the set of all $m \times n$ matrices $A = (a_{ij})$ satisfying

$$a_{ij} = 0 \text{ or } 1 \text{ for } i = 1, 2, \ldots, m \text{ and } j = 1, 2, \ldots, n;$$

$$\sum_{j=1}^{n} a_{ij} = r_i \text{ for } i = 1, 2, \ldots, m;$$

$$\sum_{i=1}^{m} a_{ij} = s_j \text{ for } j = 1, 2, \ldots, n.$$

Thus a matrix of 0's and 1's belongs to $\mathfrak{A}(R, S)$ provided its row sum vector is R and its column sum vector is S. The set $\mathfrak{A}(R, S)$ was studied by many authors [1, 4, 5, 6, 16]. Ryser [16] has defined an *interchange* to be a transformation which replaces the 2×2 submatrix :

$$B_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

of a matrix A of 0's and 1's with the 2×2 submatrix

$$B_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

If the submatrix B_0 (or B_1) lies in rows k, l and columns u, v, then we call the interchange a $(k, l; u, v)$ -*interchange*. An interchange (or any finite sequence of interchanges) does not alter the row and column sum vectors of a matrix. Ryser has shown the following result.

Theorem 1 *[16] Let A and A^* be two m and n matrices composed of 0's and 1's, possessing equal row sum vectors and equal column sum vectors. Then A is transformable into A^* by a finite number of interchanges.*

Let us consider a matrix $A \in \{0,1\}^{n\times n} \in \mathfrak{A}(R,S)$, i.e. its row sum vector R is such that $R \in \{0,1,2,\ldots,n\}^n$ and its column sum vector S is such that $S \in \{0,1,2,\ldots,n\}^n$. We define the function g_1 from $\{0,1\}^{n\times n}$ to $\mathbb{N}^{2n}$ as follows :

$$\begin{aligned} g_1(A) =& R(1)||R(2)||\ldots||R(n)|| \\ & S(1)||S(2)||\ldots||S(n) \end{aligned}$$

where $\|$ denotes the concatenation.

2.2. Three-dimensional

Irving and Jerrum [15] have studied the extension of the problem in three dimension and shown that problems that are solvable in polynomial time in the two-dimensional case become NP-Complete. Suppose that for a given $n \times n \times n$ table D of non-negative integers, and for each i, j, k, the row, column and file sums are denoted by $R(i,k), C(j,k)$ and $F(i,j)$ respectively. In other words :

$$R(i,k) = \sum_{j=1}^{n} D(i,j,k)$$

$$C(j,k) = \sum_{i=1}^{n} D(i,j,k)$$

$$F(i,j) = \sum_{k=1}^{n} D(i,j,k)$$

The following problem is studied by Irving and Jerrum [15] :

Problem 4. Three-dimensional contingency tables (3DCT)
Instance : A positive integer n, and for each i, j, k non-negative integers values $R(i,k)$, $C(j,k)$ and $F(i,j)$
Question : Does there exist an $n \times n \times n$ contingency table X of non-negative integers such that :

$$\sum_{j=1}^{n} X(i,j,k) = R(i,k)$$

$$\sum_{i=1}^{n} X(i,j,k) = C(j,k)$$

$$\sum_{k=1}^{n} X(i,j,k) = F(i,j)$$

for all i, j, k ? Irving and Jerrum show the following result :

Corollary 1 *[15] 3DCT is NP-Complete, even in the special case where all the row, column and file sums are 0 or 1.*

Let us consider a matrix $A \in \mathbb{N}^{n\times n\times n}$ such that its row sum matrix is a matrix R such that $R \in \mathbb{N}^{n\times n}$ (i.e. $R(i,k) \in \mathbb{N}$) , the column sum matrix is a matrix C such that $C \in \mathbb{N}^{n\times n}$ (i.e. $C(j,k) \in \mathbb{N}$) and the file sum matrix is a matrix F such that $F \in \mathbb{N}^{n\times n}$ (i.e. $F(i,j) \in \mathbb{N}$). We define the function g_2 as follows :

$$g_2 \ : \ \mathbb{N}^{n\times n\times n} \longrightarrow \mathbb{N}^{3n^2}$$

$$\begin{aligned} g_2(A) =& R(1,1)||R(1,2)||\ldots||R(n,n)|| \\ & C(1,1)||C(1,2)||\ldots||C(n,n)|| \\ & F(1,1)||F(1,2)||\ldots||F(n,n) \end{aligned}$$

Let us consider the following matrices A and B. We define the element product of matrices A and B as follows :

Definition 1 *Element Product of Matrices of dimension 2*
Let $A, B \in \mathbb{R}^{n_1\times n_2}$, we define the Element product of matrices A and B as follows :

$$C = \ A \,.*\ B \ ; \text{ where } c_{ij} = a_{ij} \times b_{ij} \text{ for i, j such that}$$
$$1 \leq i \leq n_1 \text{ and } 1 \leq j \leq n_2$$

Definition 2 *Element Product of Matrices of dimension 3*
Let $A, B \in \mathbb{R}^{n_1\times n_2\times n_3}$, we define the Element product of matrices A and B as follows :

$$C = \ A \,.*\ B \ ; \text{ where } c_{ijk} = a_{ijk} \times b_{ijk} \text{ for i, j, k such that}$$
$$1 \leq i \leq n_1 \ , \ 1 \leq j \leq n_2 \text{ and } 1 \leq k \leq n_3$$

3. Design of the hash function

Before the construction of our hash function, let us explain the main idea.

3.1. Explanation of the idea by an example

In page 175 of paper [1], Brualdi gives the example of the following five matrices :

$$A_1 = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \ ; \ A_2 = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \ ; \ A_3 = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$
$$A_4 = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \ ; \ A_5 = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

which belong to $\mathfrak{A}(R,S)$ where $R = S = (2,2,1)$. Let us note W the following matrix :

$$W = \begin{pmatrix} 1 & 4 & 9 \\ 2 & 8 & 18 \\ 3 & 12 & 27 \end{pmatrix}$$

Based on the *Element Product of Matrix* defined in the previous subsection, it is easy to verify that :

$$A_1 .* W = \begin{pmatrix} 1 & 4 & 0 \\ 2 & 8 & 0 \\ 0 & 0 & 27 \end{pmatrix} ; A_2 .* W = \begin{pmatrix} 1 & 4 & 0 \\ 2 & 0 & 18 \\ 0 & 12 & 0 \end{pmatrix} ; A_3 .* W = \begin{pmatrix} 1 & 4 & 0 \\ 0 & 8 & 18 \\ 3 & 0 & 0 \end{pmatrix}$$

$$A_4 .* W = \begin{pmatrix} 0 & 4 & 9 \\ 2 & 8 & 0 \\ 3 & 0 & 0 \end{pmatrix} ; A_5 .* W = \begin{pmatrix} 1 & 0 & 9 \\ 2 & 8 & 0 \\ 0 & 12 & 0 \end{pmatrix}$$

By computation, we evaluate that :

$g_1(A_1 .* W) = 5||10||27||3||12||27$ $\quad$ $g_1(A_2 .* W) = 5||20||12||3||16||18$

$g_1(A_3 .* W) = 5||26||3||4||12||18$ $\quad$ $g_1(A_4 .* W) = 13||10||3||5||12||9$

$g_1(A_5 .* W) = 10||10||12||3||20||9$

It is easy to verify that $A_1(2,2) \neq A_2(2,2)$, $A_1(2,3) \neq A_2(2,3)$, $A_1(3,2) \neq A_2(3,2)$ and $A_1(3,3) \neq A_2(3,3)$. All these differences imply that

– the second term of $g_1(A_1 .* W)$ is not equal to the second term of $g_1(A_2 .* W)$,

– the third term of $g_1(A_1 .* W)$ is not equal to the third term of $g_1(A_2 .* W)$,

– the fifth term of $g_1(A_1 .* W)$ is not equal to the fifth term of $g_1(A_2 .* W)$,

– the sixth term of $g_1(A_1 .* W)$ is not equal to the sixth term of $g_1(A_2 .* W)$.

More formally, from the construction of g_1, we can deduce easily that if $A(i,j) \neq B(i,j)$, then :

c°**)** the i-th term of $g_1(A .* W)$ would probably be different from the i-th term of $g_1(B .* W)$,

d°**)** the (n+j)-th term of $g_1(A .* W)$ would probably be different from the (n+j)-th term of $g_1(B .* W)$.

From the fact that $3DCT$ which is related to g_2 (this is an extension of g_1) is NP-Complete, we deduce that :

e°**)** Given y and a matrix W, find a matrix A such that $g_2(A .* W) = y$ is NP-Complete.

Our idea is to build a new hash function H_3 such that $H_3 = H_2 o H_1$ where

– H_2 is a classical hash function such as MD5, SHA-0, SHA-1, RIPEMD, HAVAL,...

– H_1 is a function which exploits the ideas presented in c°) , d°) and e°).

Let us denote $VOnes(n)$ the vector such that $VOnes(n) \in \{0,1\}^n$ and each of its elements is equal to 1. Also, let us denote $MOnes(n)$ the matrix such that $MOnes(n) \in \{0,1\}^{n \times n}$ and each of its elements is equal to 1. in other words :

$$\begin{aligned} VOnes(n)_i &= 1 \text{ where } 1 \leq i \leq n \\ Mones(n)_{i,j} &= 1 \text{ where } 1 \leq i,j \leq n \end{aligned}$$

We denote N_+ the set of strictly positive natural number defined as follows :

$$N_+ = \mathbb{N} \setminus \{0\} = \{1,2,3,4,\ldots\}$$

In the next sub-section, we formalize the observation made in points c°) and d°) and we take into account the NP-Completeness of 3DCT to build a new hash function.

3.2. Construction of the new hash function

For any integers a and p such that $0 \leq a \leq -1 + 2^p$, let us denote $bin(a, p)$ the decomposition of the integer a in base 2 on p positions. In other words :

$$bin(a,p) = x_{p-1}x_{p-2}\dots x_1x_0 \quad \text{and} \quad \sum_{i=0}^{p-1} x_i \times 2^i = a$$

Let us also define the following function :

$$f_0(n) = \lceil log_2(n+1) \rceil$$

$f_0(n)$ represents the number of bits necessary to represent any integer between 0 and n in base 2.

We also define the following functions :

$$f_1(A) = max\left\{ \sum_{j=1}^{n}\sum_{k=1}^{n} A(i,j,k) \ : \ 1 \leq i \leq n \right\},$$

$$f_2(A) = max\left\{ \sum_{i=1}^{n}\sum_{k=1}^{n} A(i,j,k) \ : \ 1 \leq j \leq n \right\},$$

$$f_3(A) = max\left\{ \sum_{i=1}^{n}\sum_{j=1}^{n} A(i,j,k) \ : \ 1 \leq k \leq n \right\},$$

$$f_4(A) = max\left\{ f_1(A), f_2(A), f_3(A) \right\}.$$

$f_4(A)$ represents the maximun of sum of any n consecutive elements of the matrix A belonging to the same row, or to the same column or to the same file. $f_0of_4(A)$ represents the number of bits necessary to represent in base 2 the sum of any n consecutive elements of the matrix A belonging to the same row, or to the same column, or to the same file.

Subsequently, in the aim to be more precise, we redefine g_2 as follows :

$$\begin{aligned} g_2(A) =& bin(R(1,1), f_0of_4(A))||bin(R(1,2), f_0of_4(A))|| \dots ||bin(R(n,n), f_0of_4(A))|| \\ & bin(C(1,1), f_0of_4(A))||bin(C(1,2), f_0of_4(A))|| \dots ||bin(C(n,n), f_0of_4(A))|| \\ & bin(F(1,1), f_of_4(A))||bin(F(1,2), f_0of_4(A))|| \dots ||bin(F(n,n), f_0of_4(A)) \end{aligned}$$

Let us define the following problem :

Problem 5 :

Instance : A positive integer n, two binary strings x and y

two matrices $V, W \in \mathbb{N}^{n\times n\times n}$
Query : Find two matrices $A, B \in \{0,1\}^{n\times n\times n}$
such that :

$$A \neq B$$
$$g_2(A .* V) = g_2(B .* V) = x$$
$$g_2(A .* W) = g_2(B .* W) = y$$

Let us characterize the complexity of $Problem5$.

Proposition 1 *Problem 5 is NP-Complete.*

Proof Idea of Proposition 1 :
We want to show how to transform a solution of 3DCT to a solution of Problem 5 . Without loss of generality, we work in dimension 2. Let us suppose that we want to find a matrix $A \in \{0,1\}^{3\times 3}$ such that :

$$\sum_{j=1}^{3} A(i,j) = R(i) \tag{1a}$$

$$\sum_{i=1}^{3} A(i,j) = C(j) \tag{1b}$$

where $R = (3,2,1)$ and $C = (2,3,1)$.
It is easy to see that the determination of the matrix $A \in \{0,1\}^{3\times 3}$ which verifies Equations (1) is also equivalent to determining the matrix $B \in \{0,1\}^{6\times 6}$ such that :

$$\sum_{j=1}^{6} B(i,j) = Rd(i) \tag{2a}$$

$$\sum_{i=1}^{6} B(i,j) = Cd(j) \tag{2b}$$

where $Rd = (3,2,1,3,2,1)$ and $Cd = (2,3,1,2,3,1)$.
Remark 1 : Rd (respectively Cd) is a duplication of R (respectively C).
It is easy to see that from the matrix :

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

which verifies Equations (1), we can associate the two following matrices B_2 and B_3

$$B_2 = \begin{pmatrix} A & 0_{3\times 3} \\ 0_{3\times 3} & A \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

$$B_3 = \begin{pmatrix} 0_{3\times 3} & A \\ A & 0_{3\times 3} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

which verify Equations (2). This is the idea of the transformation which associates to one solution of the problem defined in Equations (1) two distinct solutions of the problem defined in Equations (2).
Before the proof, let us introduce the function duplic (which is pseudo-duplication) of x. We note :

$$x = x(1)x(2)\dots x(p) \tag{3}$$

where $x(i) \in \{0,1\}$ and $p = 3 \times n^2 \times \lceil log_2(n+1) \rceil$. We define the function t as follows :

$$t(i,n) = i \times n \times \lceil log_2(n+1) \rceil \; ; \;\; 0 \leq i \leq 3n. \tag{4}$$

The function duplic is defined as follows :

$$\begin{aligned} duplic(x,n) = & dcopy(x,1,n)||dcopy(x,2,n)||\dots||dcopy(x,n,n)|| \\ & dcopy(x,1,n)||dcopy(x,2,n)||\dots||dcopy(x,n,n)|| \\ & dcopy(x,n+1,n)||dcopy(x,n+2,n)||\dots||dcopy(x,2n,n)|| \\ & dcopy(x,n+1,n)||dcopy(x,n+2,n)||\dots||dcopy(x,2n,n)|| \\ & dcopy(x,2n+1,n)||dcopy(x,2n+2,n)||\dots||dcopy(x,3n,n)|| \\ & dcopy(x,2n+1,n)||dcopy(x,2n+2,n)||\dots||dcopy(x,3n,n) \end{aligned}$$

where $dcopy(x,i,n)$ is defined as follows :

$$dcopy(x,i,n) = strcopy(x,i,n)||strcopy(x,i,n)$$

and

$$strcopy(x,i,n) = x(1+t(i-1,n))x(2+t(i-1,n))\dots x(t(i,n))$$

For illustration, $duplic(x,3)$ is defined as follows :

$$\begin{aligned} duplic(x,3) = & x(1)\dots x(6)x(1)\dots x(6)x(7)\dots x(12)x(7)\dots x(12)x(13)\dots x(18)x(13)\dots x(18)|| \\ & x(1)\dots x(6)x(1)\dots x(6)x(7)\dots x(12)x(7)\dots x(12)x(13)\dots x(18)x(13)\dots x(18)|| \\ & x(19)\dots x(24)x(19)\dots x(24)x(25)\dots x(30)x(25)\dots x(30)x(31)\dots x(36)x(31)\dots x(36)|| \\ & x(19)\dots x(24)x(19)\dots x(24)x(25)\dots x(30)x(25)\dots x(30)x(31)\dots x(36)x(31)\dots x(36)|| \\ & x(37)\dots x(42)x(37)\dots x(42)x(43)\dots x(48)x(43)\dots x(48)x(49)\dots x(54)x(49)\dots x(54)|| \\ & x(37)\dots x(42)x(37)\dots x(42)x(43)\dots x(48)x(43)\dots x(48)x(49)\dots x(54)x(49)\dots x(54) \end{aligned}$$

Remark 2 : In the definition of $strcopy(x,i,n)$, the term $x(1+t(i-1,n))x(2+t(i-1,n))\dots x(t(i,n))$ means the concatenation of all the elements between $x(1+t(i-1,n))$ and $x(t(i,n))$. In other words : $x(1+t(i-1,n))x(2+t(i-1,n)\dots x(t(i,n)) = x(1+$

$t(i-1,n))x(2+t(i-1,n)\ldots x(j)\ldots x(-1+t(i,n))x(t(i,n))$ where $1+t(i-1,n) \leq j \leq t(i,n)$.

Proof of Proposition 1 : It suffices to show that $3DCT \leq_m^P Problem\ 5$.

Let us suppose that the procedure *Generalize* solves Problem 5 and we want to show how to build a procedure *Sol3DCT* which solves 3DCT.
The procedure *Sol3DCT* takes as input a binary string x, an integer n and returns as output the matrix A of size n such that $g_2(A) = x$. The procedure *Generalize* takes as input :

– p the dimension of the matrices

– two binary strings x and y

– two matrices V and W

and returns as output :

– two matrices C and D such that :

– $g_2(C\ .*\ V) = g_2(D\ .*\ V) = x$ and $g_2(C\ .*\ W) = g_2(D\ .*\ W) = y$

We show in the procedure below how to use $Generalize$ as a subroutine to solve $Sol3DCT$.

```
Procedure Sol3DCT( n : integer , x : string , var A : matrix ) ;
          V, W , C , D : matrix
          p , i , j , k : integer
          z : string
          begin
1 :           V ⟵ MOnes(2n)
2 :           W ⟵ Mones(2n)
3 :           z ⟵ duplic(x, n)
4 :           p ⟵ 2 × n
5 :           Generalize(p, z, z, V, W, C, D)
6 :           For i = 1 to n do
7 :                 For j = 1 to n do
8 :                       For k = 1 to n do
9 :                             A(i, j, k) ⟵ C(i, j, k) + C(i, j + n, k)
10 :                      Endfor
11 :                Endfor
12 :          Endfor
          end
```

Remark 3 : In the procedure **Sol3DCT**, the matrix A belongs to the set $\{0,1\}^{n\times n\times n}$, whereas the matrices C, D belong to the set $\{0,1\}^{2n\times 2n\times 2n}$.

The string z of the procedure **Sol3DCT** (see instruction 3) is constructed such that $g_2(A) = x$ if and only if the matrices C and D defined in Equations (5) and (6) are the solutions of Problem 5 with the following entries :

– 2n the dimension of the matrices,

– two binary strings z and z,

– two matrices V and W such that $V = Mones(2n)$, $W = Mones(2n)$.

The terms of the matrix C are :

$$\begin{cases} C(i,j,k) = A(i,j,k), & \text{if } 1 \leq i,j,k \leq n; \\ C(i,j+n,k) = 0, & \text{if } 1 \leq i,j,k \leq n; \\ C(i+n,j,k) = 0, & \text{if } 1 \leq i,j,k \leq n; \\ C(i+n,j+n,k) = A(i,j,k), & \text{if } 1 \leq i,j,k \leq n; \\ C(i,j,k+n) = 0, & \text{if } 1 \leq i,j,k \leq n; \\ C(i,j+n,k+n) = A(i,j,k), & \text{if } 1 \leq i,j,k \leq n; \\ C(i+n,j,k+n) = A(i,j,k), & \text{if } 1 \leq i,j,k \leq n; \\ C(i+n,j+n,k+n) = 0, & \text{if } 1 \leq i,j,k \leq n. \end{cases} \tag{5}$$

The terms of the matrix D are :

$$\begin{cases} D(i,j,k) = 0, & \text{if } 1 \leq i,j,k \leq n; \\ D(i,j+n,k) = A(i,j,k), & \text{if } 1 \leq i,j,k \leq n; \\ D(i+n,j,k) = A(i,j,k), & \text{if } 1 \leq i,j,k \leq n; \\ D(i+n,j+n,k) = 0, & \text{if } 1 \leq i,j,k \leq n; \\ D(i,j,k+n) = A(i,j,k), & \text{if } 1 \leq i,j,k \leq n; \\ D(i,j+n,k+n) = 0, & \text{if } 1 \leq i,j,k \leq n; \\ D(i+n,j,k+n) = 0, & \text{if } 1 \leq i,j,k \leq n; \\ D(i+n,j+n,k+n) = A(i,j,k), & \text{if } 1 \leq i,j,k \leq n. \end{cases} \tag{6}$$

■

The main idea of the design of the Collision-resistant hash function H_3 is that :

– the hash function H_3 is the composition of two functions H_1 and H_2,

– the function H_1 is a function for which *Problem 1*, *Problem 2* and *Problem 3* can't be solved efficiently and H_1 is not a compression function.

– H_2 is a hash function such as SHA-256, RIPEMD, or HAVAL, ...

Notation 1 *Let us consider two vectors V_1 and V_2. We say that V_1 is not a linear combination of V_2 and we note V_1 is NLC of V_2 if and only if $\nexists\, \alpha \in \mathbb{R}$ such that $V_1 = \alpha V_2$.*

Two matrices $F, G \in N_+^{n \times n \times n}$ verify the hypotheses (7) if and only if :

$$F \neq G \tag{7a}$$

$$\forall i, j \text{ such that } 1 \leq i, j \leq n, \text{ the vector F(i, j, } *) \text{ is NLC of the vector G(i, j, } *) \tag{7b}$$

$$\forall j, k \text{ such that } 1 \leq j, k \leq n, \text{ the vector F(} *, \text{j, k) is NLC of the vector G(} *, \text{j, k)} \tag{7c}$$

$$\forall i, k \text{ such that } 1 \leq i, k \leq n, \text{ the vector F(i, } *, \text{k) is NLC of the vector G(i, } *, \text{k)} \tag{7d}$$

$$\forall i, j \text{ such that } 1 \leq i, j \leq n, \text{ the vector F(i, j, } *) \text{ is NLC of the vector VOnes(n)} \tag{7e}$$

$$\forall j, k \text{ such that } 1 \leq j, k \leq n, \text{ the vector F(} *, \text{j, k) is NLC of the vector VOnes(n)} \tag{7f}$$

$$\forall i, k \text{ such that } 1 \leq i, k \leq n, \text{ the vector F(i, } *, \text{k) is NLC of the vector VOnes(n)} \tag{7g}$$

$$\forall i, j \text{ such that } 1 \leq i, j \leq n, \text{ the vector G(i, j, } *) \text{ is NLC of the vector VOnes(n)} \tag{7h}$$

$$\forall j, k \text{ such that } 1 \leq j, k \leq n, \text{ the vector G(} *, \text{j, k) is NLC of the vector VOnes(n)} \tag{7i}$$

$$\forall i, k \text{ such that } 1 \leq i, k \leq n, \text{ the vector G(i, } *, \text{k) is NLC of the vector VOnes(n)} \tag{7j}$$

The matrices V and W used as entries in the procedures H_1 and H_3 below verify the hypotheses defined by Equations (7). We note ϵ the empty chain. Let us define the function $VectMat$ which takes as input a vector $Vect$ of size n^3 and returns as output an equivalent matrix A of size $n \times n \times n$.

Procedure $VectMat$ (Vect : Table[1..n^3] of bit ; Var A : Table[1..n, 1..n, 1..n] of bit)
  Var i, j, k , t : integer
  Begin
    $t \leftarrow 1$
    For i = 1 to n do
      For j = 1 to n do
        For k = 1 to n do
          $A(i, j, k) \leftarrow Vect(t)$
          $t \leftarrow t + 1$
        endfor
      endfor
    endfor
  End

The function H_1 is defined as follows :

Function H_1 :
Entry. M_0 the initial message
V : Table[1..n, 1..n,1..n] of integer
W : Table[1..n, 1..n,1..n] of integer
n : an integer
Output. M_2 : an intermediate message
Var i, p : integer
Begin
1. Pad M_0 with one bit equal to 1, followed by a variable number of zero bits and a block of bits encoding the length of M_0 in bits, so that the total length of the padded message is the smallest possible multiple of n^3. Let M_1 denote the padded message
2. Cut M_1 into a sequence of n^3-bits vectors
$B_1, B_2, \ldots, B_i, \ldots B_p$
3. $M_2 \leftarrow \epsilon$
4. For i = 1 to p do
4.1 $VectMat(B_i, A)$
4.2 $M_2 \leftarrow M_2 \parallel g_2(A. * V) \parallel g_2(A. * W)$
Endfor
5. return M_2
End

Our hash function H_3 is defined as the composition of the function H_1 and H_2, where H_2 is a hash function such as SHA-256, RIPEMD, HAVAL... The matrices V and W used as entry in the hash function H_3 must verify the hypotheses defined in Equations (7). To obtain the hash of the message M_0 by H_3, we proceed as follows :

– we obtain the intermediate message M_2 by application of the function H_1 to the message M_0,

– by application of the hash function H_2 to M_2, we build the hash of the initial message.

Formally, the hash function is defined as follows :

Procedure H_3 :
Entry. M_0 the initial message
V : Table[1..n, 1..n,1..n] of integer
W : Table[1..n, 1..n,1..n] of integer
n : an integer
Output. $Result$: the hash of the message M_0
Begin
$M_2 \leftarrow H_1(M_0, V, W, n)$
$Result \leftarrow H_2(M_2)$
End

Comment :

We can represent roughly the function H_1 as follows :

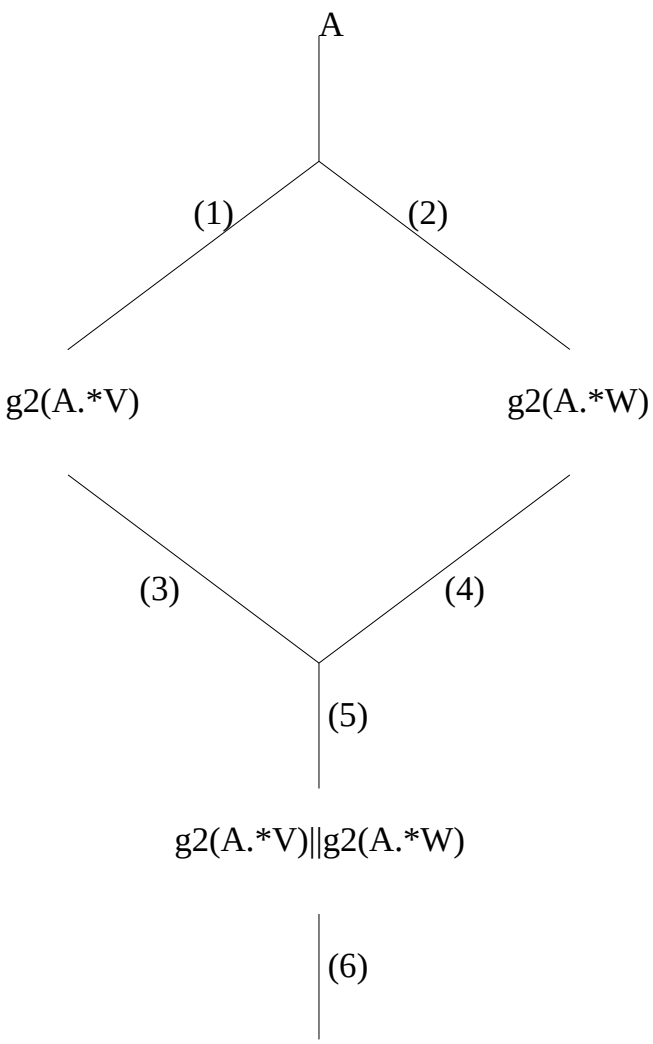

Figure 1. Roughly Representation of the function H_1

In the Figure 1 :

– the aim of the branches (1) and (2) is to make that the Problem 2 and Problem 3 are difficult to solve efficiently for the function H_3

– the aim of the branch (6) is to make sure that Problem 1 is difficult to solve efficiently for the function H_3

During some attacks, an adversary is needed to solve the following problem :

Problem 6 :

Instance : Matrices A, V, W

Binary strings : $g_2(A\ .*\ V)$ and $g_2(A\ .*\ W)$

Query : Find a matrix B such that $A \neq B$ and :

$$g_2(A\ .*\ V) = g_2(B\ .*\ V)$$

$$g_2(A\ .*\ W) = g_2(B\ .*\ W)$$

Based on Problem 5, we deduce that $Problem\ 6$ is NP-Complete.

Second Preimage attack and Collision of the function H_3 are difficult because :

– Problem 5 and Problem 6 are NP-Complete,

– From the fact that V and W verify the hypotheses (7), we deduce that if we take two matrices A and B such that $A \neq B$, then we would probably have $g_2(A\ .*\ V)||g_2(A\ .*\ W) \neq g_2(B\ .*\ V)||g_2(B\ .*\ W)$.

First Preimage attack of the function H_3 is difficult because the 3DCT is NP-Complete. Truncated differential attack of H_1 is possible, but the differential attack of H_3 is difficult because 3DCT is NP-Complete and also Problem 5 is NP-Complete.

4. Numerical Simulation

Let's consider the two messages x1 and x2 :

$$\begin{aligned} x1 = & d131dd02c5e6eec4693d9a0698aff95c \\ & 2fcab58712467eab4004583eb8fb7f89 \\ & 55ad340609f4b30283e488832571415a \\ & 085125e8f7cdc99fd91dbdf280373c5b \\ & d8823e3156348f5bae6dacd436c919c6 \\ & dd53e2b487da03fd02396306d248cda0 \\ & e99f33420f577ee8ce54b67080a80d1e \\ & c69821bcb6a8839396f9652b6ff72a70 \end{aligned}$$

$$\begin{aligned} x2 = & d131dd02c5e6eec4693d9a0698aff95c \\ & 2fcab50712467eab4004583eb8fb7f89 \\ & 55ad340609f4b30283e4888325f1415a \\ & 085125e8f7cdc99fd91dbd7280373c5b \\ & d8823e3156348f5bae6dacd436c919c6 \\ & dd53e23487da03fd02396306d248cda0 \\ & e99f33420f577ee8ce54b67080280d1e \\ & c69821bcb6a8839396f965ab6ff72a70 \end{aligned}$$

We have MD5(x1)=MD5(x2)= EFE502F744768114B58C8523184841F3
after applying our hash function on these messages using $n = 8$, $V[i][j][k] = i+8j+64k$, $W[i][j][k] = 700 - (j + 8 * k + 64 * i)$ for $1 \leq i \leq n$ we obtain :
H_3(x1)= 5fe0e56f9a4ab66a47d73ce660a2c4eb and
H_3(x2) = 620e2f3cfe0afc403c0a8343173526fc.
It follows that $MD5(x1) = MD5(x2)$ whereas $H_3(x1) \neq H_3(x2)$.

5. Conclusion

From a classical hash function H_2, we have built a new hash function H_3 from which First Preimage attack, Second Preimage attack and Collision attack are difficult to solve. Our new hash function is a composition of functions. The construction used the NP-completeness of Three-dimensional contingency tables and the relaxation of the constraint that a hash function should also be a compression function. The complexity of our new hash function increases with regard to the complexity of classical hash functions.

6. Bibliographie

[1] R. A. Brualdi, *Matrices of Zeros and Ones with Fixed Row and Column Sum Vectors*, Linear Algebra and its Applications, **33**, 1980, pp. 159-231.

[2] L. Cox, *Suppression methodology and statistical disclosure control*, J. Amer. Statist. Assoc., 75(1980), pp. 377-385.

[3] I. P. Fellegi, *On the question of statistical confidentiallity*, J. Amer. Statist. Assoc., 67, (1972), pp. 7-18.

[4] D. R. Fulkerson, *An upper bound for the permanent of a fully indecomposable matrix*, Pacific J. Math., **10**, 1960 , pp. 831-836.

[5] D. Gale, *A Theorem on flows in networks*, Pacific J. Math., **7**, 1957 , pp. 1073-1082.

[6] R. M. Haber, *Minimal term rank of a class of (0,1)-matrices*, Canad. J. Math., **15**, 1963 , pp. 188-192.

[7] Hongbo Yu, Xiaoyun Wang, *Multi-Collision Attack on the Compression Functions of MD4 and 3-Pass Haval*, Lecture Notes in Computer Science, **4817** , Springer 2007, pp. 206-226.

[8] Hongbo Yu, Gaoli Wang, Guoyan Zhang, Xiaoyun Wang, *The Second-Preimage Attack on MD4*, Lecture Notes in Computer Science, **3810** , Springer 2005, pp. 1-12.

[9] M. -Y. Kao, D. Gusfield Hongbo Yu, *Efficient detection and protection of information in cross tabulated tables : Linear invariant set*, SIAM J. Disc. Math., 6 (1993), pp. 460-473.

[10] Xiaoyun Wang, Hongbo Yu, Yiqun Lisa Yin, *Efficient Collision Search Attack on SHA-0*, Lecture Notes in Computer Science, **3621** , Springer 2005, pp. 1-16.

[11] Xiaoyun Wang, Yiqun Lisa Yin, Hongbo Yu, *Finding Collisions in the Full SHA-1*, Lecture Notes in Computer Science, **3621**, Springer 2005, pp. 17-36.

[12] Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Cheng, Xiuyuan Yu, *Cryptanalysis of the Hash Functions MD4 and Ripemd*, Lecture Notes in Computer Science, **3494** , Springer 2005, pp. 1-18.

[13] Xiaoyun Wang, Hongbo Yu, *How to Break MD5 and Other Hash Functions*, Lecture Notes in Computer Science, **3494** , Springer 2005, pp. 19-35.

[14] Bert den Boer, Antoon Bosselaers, *Collisions for the compression functions of MD5*, Lecture Notes in Computer Science, **765**, Springer 1994, pp. 293-304.

[15] R. W. Irving, M. R. Jerrum, *Three-Dimensional Statistical Data Security Problems*, SIAM J. Comput., Vol. **23**, No 1, pp. 170-184, 1994.

[16] H. J. Ryser, *Combinatorial properties of matrices of zeros and ones*, Canad. J. Math., Vol. **9**, pp. 371-377, 1957.

[17] G. Sande, *Automated cell suppression to preserve confidentiality of business statistics*, Statist. J. United Nations ECE 2 (1984) pp. 33-41.