International Journal of Enhanced Research in Management & Computer Applications
ISSN: 2319-7471, Vol. 5 Issue 4, April-2016, Impact Factor: 1.544

Depth study and Random balancing function design for One-way Dynamics Hash Functions

Younes ASIMI, Ahmed ASIMI, Zakariae TBATOU, Azidine GUEZZAZ, Yassine SADQI

Information Systems and Vision Laboratory,
Team: Security, Cryptology, Access Control and Modeling,
Department of Mathematics and Computer Sciences, Faculty of Sciences, Ibn Zohr University, Agadir, Morocco

ABSTRACT

A cryptographic hash function is very useful in web applications to prove the authentication of users, servers and messages, in the confidential exchange of a signed data to guarantee their integrity, and in binary sequences cryptographically safe generator. Its security is strongly linked to collisions, and first or second preimages resistance. This article is a depth prelude to propose an one-way hash function of a dynamic length. The objective of this alternative is to strengthen the security of hash functions. We formulate a theoretical study confirming the usefulness to innovate a cryptographic hash function. Specifically, we affirm the impact of our solution against probabilistic attacks, especially with the arrival of quantum computers. To bridge the weaknesses of a classical padding, we propose a new random balancing binary sequences function to strengthen the entropy of hash functions against probabilistic attacks. The results confirm the unpredictable nature of binary sequences regenerated in minimum conditions.

Keywords: Cryptographic hash function, dynamic length, probabilistic attacks, collisions and first or second preimages, random balancing binary sequences functions.

1. INTRODUCTION AND NOTATIONS

The passwords authentication systems are the most used today to ensure the identity of Internet users in a public environment such as web applications. In fact, they give veritable solutions to enterprise security. The robustness of passwords generated by an authentication system is the first step to ensure the confidentiality and integrity of data, and the protection against hackers and other malicious systems. The resistance against attacks (collision, dictionary, brute force,...)[25, 26] is strongly linked to the complexity of passwords chosen by users and condensate generated by a given hash function. It is characterized by a compression function of a fixed and predetermined length. It is therefore vulnerable to attack by an exhaustive search in order to find a collision whose the existence is unavoidable, and to probabilistic attacks. Originally, it reflects its ability to resist against the strong computing power provided by quantum computers [21, 22]. Cryptographic hash functions have known a variant attacks [3]. The length extension attacks were remedied by Coron et al [6]. Also, Joux [5] discovered the multi-collision attack that seeks k internal collisions from k different messages. These collisions give 2k different ways allowing to have a same final chaining variable. This vulnerability affect at bottom the security of any domain extension algorithm whose the internal state length equal to that decadence. The solutions proposed, up to now, have sought to overcome these problems by the theoretical calculation complexity and increased the length of output [5, 13, 19, 20].

Throughout this part, we designate by:

$X = \{x_1, x_2, x_3, ..., x_k\}$	: A finite set such that $h(x_i) \neq h(x_j)$ for all $i \neq j$.
$Y = \{y_1, y_2, y_3, ..., y_{k'}\}$	: A finite set such that $h(y_i) \neq h(y_j)$ for all $i \neq j$.
l_n	: A fixed length of a classical hash functions.
l_m	: A minimum output length of a dynamic hash function.
l_M	: A maximum output length of a dynamic hash function, it equal to $2 * l_m$.
$\|\|$	: Concatenation.

h	: One-way hash function.
f and g	: Our probabilistic functions.
f_b	: Our balancing function.
0^n	: Generation of a null binary sequence of length n.
n	:The possible number of cases of a hash function produces hash values of a given length.
Pr	: Probability.
# T	: Cardinal of a finite set T.
$Supp(x_1x_2x_3x_4...x_n)$	: A finite set of $i \in \{1,...,n\}$ such that $x_i = 1$.
$l_m, l_M - 1$	: It corresponds to $[l_m, l_M - 1] \cap$.
	: Set of natural numbers.
$>$	: Superior.
$\oplus$	: XOR operation.
e *and* ln	: Exponential and natural logarithm functions.
pad and rpad	: Classical and random balancing functions.
$\sqrt{}$	: Square root.
|x|	: The length of a message to hash.

2. RELATED WORK

Cryptographic hash functions grant us a real solution affecting vast areas of applications. They represent the core of many cryptographic systems (symmetric and asymmetric). The robustness of these systems is based mainly on their capacity to resist against different attacks. Firstly, they were created to simplify database management. Side security, they must resist to the research of collisions and preimage [3, 11, 13]. They ensure the integrity of data transmitted on the network and the confidentiality of data stored in a database. They generate from a given input (password, image, text, ...) of any length, an output of a fixed length. The construction of these cryptographic functions is founded on many models: Merkle-Damgarad [3, 7, 8], Sponge [9, 10], EMD [2], ROX [14] and HAIFA [1]. They consist of two principal elements: the compression function and the domain extension algorithm. So far, an iterative algorithm of Merkle-Damgard presents the base of hash functions construction [3].

Hash functions have a nature perfectly deterministic. Hence, the collision problem exists probably forever. Effectively, the security of a hash function based on the difficulty of practically find collisions [11, 12]. In general, they are exist two types of attacks those break the robustness of a hashing function: probabilistic and structural attacks. Today, probabilistic attack begets more concerns of security side than structural attack. Following their importance in cryptographic applications, the experts thought to a new cryptographic hash function concept with a high resistance level against the collision attack [15, 18]. In 2012, NIST published a new cryptographic hash function SHA3 [15]. This algorithm is based on cryptographic primitive Keccak [16] and the sponge model [9, 10]. Also, DRISSI et al [18] published a new variant of the one-way hash functions based on Goppa Codes "OHFGC" and a model presented by MERKLE and DAMGARAD. These two contributions give strength future solutions to cryptographic systems.

3. HASH FUNCTIONS OF A DYNAMIC LENGTH COMPLEXITY

In this section, we present a statistical study of hash functions. We demonstrate the impact of the output length on their ability to withstand collision attacks. We evaluate firstly the hash functions those generate hash values of dynamic lengths, and, we then analyze the obtained outcomes.

A. Hash functions of fixed lengths

Initially, a hash function is defined as an algorithm allowing to associate a hash value of a fixed length to a variable length data and potentially very large.

$$h : \{0,1\}^* \longrightarrow \{0,1\}^{l_n}$$
$$x \longrightarrow h(x)$$

In cryptography, hash functions are used to break the correlation between the input information (passwords) and computed hash value. As example, in a database, the security relies on the ability of such a function to resist against collisions. For an attacker (dictionary attack or brute force), the most important is to find a pre-image, which is linked on the flowing problem: given a hashed $h(x_i)$ with $x_j \in X$, find another $y_j \in Y$, which satisfy the following equation: $h(y_i) = h(x_j)$.

For a hash function that produces hash values of a fixed length l_n bits, the number of possible cases is $n=2^{l_n}$. Hence, the probability that at least one input satisfy $h(y)=h(x)$ for k random input values is: $\Pr(h(y)=h(x)) = k/2^{l_n}$.

Hence, the probability to have at least one matching between X and Y (for each $x_i \in X$ there exists $y_i \in Y$, such that $h(y_i) = h(x_j)$) is equal to:

$$\Pr(h(y_i)=h(x_j)) = 1 - e^{-k^2/2^{l_n}}$$

Therefore, for a hash function produces 2^{l_n} outputs, with a probability $1/2$ of at least one between corresponding X and Y, it must necessarily $k = 2^{l_n/2}\sqrt{\ln(2)}$ inputs.

B. Hash function of dynamic lengths

In the classical case, the definition space of a hash function is surely greater than that calculated for a hash value. This theory implies strongly the existence of a collision at a given iteration. We then push us to design a new hash function of a dynamic length. In our proposal, we energize the hash values space to evolve the resistance of our function against generic attacks. The real length of a given hash value will be between the minimum l_m and the maximum l_M output length. So we define h as follows:

$$h : \{0,1\}^* \to \{0,1\}^l \quad with \quad l \in [l_m, l_M - 1]$$
$$x \to h(x)$$

According to [19], for a hash function that produces a hash length varies between l_m and l_M bits, the possible number of cases is: $2^{l_m}(2^{l_M+1}-1)$.

Proposition 1: Let $y \notin X$, the probability to have $h(y) \in h(X)$ is: $\Pr(h(y) \in h(X)) = k/2^{l_m}(2^{l_M+1}-1)$.

Demonstration: For a given $h(x)$, according to [19], the probability to have got y in Y satisfying $h(y) = h(x)$ is: $\Pr(h(y)=h(x)) = 1/2^{l_m}(2^{l_M+1}-1)$.

So we get: $\Pr(h(y) \neq h(x)) = 1 - 1/2^{l_m}(2^{l_M+1}-1)$.

Therefore: $\Pr(h(y) \notin h(X)) = \left(1 - 1/2^{l_m}(2^{l_M+1}-1)\right)^k$.

Hence we deduce: $\Pr(h(y) \in h(X)) = 1 - \left(1 - 1/2^{l_m}(2^{l_M+1}-1)\right)^k$.

According to Newton's binomial law, we can approximate this probability to:

$$\begin{aligned}\Pr(h(y) \in h(X)) &= 1 - \left(1 - 1/2^{l_m}(2^{l_M+1}-1)\right)^k \\ &\approx 1 - \left(1 - k/2^{l_m}(2^{l_M+1}-1)\right) \\ &= k/2^{l_m}(2^{l_M+1}-1) \qquad (l_m > 0)\end{aligned}$$

Proposition 2: Let $y \in Y$, the probability to have got $x \in X$ such that $h(x) \neq h(y)$ is: $1-{kk'}/{2^{l_m}(2^{l_M+1}-1)}$, and we write: $\Pr(h(x) \neq h(y)) \geq \left(1-{kk'}/{2^{l_m}(2^{l_M+1}-1)}\right)$.

Demonstration: We have:

$$\Pr(h(Y) \cap h(X)=\emptyset)=\Pr(\bigcup_{i=1}^{k'}\{h(y_i)\} \cap h(X)=\emptyset)$$
$$=\prod_{i=1}^{k'}\Pr(\{h(y_i)\} \cap h(X)=\emptyset)$$

Hence, according to proposition 1:

$$\Pr(h(Y) \cap h(X)=\emptyset)=\prod_{i=1}^{k'}\left(1-{1}/{2^{l_m}(2^{l_M+1}-1)}\right)^k$$
$$=\left(\left(1-{1}/{2^{l_m}(2^{l_M+1}-1)}\right)^k\right)^{k'}$$
$$=\left(1-{1}/{2^{l_m}(2^{l_M+1}-1)}\right)^{kk'}$$
$$=1-{kk'}/{2^{l_m}(2^{l_M+1}-1)} \qquad (l_m > 0)$$

Corollary 1: We assume that $k=k'$, and under the same assumptions of proposition 2, we then get:

1) $\Pr(h(y)=h(x)) \geq 1-e^{-k^2/2^{l_m}(2^{l_M+1}-1)}$.
2) $\Pr(h(y)=h(x))={1}/{2} \Leftrightarrow k=\sqrt{2^{l_m}\left(2^{l_M+1}-1\right)\ln(2)}$

Demonstration:

1. $\Pr(h(y)=h(x))=1-\left(1-{k^2}/{2^{l_m}(2^{l_M+1}-1)}\right)$

For $l_m > 0$, we get : $\Pr(h(y)=h(x)) \geq 1-e^{-k^2/2^{l_m}(2^{l_M+1}-1)}$.

2.

$$1-e^{-k^2/2^{l_m}(2^{l_M+1}-1)} = {1}/{2} \Leftrightarrow e^{-k^2/2^{l_m}(2^{l_M+1}-1)} = {1}/{2}$$
$$\Leftrightarrow {k^2}/{2^{l_m}(2^{l_M+1}-1)} = \ln(2)$$
$$\Leftrightarrow k^2 = 2^{l_m}(2^{l_M+1}-1)\ln(2)$$
$$\Leftrightarrow k = \sqrt{2^{l_m}(2^{l_M+1}-1)\ln(2)}$$

Hence, we demonstrate this corollary.

From the corollary 1, we define the probabilistic functions f and g as following:

- The function f, defined as follows: $f :\to 1-e^{-x^2/2^{l_m}(2^{l_M+1}-1)}$ for all $x \geq 0$, allows us to estimate an ability of a dynamic hash function to withstand attacks as that the brute force attack.

The function g, defined as follows: $g :\to 1-e^{-k^2/2^{x}(2^{2x+1}-1)}$ for all $x \geq 0$, allows us to study the impact of outputs space of a dynamic hash function on their unpredictability.

C. Discussion results

So far, the one-way hash functions are designed to have a fixed length. That means, whatever the size of an input message, the hash function produces the same length hash values. This property gene actually the hash values space calculated. In addition, it gives more successful frequency to collision attacks. For an attacker, it is sufficient to create a dictionary of hash values for a given function in order to find a collision, or to try the brute force attack. To discard these concerns, researchers have thought to change the length of the hash values [4, 6, 7, 15, 17]. But in both cases, hash functions produce outputs of the same length. Of course, they also contributed to their complexity by the proposal of new concepts [7, 15, 16, 18]. In this part, we focus on the analysis of two probabilistic functions f and g.

1) Analysis of function f

The aim of function f is to analyze the ability of a dynamic hash function to resist against probabilistic attacks depending on the possible entries k. This function f varies depending of possible entries.

The derivative of f is: $f'(x) = \frac{2x}{2^{l_m}(2^{l_M+1}-1)} e^{-x^2 / 2^{l_m}(2^{l_M+1}-1)} > 0$.

Its variation table is:

Table I: Variation of f depending of possible entries

.

x	0	$+\infty$
f	0 →	1

It is a normal result. It effectively reflects the classic case of a fixed length hash function. It is clearly that when the variable x is approach the infinite, the function f converges to one. In other words, as long an attacker exerts more exhaustive search over a given hash values, more the chance of a collision increases. For this reason, we introduce our vision to vary hash functions lengths.

2) Analysis of function g

The goal of a function g is to break the strict correlation between the number possible entries and the probability to have a collision from a given hash value. Hence, we study our function g depending of the minimum length of outputs. We set the number of possible entries k, and then we evaluate the robustness of a hash function to resist attacks based on probabilistic dynamic output length.

The derivative of g is: $g'(x) = -k^2 \frac{3\ln(2)\left(2^{3x+1}-2^x\right)}{\left(2^{3x+1}-2^x\right)^2} e^{-k^2 / \left(2^{3x+1}-2^x\right)} < 0$, and its variation table is:

Table II: Variation of g depending of possible entries

x	0	$+\infty$
g	$1-e^{-k^2}$ →	0

While a null length hash value does not exist, in our proposal, we recommend a minimum length respond to the requirements of information security [19, 20]. This result explains exactly our hope. This means, as we increase the minimum length of output, the probability of a collision tends to zero. Formally, the space of the range depends on the minimum output length. In other words, our probability function relies on two variables: minimum length and the capacity of the range. So, to find a collision, it must to solve an equation with two unknowns.

These results show the importance of introducing hash functions of a dynamic length. More exactly, to find a probability of $1/2$, the number of possible entries is multiplied by $\sqrt{2^{l_M+1}-1}$. This involved, to perform a dictionary attack on a given minimal length l_m, we must to build l_m dictionaries of lengths between l_m and $l_M = 2l_m$ instead of a single dictionary length l_n. This evolution expresses the impact of the field of definition of the hash values calculated on the complexity of a hash function. So, a dynamic length hash function is a very effective solution against a variant type of attack that exploits the probabilistic laws. With this proposal, the search of collisions will be very difficult and complicated. Whence, this motivation encourages us to propose a dynamic cryptographic hash function.

4. RANDOM BALANCING FUNCTION (RPAD)

Effectively, hash functions handle in input the messages of any lengths. For meet of this requirement, a pretreatment is required on the message to hash. This is thanks to an initial processing on the input messages. It helps to balance the binary sequences to be treated with a proper length of a given compression function.

A. Classical balancing function (pad)

In general, the processed hash functions use conventional transformations [27]. They concatenate to messages to hash a bit is 1, then, they complement them by bits equal to 0 in order to obtain a total length multiple of the compression function length r:

$$pad(M) = M \,||\, 1 \,||\, 0^{(r-1)-|M| \bmod r}.$$

Security perspective, this balancing should be performed by an injective transmission function. The worst, classical padding can have a negative effect on the strength of a hash function [5]. Specifically, the smaller the length of the last binary sequence to hash is smaller, more the length of padding becomes more dominant. This gives more chance for k-multicollisions [5]. It suffices to note here that we can have, for a length of 224 or 512 bits, a final binary sequence that only a few first bits is 1 (two or three bits ...). This security concern pushes us to design a new random balancing function.

B. Random balancing function (rpad)

We inspire this random balancing function of a binary sequence from the obtained results in the synchronous stream cipher generator based on quadratic fields [24]. This theory allows us to balance N binary sequences of any length. The practical and analytical results confirm the impact of this theory on the cryptographic quality of binary sequences regenerated by this regenerator. Our purpose, in this section, is to exploit these results to design a new random balancing function. We focus then on the last sequence to hash. For a compression function of a given length n, this function executes as follows:

- For a message M of any length:
  - We subdivide its to N blocks having the same length r : $M = X_1^r \,||\, X_2^r \,||\, ... \,||\, X_{N-1}^r \,||\, X_N^*$.
  - If the length of the last block is less than the length of the compression function n, we apply our transformation, otherwise, the length is balanced.
- Assume that $|\, X_N^* \,| = m < r$, if the cardinal $\#Supp(X_N^*) > 0$,we define our transformation as followings:

  $rpad(M,r) = X_N^* \,||\, f_b(X_N^*, r-m)$; else $rpad(M,r) = X_N^* \,||\, 1 \,||\, f_b(X_N^* \,||\, 1, r-m-1)$.

  - For a binary vector $X_N = \left(x_{N1}, ..., x_{Nl_N}\right)$, we define our balancing function f_b as followings:

$$\begin{cases} y_{Ni} = x_{Ni} & \text{for all} \quad 1 \le i \le l_N \\ y_{N(l_N+t)} = x_{N(t \bmod ((t+l_N)/l_N))} \oplus x_{Nt} & \text{for all} \quad 1 \le t \le r\text{-}l_N \end{cases}$$

C. Implementation

In a web application, passwords chosen by users generally have different lengths. For this reason, security experts have thought to use the balancing functions. The purpose, after the subdivision of an input message, is to balance the hashed sequences to fixed length required by a given compression function. Rather, in certain situations, the latter sequence can have a different length than that required by a compression function. The following figure 1 shows the balancing result of a hashed sequence (01101010) using a classical transformation.

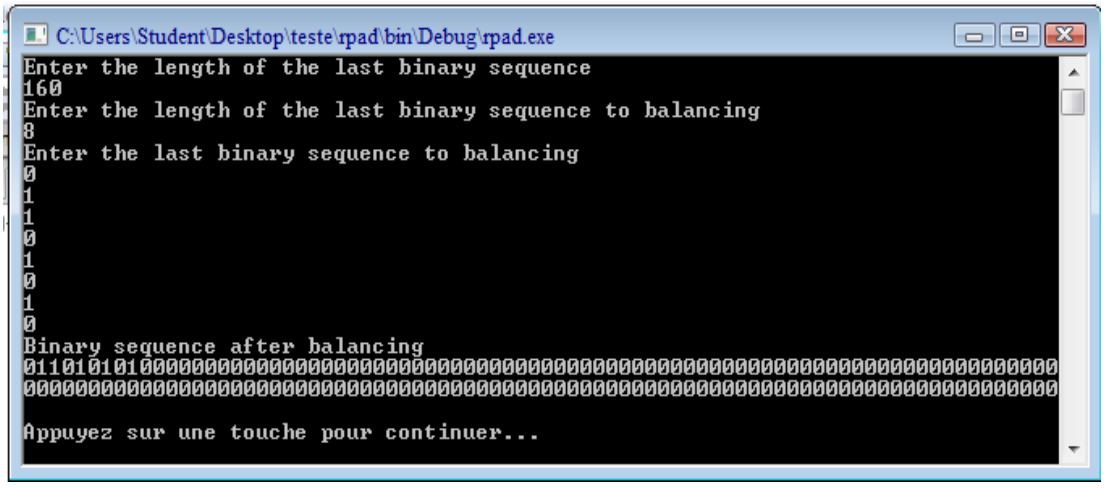

Fig.1: Classical balancing function

An immediate consequence of this result is formulated by Joux [5]. A hash function based on a defined-Damgård Merkle algorithm and a classical padding gives more chance to multi-collisions. In this sense, AJ Menezes et al [23] showed that it is possible for a hash function MDx of a length n and a classical padding of a length l to have 2^{l+1} second preimages. This shows the importance to innovate this random solution able to certify more resistance against attacks by equivalence.

As we mentioned earlier, the possibility to have a last null binary sequence is possible. For this reason, we introduced the second form of a balancing function. The following figure shows the result for a null binary sequence composed of three bits (000).

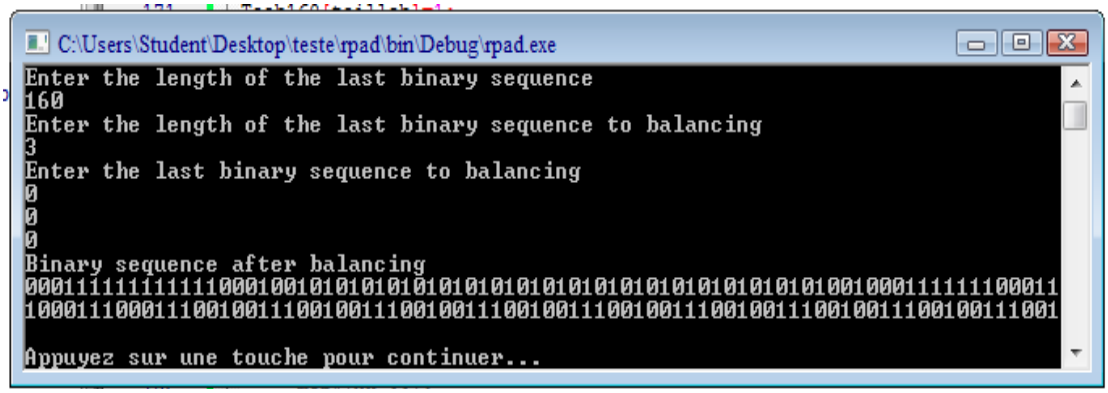

Fig.2: Balancing a null binary sequence

By comparison, even with a null binary system, this last result is greatly evolved compared with the obtained using a classical balancing function. Certainly, the result in figure 1 has a negative effect on the robustness of the hash function. The worst, if the last binary sequence to hash is also null. The possibility to enhance a calculated hash value does not exist anymore. This security concern encouraged us to propose this random balancing function. In this section, we aim to extract practically the internal characteristics of our balancing binary sequences functions. We present the obtained results by balancing the four binary sequences of different lengths (3, 5, 7 and 9 bits). We take these periodic binary sequences (1010…) to assess our balancing function in minimal conditions.

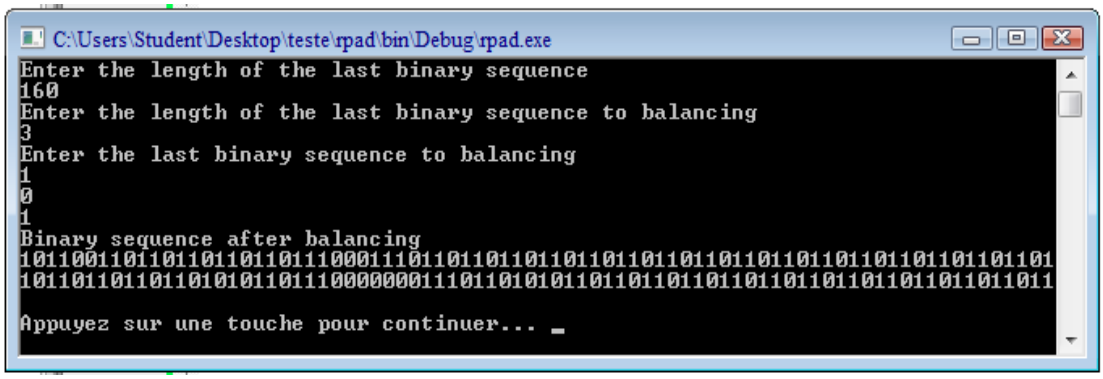

Fig.3: Balancing of a binary sequence of three bits

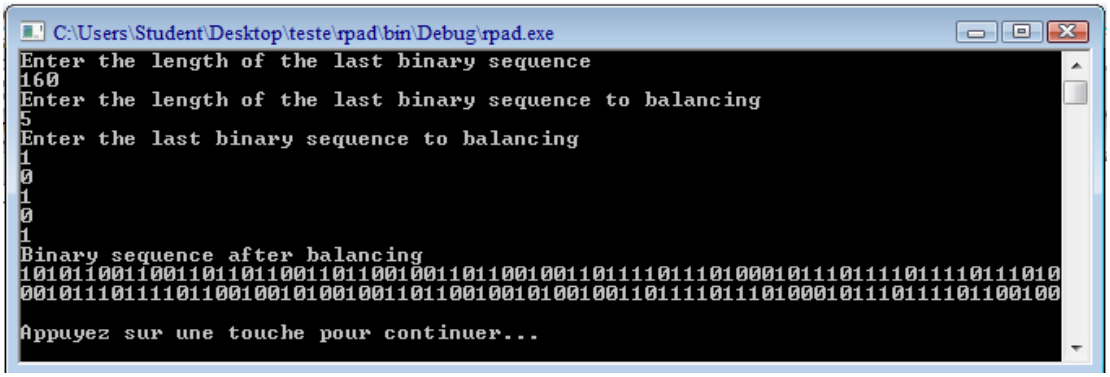

Fig.4: Balancing of a binary sequence of five bits

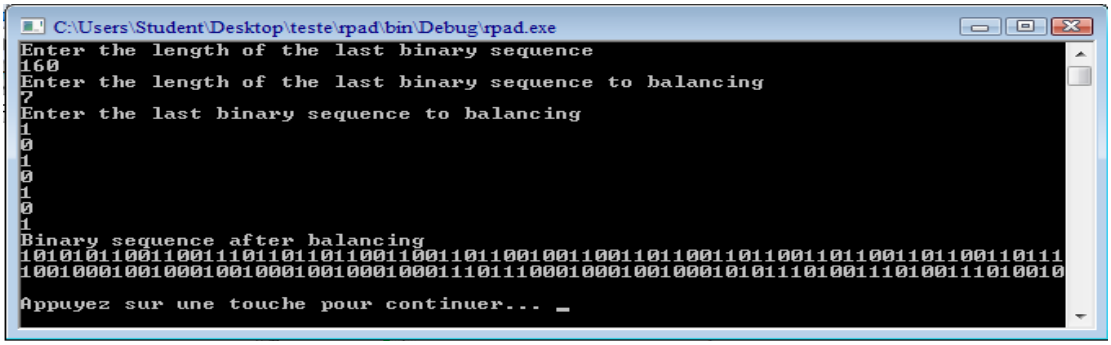

Fig.5: Balancing of a binary sequence of seven bits

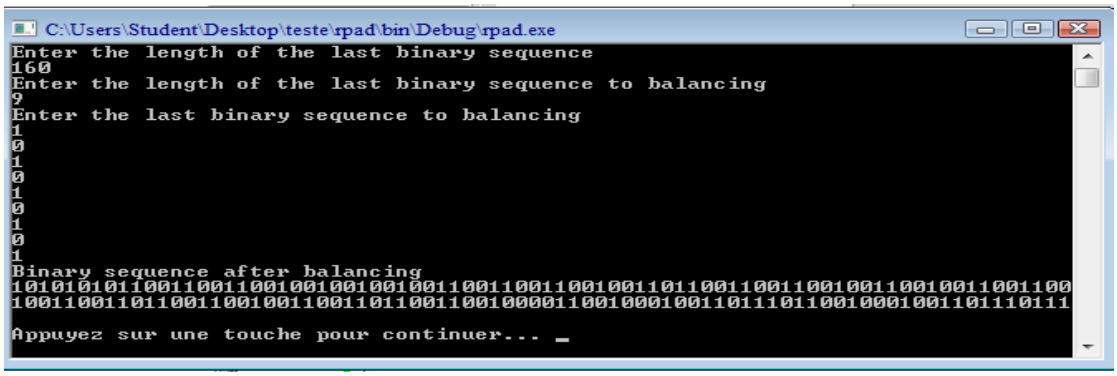

Fig.6: Balancing of a binary sequence of nine bits

Even with periodic linear chains, the results (Fig.2, Fig.3, Fig.4, Fig.5, Fig.6) are effectively random and different between them. This shows the ability of our balancing binary sequences functions to meet the minimum perturbations. It is sufficient to compare with the case of a classical balancing function (Fig.1) to extract the nature of our proposed function. To ensure the random nature of these regenerated binary strings, we introduce in the following paragraph an analytical study confirming these results practically.

D. Analytical study

This analytical study came from the perspective of study behavior and highlights the characteristics of balancing binary sequence functions. It aims to evaluate the distribution of primitive signals generated in minimum conditions. Indeed, we will study the distribution of distances between binary sequences regenerated. This analysis of Hamming distances gives a detailed estimate on the complexity and correlation of binary sequences. More specifically, a class of binary sequences is called random, if the distribution of Hamming distances [19] of this class accumulates near to her half length.
In this section, we aim to estimate the capacity of our balancing binary sequences function to meet the minimum disruption. In this interest, we choose a class of hundred binary sequences $X_N^* \in [1,10,11,100...,1100100]$. The goal is to have binary strings which successively different by one bit. Then we evaluate the hamming distance between all binary strings regenerated by application a classical balancing function and our balancing function.
First, we consider the cryptographic quality of two classes of 160 bits binary sequences length. The following figure shows the results obtained by applying the classical balancing function.

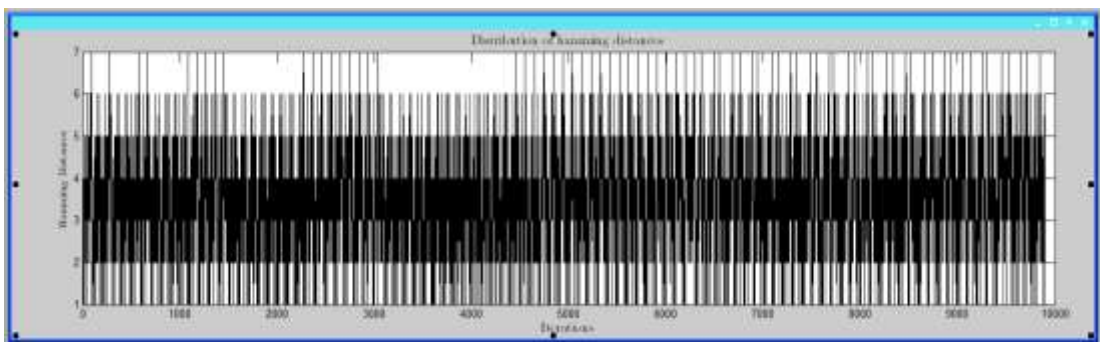

Fig.7: Distribution of Hamming distances obtained by a classical balancing function

The half of length equal to 80 bits. Rather, this result gives hamming distances between 1 and 7. This distribution of distances explains actually the weaknesses of using this classical balancing function. The figure below presents the results of applying our balancing function.

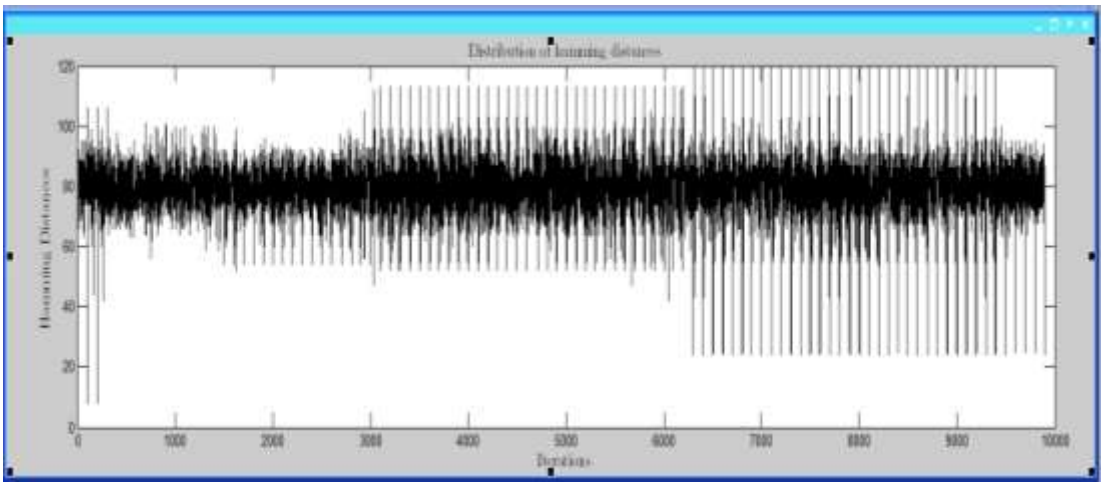

Fig.8: Distribution of hamming distances obtained by our balancing function (160-bit)

For the same class of binary sequences, but this time by applying our function, these results are highly evolved compared to those obtained by the application of the classical balancing function. From Figure 8, we see that most distances accumulate in the vicinity of the half length. This distribution of hamming distances gives a clear view of the random nature of primitive signals regenerated by our balancing binary sequences function. For more credibility to our function, we extend the class length over four times. The aim is to evaluate the impact of the length of binary strings regenerated on their cryptographic quality. The figure follow provides the results for the same class of hundred binary sequences. But this time, we take our balancing binary sequences function to regenerate primitive signals of 512 bits length.

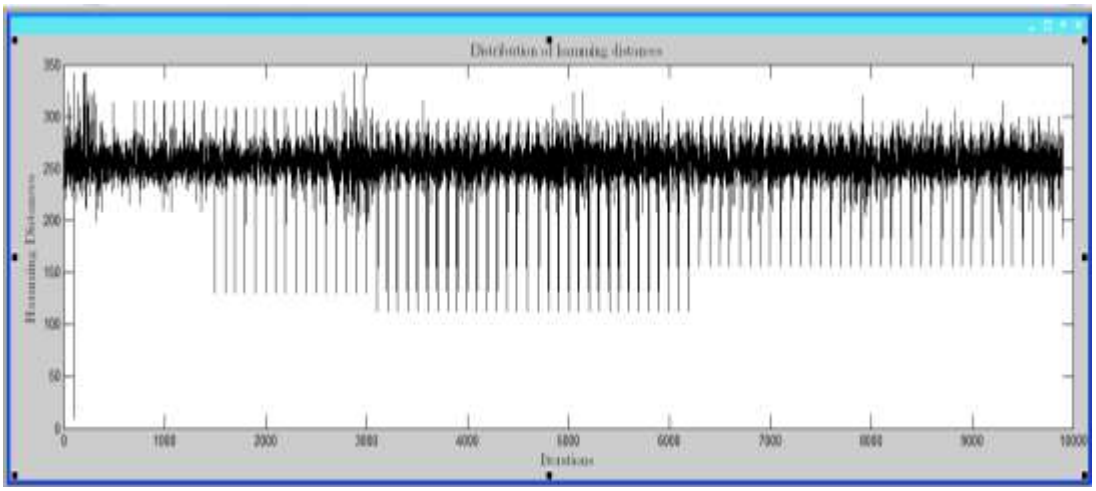

Fig.9: Distribution of hamming distances obtained by our balancing function (512-bit)

Certainly, the results obtained are somewhat different compared to those obtained in Figure 8. Rather, the most relevant notices we inspire from these results, that even with this extension of length, we still have an accumulation of hamming distances calculated in the vicinity of the half length (256 bits). This gives us greater conformity on the random and unpredictable nature of primitive signals regenerated by our balancing binary sequences function. Because, these results are obtained under minimal conditions also for minimal perturbations on the binary sequences of the class.

CONCLUSION AND FUTURE WORK

The construction of a cryptographic hash function required assessments in terms of safety and performance. Those two criteria define the basic principles of all expertise. A hash function should resist originally to the current attacks, also, to have got a long-term vision. In parallel, it provides a level of performance at least equivalent to those available until now. The latter specifies the mass of resources consumed and the speed of data processing by a given function. This work expresses the importance of introducing a new dynamic cryptographic hash function. It presents a detailed theoretical

study indicating determinism and benefits of such hash functions. Also, it implements a new random balancing binary sequences functions rpad. The analytical study confirms the ability of this function to regenerate, in minimum conditions, unpredictable binary sequences. Soon, we interest to design an one-way hash function with a dynamic length based on an algebraic structure.

REFERENCES

[1]. Biham and O. Dunkelman. A Framework for Iterative Hash Functions : HAIFA. In Proceedings of Second NIST Cryptographic HashWorkshop, 2006. http://csrc.nist.gov/groups/ST/hash/second_workshop.html.
[2]. M. Bellare and T. Ristenpart. Multi-Property-Preserving Hash Domain Extension and the EMD Transform. In X. Lai and K. Chen, editors, Advances in Cryptology-ASIACRYPT 2006, volume 4284 of Lecture Notes in Computer Science, pages 299–314. Springer-Verlag.
[3]. E. Andreeva, G. Neven, B. Preneel and T. Shrimpton. Seven-Property-Preserving Iterated Hashing : ROX. In K. Kurosawa, editor, Advances in Cryptology ASIACRYPT 2007, volume 4833 of Lecture Notes in Computer Science, pages 130–146. Springer-Verlag.
[4]. S. Lucks. A Failure-Friendly Design Principle for Hash Functions. In B.K. Roy, editor, Advances in Cryptology – ASIACRYPT 2005, volume 3788 of Lecture Notes in Computer Science, pages 474–494. Springer-Verlag, 2005.
[5]. A. Joux. Multi-collisions in Iterated Hash Functions. Application to Cascaded Constructions. In M. Franklin, editor, Advances in Cryptology – CRYPTO 2004, volume 3152 of Lecture Notes in Computer Science, pages 306–316. Springer-Verlag,2004.
[6]. J.S. Coron, Y Dodis, C Malinaud and P Puniya. Merkle-Damgard Revisited : How to Construct a Hash Function. In V. Shoup, editor, Advances in Cryptology-CRYPTO 2005, volume 3621 of Lecture Notes in Computer Science, pages 430–448. Springer-Verlag.
[7]. R.C. Merkle. One Way Hash Functions and DES. In G. Brassard, editor, Advances in Cryptology – CRYPTO 1989, volume 435 of Lecture Notes in Computer Science, pages 428–446. Springer-Verlag, 1989.
[8]. Damgård. A Design Principle for Hash Functions. In G. Brassard, editor, Advances in Cryptology – CRYPTO 1989, volume 435 of Lecture Notes in Computer Science, pages 416–427. Springer-Verlag, 1989.
[9]. G. Bertoni, J. Daemen, M. Peeters and G. Van Assche. RadioGatun, a Belt-and-Mill Hash Function. In Proceedings of Second NIST Cryptographic HashWorkshop, 2006. http://csrc.nist.gov/groups/ST/hash/second_workshop.html.
[10]. G. Bertoni, J. Daemen, M. Peeters and G. Van Assche. On the Indifferentiability of the Sponge Construction. In N.P. Smart, editor, Advances in Cryptology – EUROCRYPT2008, volume 4965 of Lecture Notes in Computer Science, pages 181–197. Springer-Verlag, 2008.
[11]. John Kelsey and Bruce Schneier. Second preimages on n-bit hash functions for much less than 2n work. In Ronald Cramer, editor, EUROCRYPT, volume 3494 of LNCS, pages 474–490. Springer, 2005.
[12]. John Black, Martin Cochran, and Thomas Shrimpton. On the Impossibility of Highly-Efficient Blockcipher-Based Hash Functions. Advances in Cryptology -- EUROCRYPT '05, Aarhus, Denmark, 2005.
[13]. S. Hirose, Some Plausible Constructions of Double-Block-Length Hash Functions. In: Robshaw, M.J.B. (ed.) FSE 2006, LNCS, vol. 4047, pp. 210-225, Springer, 2006.
[14]. Haichang Gao, Zhongjie Ren, Xiuling Chang, Xiyang Liu Uwe Aickelin, "A New Graphical Password Scheme Resistant to Shoulder-Surfing ."
[15]. NIST Selects Winner of Secure Hash Algorithm (SHA-3) Competition." http://www.nist.gov/itl/csdsha-100212.cfm.
[16]. Guido B1, Joan D1, Michaël P2, Gilles VA1, "The Keccak reference." http://keccak.noekeon.org/. Version 3.0. January 14, 201.
[17]. P. Wang, Y. Kim, V. Kher, and T. Kwon, " Strengthening password based authentication protocols against online dictionary attacks." Proceed-ings of ACNS?2005, LNCS 3531, pp. 17-32, SpringerVerlag, May 2005.
[18]. Ahmed Drissi, Asimi Ahmed, " One-way Hash function based on Goppa Codes« OHFGC »." Applied Mathematical Sciences 11/2013; 7(143):7097-7104.
[19]. Younes Asimi, Abdallah Amghar, Ahmed Asimi, and Yassine Sadqi, New Random Generator of a Safe Cryptographic Salt Per Session, International Journal of Network Security, Vol.18, No.3, PP.445-453, May 2016.
[20]. Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid, Recommendation for Key Management –Part 1: General (Revision 3), C O M P U T E R S E C U R I T Y, NIST Special Publication 800-57, July 2012.
[21]. D.Augot,M.Finiasz and N.Sendrier. A family of fast syndrome based cryptographic hash functions. In E. Dawson and S. Vandeny eds. My crypt 2005 springer LNCS (2005)
[22]. D.Augot, M.Finiasz. ph Gaborit,S.Manuel and N.Sendrier SHA-3 proposal :FSB submission to the SHA-3 NIST competition 2008
[23]. A.J. Menezes, S.A. Vanstone et P.C. Van Oorschot : Handbook of Applied Cryptography. CRC Press, Inc., Boca Raton, FL, USA, 1996. 32
[24]. Y.ASIMI, A.ASIMI, "A Synchronous Stream Cipher generator based on Quadratic Fields (SSCQF)", in International Journal of Advanced Computer Science and Applications (IJACSA), Vol. 6, PP.151-160, No. 12, 2015.
[25]. Y.ASIMI, A. AMGHAR, A.ASIMI, Y.SADQI, "Strong zero-knowledge Authentication based on the Session Keys (SASK)", in journal International Journal of Network Security & Its Applications (IJNSA), Vol.7, No.1, January 2015.
[26]. Y.ASIMI, A. AMGHAR, A.ASIMI, Y.SADQI, "Strong zero-knowledge Authentication based on the Virtual Passwords (SAVP)", in International Journal of Network Security, Vol.18, No.4, PP.601-616, July 2016.
[27]. Emmanuel Bresson et al: SHABAL – A submission to Advanced Hash Standard. Submission to NIST, 2008. 59, 62, 71, 73, 81, 84,122.