IOSR Journal of Mathematics (IOSR-JM)
e-ISSN: 2278-5728. Volume 5, Issue 4 (Jan. - Feb. 2013), PP 47-55
www.iosrjournals.org

A Keyed Hash Function

Joju K.T.[1], Lilly P.L [2]
[1](*Department of Mathematics Prajyoti Niketan College Pudukad ,Kerala,India).*
[2](*Department of Mathematics St.Joseph's College Irinjalakuda, Kerala,India*)

Abstract : *We constructed a hash function by using the idea of cayley graph, hash function based on computing a suitable matrix product in groups of the form $SL_2(F_2^n)$. We found collision between palindrome bit strings of length 2n+2 for the new construction. Here we reinforce the hash function by adding key to it and we claim that it will resists palindrome collision.*

Keywords *- Cayley graph, Group, Hash function, Irreducible polynomial, Message Authentication Code, palindrome collision.*

I. Introduction

I.I Cryptographic Hash Functions and MACs

Hash functions[1] are functions that compress an input of arbitrary length into fixed number of output bits, the hash result. If such a function satisfies additional requirements it can be used for cryptographic applications, for example to protect the authenticity of messages sent over an insecure channel. The basic idea is that the hash result provides a unique imprint of a message, and that the protection of a short imprint is easier than the protection of message itself. Related to hash functions are message authentication codes (MACs). These are also functions that compress an input of arbitrary length into a fixed number of output bits, but the computation depends on a secondary input of fixed length, the key. Therefore MACs are also referred to as keyed hash functions. In practical applications the key on which the computation of a MAC depends is kept secret between two communicating parties. For an (unkeyed) hash function, the requirement that the hash result serves as a unique imprint of a message input implies that it should be infeasible to find colliding pairs of messages. In some applications however it may be sufficient that for any given hash result it is infeasible to find another message hashing to same result. Depending on these requirements Praneel [2] provides the following informal definitions for two different types of hash functions.

A one-way hash function is a function h that satisfies the following conditions:

1. The input x can be of arbitrary length and the result h(x) has a fixed length of n bits.
2. Given h and an input x, the computation of h(x) must be easy.
3. The function must be one-way in the sense that given a y in the image of h , it is hard to find a message x such that h(x) = y (preimage-resistance), and given x and h(x) it is hard to find a message $x' \neq x$ such that h(x') = h(x)(second preimage- resistance).

A collision-resistant hash function is a function h that satisfies the following conditions:

1. The input x can be of arbitrary length and the result h(x) has a fixed length of n bits.
2. Given h and an input x, the computation of h(x) must be easy.
3. The function must be collision-resistant: this means that it is hard to find two distinct messages that hash to the same result(i.e., find x and x' with $x \neq x'$ such that h(x) = h(x')).

For a message authentication code, the computation depends on a secondary input, the secret key. The main idea is that an adversary without knowledge of this key should be unable to forge the MAC result for any new message, even when many previous messages and their corresponding MAC results are known. The following informal definition was given by Praneel [2]. A message authentication code or MAC is a function h satisfies the following conditions:

1. The input x can be of arbitrary length and the result h(K,x) has a fixed length of n bits. The function has a secondary input the key K, with a fixed length of k bits.
2. Given h, K and an input x, the computation of h (K, x) must be easy.
3. Given a message x (with unknown K), it must be hard to determine h(K,x).
4. Even when a large set of pairs $\{x_i, h(K, x_i)\}$ is known, it is hard to determine the key K or to compute h(K, x') for any new message $x' \neq x_i$

Definition 2.1 A hash function h: D → R where the domain D = $\{0,1\}^*$, and the range R = $\{0,1\}^n$ for some $n \geq 1$.
Definition 2.2 A MAC is a function h: K X M→ R where the key space K =$\{0,1\}^k$, the message space M = $\{0,1\}^*$, and the range R = $\{0,1\}^n$ for $k,n \geq 1$

Since its introduction at CRYPTO'94 the Tillich-Zemor hash function[3] has kept on appealing cryptographers by its originality, its elegance, its simplicity and its security[4],[5]. The function computation can be parallelized and even the serial version is quite efficient as it only requires XOR, SHIFT and TEST operations. Uniform distribution of the outputs follows from a graph theoretical interpretation of the hash computation.

Joju K.T and Lilly P.L[6] had constructed a hash function by using the idea of cayley graph, hash function based on computing a suitable matrix product in groups of the form $SL_2(F_2^n)$.But it is vulnerable. We found collision for the same(palindrome collision). Here we construct a hash function which is collision resistant. In this paper we present the reinforced version of the hash function by adding key to the same. We claim that it is collision resistant. This paper is organized as follows:

The hash function proposed by us is recalled in section 2. In section 3 we present the new keyed hash function and verify that the keyed hash function resists the palindrome collision.

II. Preliminaries

2.1 Tillich-Zemor Hash function with new generators

.Let n be a positive integer and let p(x) be an irreducible polynomial of degree n over the field F_2 [4]. Let A_0 and A_1 be the following two matrices :

$A_0 = \begin{pmatrix} 0 & 1 \\ 1 & x \end{pmatrix}$, $A_1 = \begin{pmatrix} 1 & x+1 \\ 1 & x \end{pmatrix}$,that have determinant 1. We call these matrices the generators of the hash function. Let v = $b_1 \ldots. b_m \in \{0,1\}^*$, be the bitstring representation of a message . The hash value of v is defined as:

$H(b_1 \ldots. b_m) = A_{b_1} \ldots\ldots A_{b_m} \bmod p(x)$

Let K = $F_2[x]/(p(x)) \approx F_2^n$[7]. The image of the hash function are the matrices of the group SL_2 (K), that is the group of matrices with elements in K and determinant 1.

Let h $(b_1 \ldots. b_m) = A_{b_1} \ldots\ldots A_{b_m}$ be the hash function without modular reduction. i.e h: {0,1}* → $SL_2(F_2[x])$.

2.2. Palindrome Collision

If v = $b_1 \ldots. b_m \in$ M is a bitstring of length m; we denote $v^r = b_m \ldots. b_1$, the reversal of v, ie the reflection of v which interchanges $b_1$withb_m , b_2 with b_{m-1},etc. Bitstring v ∈ M satisfying v = v^r are known as palindrome. In order to have the palindrome collision we will make the following change in the generators[6].

Let $B_0 = A_0 = \begin{pmatrix} 0 & 1 \\ 1 & x \end{pmatrix}$ and $B_1 = A_0 A_1 A_0^{-1} = \begin{pmatrix} 0 & 1 \\ 1 & x+1 \end{pmatrix}$. Define the hash functions H' and h' with new generators as follows:

$H'(b_1 \ldots. b_m) = B_{b_1} \ldots\ldots B_{b_m} \bmod p(x)$ and $h'(b_1 \ldots. b_m) = B_{b_1} \ldots\ldots B_{b_m}$.Then we have the following proposition[6].

Proposition 1**.** Let v , v' ∈ M. Then H(v) = H(v') iff H'(v) = H'(v').

That is , collision for H and H' are equivalent.

In [4] we observed the following property of palindrome messages.

Proposition 2**.** Let v be a palindrome of even length say v = $b_m \ldots. b_1 b_1 \ldots.. b_m$. Let $a_0 \ldots. a_m$ be the following polynomials

$$d_i = \begin{cases} 1, & if\ i = 0 \\ x + b_1 + 1, & if\ i = 1 \\ (x + b_i)d_{i-1} + d_{i-2} & if\ 1 < i \leq m \end{cases}$$

Then h'(v) = $\begin{pmatrix} a^2 & b \\ b & d^2 \end{pmatrix}$ for d = d_m, a = d_{m-1} and for some b ∈ $F_2[x]$

Moreover, h'(0v0)+ h'(1v1) = $\begin{pmatrix} 0 & d^2 \\ d^2 & d^2 \end{pmatrix}$.

From proposition 2 we see that the square root of the lower right entries of $h'(b_1b_1)$;$h'(b_2b_1b_1b_2)$;$h'(b_3b_2b_1b_1b_2b_3)$; etc [8],[9], satisfying a Euclidean algorithm sequence (in reverse order) where each quotient is either x or x+1. These sequences are often called maximal length sequences for the Euclidean algorithm or maximal length Euclidean sequences. Mesirov and sweet[10] showed that, when d ∈ $F_2[x]$ is an irreducible, there exists exactly two polynomials a such that d,a are the first terms of a maximal length Euclidean sequences. In our collision algorithm[6] we apply Mesirov and Sweet's algorithm to the irreducible polynomial d= p(x) .

Proposition 3**. (** Mesirov and Sweet**)** Given any irreducible polynomial p of degree n over F_2, there is a sequence of polynomials $p_n, p_{n-1}, \ldots p_0$ with p_n = p, and ,p_0 = 1 and additionally the degree of p_i is equal to i and

$p_i \equiv p_{i-2} \bmod p_{i-1}$.

Note that once we know a polynomial q = p_{n-1} as mentioned in proposition 3 which matches our given polynomial p_n = p, the Euclidean algorithm will uniquely compute the sequence

p_n, p_{n-1},....,p_1,p_0 = 1.
The quotients $x+\beta_i$(i = 1,....n) occurring in Euclid's algorithm allow us to derive the bits b_i of the palindrome in proposition 2 .We have $p_1=x+b_1+1$ and therefore $b_1 = \beta_1+1$, while $b_i = \beta_i$ for some i >1.That is the bit β_1 has to be inverted. Thus the desired collision will be
$H'(0\beta_n....\beta_1^{-1}\beta_1^{-1}.....\beta_n0) = H'(1\beta_n....\beta_1^{-1}\beta_1^{-1}.....\beta_n1)$ where β_1^{-1}indicates the inversion of β_1.

2.3. To find the maximal length Euclidean sequence:

1. Construct a matrix A∈ $F_2^{(n+1)xn}$ from the n+1 polynomials
   $g_0 \equiv x^0 \bmod p(x)$,

   $g_i \equiv x^{i-1}+x^{2i-1}+x^{2i} \bmod p(x)$ for i= 1,2,......,n

   Placing in the i^{th} row of A the coefficients

   $a_{i,0}$, $a_{i,1}$,......$a_{i,n-1}$ of the polynomial

   $g_i= a_{i,0}+a_{i,1}\, x+......a_{i,n-1}x^{n-1}$.
2. Solve the linear system Au^t = (10......01) where u = (u_1.......u_n).
3. Compute q(x) by multiplying p(x) by $\sum_{i=1}^{n} u_i x^{-i}$ and taking only the non negative powers of x.

2.4. To find Collision for specified parameters

For each choice of F_{2^n} = $F_2[x]/(p(x))$ we obtain two bitstrings v_1 , v_2 ∈ {0,1}*=M with $H'(0v_iv_i^r0)$ = $H'(1v_iv_i^r1)$ for i= 1,2. ie , we obtain two collisions of bitstrings of length 2n+2. v_2 can be obtained by reversing v_1 followed by inverting the first and last bit[10]. By proposition .1 we have $H(0v_iv_i^r0) = H(1v_iv_i^r1)$.From[4] wehave

Collision for $SL_2(F_2[X]/x^2+x+1)$
By collision algorithm we have
$H(ov_1v_1^ro) = H(011110) = \begin{pmatrix} 1 & 1+x \\ 1 & x \end{pmatrix} = H(111111) = H(1v_1v_1^r1)$ and
$H(0v_2v_2^r0) = H(000000) = \begin{pmatrix} 0 & 1 \\ 1 & x \end{pmatrix} = H(100001) = H(1v_2v_2^r1)$

Collision for $SL_2(F_2[X]/x^3+x+1)$
$H(ov_1v_1^ro) = H(00100100) =\begin{pmatrix} x^2 & x \\ 1 & x \end{pmatrix} = H(10100101) = H(1v_1v_1^r1)$ and
$H(0v_2v_2^r0) = H(01111110) =\begin{pmatrix} 1 & 1+x \\ 1 & x \end{pmatrix} = H(11111111) = H(1v_2v_2^r1)$.

Collision for $SL_2(F_2[X]/x^{11}+x^2+1)$[6]
$H(ov_1v_1^ro)$ = H(0,0,1,0,1,1,0,0,0,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0)

$$= \begin{pmatrix} x^4+x^7+x^9+x^{10} & x^2+x^5+x^8+x^{10} \\ 1 & x \end{pmatrix}$$

= H(1,0,1,0,1,1,0,0,0,0,1,1,1,1,0,0,0,0,1,1,0,1,0,1) = $H(1v_1v_1^r1)$

$H(ov_2v_2^ro)$ = H((0,0,1,0,0,0,0,1,1,0,1,1,1,1,0,1,1,0,0,0,0,1,0,0)

$$= \begin{pmatrix} 1+x^2+x^6+x^7+x^9+x^{10} & x+x^2+x^3+x^7+x^8+x^{10} \\ 1 & x \end{pmatrix}$$

= H(1,0,1,0,0,0,0,1,1,0,1,1,1,1,0,1,1,0,0,0,0,1,0,1) = $H(1v_2v_2^r1)$
Similarly we can find collision for the Tillich-Zemor Hash function with new generators by using palindrome bitstrings of length 2n+2 where n is the degree of irreducible polynomial p(x). Calculations are done by using 'SCILAB' (An open source software for numerical computations) .Program is given in appendix 1 or [11]

III. New Hash Function

Let K = {(k_1.....k_k): k_i = 0 or 1}be the key space. Then K contains 2^k elements. We define the new hash function as follows:

$H_1(m_1 \ldots . m_k m_{k+1}$ ....) = $A_{m_1}^{k_1}$......$A_{m_k}^{k_k}\,A_{m_{k+1}}^{k_1}$ … ..mod p(x)

Where $A_{m_i}^{k_i} = \begin{cases} I\,, if\ k_i = 0 \\ A_{m_i} if\ k_i = 1 \end{cases}$

If k_i = 1 ∀ i then H_1 = H. That is the Tillich-Zemor hash function is a particular member of this new keyed hash function[10]. We will show that it is easy to calculate the hash value by using H_1.For that, the representation of the statement, "This new keyed hash function is collision resistant. Hence it is preimage and second preimage resistant." in binary it is the following:

00100010 01010100 01101000 01101001 01110011 00100000 01101110 01100101 01110111 00100000 01101011 01100101 01111001 01100101 01100100 00100000 01101000 01100001 01110011 01101000 00100000 01100110 01110101 01101110 01100011 01110100 01101001 01101111 01101110 00100000 01101001 01110011 00100000 01100011 01101111 01101100 01101100 01101001 01110011 01101001 01101111 01101110 00100000 01110010 01100101 01110011 01101001 01110011 01110100 01100001 01101110 01110100 00101110 01001000 01100101 01101110 01100011 01100101 00100000 01101001 01110100 00100000 01101001 01110011 00100000 01110000 01110010 01100101 01101001 01101101 01100001 01100111 01100101 00100000 01100001 01101110 01100100 00100000 01110011 01100101 01100011 01101111 01101110 01100100 00100000 01110000 01110010 01100101 01101001 01101101 01100001 01100111 01100101 00100000 01110010 01100101 01110011 01101001 01110011 01110100 01100001 01101110 01110100 00101110 00100010

The keyed Tillich-Zemor hash value of this message for the polynomial p(x) = x^{127}+x+1 with the key (k_1k_2.......k_{256}) =
(10001100011000110001100011000110001100011000110001100011000
1000110001100011000110001100011000110001100011000110001100011000110001100011000110001
1000110001100011000110001100011000110001100011000110001100011000110001100011000110001
10001100011) is the following matrix.

column 1

$x+x^{2}+x^{3}+x^{4}+x^{5}+x^{10}+x^{11}+x^{13}+x^{14}+x^{15}+x^{16}+x^{18}+x^{23}+x^{25}+x^{27}+x^{28}+x^{29}+x^{30}+x^{32}+x^{33}+x^{34}+x^{36}+x^{38}+x^{36}+x^{38}+x^{39}+x^{43}+x^{48}+x^{49}+x^{50}+x^{51}+x^{52}+x^{53}+x^{54}+x^{55}+x^{56}+x^{58}+x^{60}+x^{61}+x^{64}+x^{67}+x^{73}+x^{75}+x^{77}+x^{78}+x^{80}+x^{81}+x^{82}+x^{83}+x^{84}+x^{86}+x^{87}+x^{91}+x^{92}+x^{93}+x^{94}+x^{95}+x^{96}+x^{97}+x^{98}+x^{101}+x^{103}+x^{105}+x^{106}+x^{107}+x^{109}+x^{110}+x^{111}+x^{114}+x^{115}+x^{116}+x^{118}+x^{119}+x^{123}+x^{126}$.

$x+x^{5}+x^{6}+x^{7}+x^{8}+x^{11}+x^{12}+x^{13}+x^{15}+x^{17}+x^{18}+x^{19}+x^{20}+x^{21}+x^{22}+x^{24}+x^{27}+x^{28}+x^{29}+x^{31}+x^{37}+x^{40}+x^{42}+x^{44}+x^{48}+x^{50}+x^{53}+x^{54}+x^{56}+x^{58}+x^{61}+x^{63}+x^{64}+x^{65}+x^{66}+x^{71}+x^{74}+x^{75}+x^{77}+x^{78}+x^{79}+x^{80}+x^{83}+x^{84}+x^{85}+x^{87}+x^{88}+x^{90}+x^{92}+x^{97}+x^{99}+x^{102}+x^{104}+x^{105}+x^{106}+x^{107}+x^{108}+x^{111}+x^{117}+x^{119}+x^{122}$.

column 2

$1+x+x^{2}+x^{3}+x^{5}+x^{7}+x^{8}+x^{9}+x^{10}+x^{11}+x^{12}+x^{15}+x^{16}+x^{23}+x^{24}+x^{26}+x^{27}+x^{28}+x^{31}+x^{32}+x^{34}+x^{35}+x^{37}+x^{38}+x^{40}+x^{41}+x^{43}+x^{44}+x^{45}+x^{47}+x^{49}+x^{50}+x^{51}+x^{54}+x^{57}+x^{60}+x^{61}+x^{62}+x^{64}+x^{65}+x^{66}+x^{67}+x^{69}+x^{70}+x^{71}+x^{75}+x^{76}+x^{77}+x^{78}+x^{80}+x^{81}+x^{82}+x^{84}+x^{85}+x^{88}+x^{90}+x^{91}+x^{93}+x^{94}+x^{95}+x^{98}+x^{99}+x^{100}+x^{101}+x^{102}+x^{103}+x^{104}+x^{111}+x^{112}+x^{114}+x^{115}+x^{116}+x^{118}+x^{119}+x^{120}+x^{122}+x^{123}+x^{125}+x^{126}$.

$X^{3}+x^{4}+x^{5}+x^{9}+x^{12}+x^{15}+x^{16}+x^{17}+x^{18}+x^{20}+x^{21}+x^{22}+x^{25}+x^{27}+x^{29}+x^{32}+x^{37}+x^{38}+x^{39}+x^{40}+x^{41}+x^{42}+x^{44}+x^{46}+x^{47}+x^{48}+x^{49}+x^{50}+x^{51}+x^{54}+x^{59}+x^{60}+x^{61}+x^{62}+x^{64}+x^{65}+x^{69}+x^{70}+x^{72}+x^{75}+x^{77}+x^{78}+x^{79}+x^{82}+x^{84}+x^{90}+x^{97}+x^{99}+x^{100}+x^{101}+x^{102}+x^{104}+x^{106}+x^{109}+x^{110}+x^{112}+x^{114}+x^{115}+x^{117}+x^{118}+x^{119}+x^{120}+x^{121}+x^{124.}$

That is the calculation of the hash value using the keyed hash function H_1 is easy.

We claim that this keyed hash function resists the palindrome collision.

In2.4 we found palindrome collisions for $SL_2(F_2[X]/x^2+x+1)$,$SL_2(F_2[X]/x^3+x+1)$ and $SL_2(F_2[X]/x^{11}+x^2+1)$ But it will not be true with respect to the new keyed hash function.For a particular key say,K = (01010101.....01)

Verification of Collision resistance in $SL_2(F_2[X]/x^2+x+1)$

$H_1(ov_1v_1^ro) = H_1(011110) = \begin{pmatrix} x & 1 \\ 0 & 1+x \end{pmatrix}$ and

$H_1(1v_1v_1^r1) = H_1(111111) = \begin{pmatrix} 0 & x \\ x+1 & x \end{pmatrix}$. Hence $H_1(ov_1v_1^ro) \neq H_1(1v_1v_1^r1)$.

Also $H_1(0v_2v_2^r0) = H_1(000000) = \begin{pmatrix} x & x \\ x & 1 \end{pmatrix}$and $H_1(1v_2v_2^r1)$

= $H_1(100001) = \begin{pmatrix} x+1 & 0 \\ 0 & x \end{pmatrix}$ .Hence $H_1(0v_2v_2^r0) \neq H_1(1v_2v_2^r1)$.

Verification of Collision resistance in $SL_2(F_2[X]/x^3+x+1)$

$\mathrm{H}(\mathrm{ov}_1\mathrm{v}_1{}^{\mathrm{r}}\mathrm{o}) = \mathrm{H}(00100100) = \begin{pmatrix} 1+x+x^2 & x+x^2 \\ 1+x+x^2 & x \end{pmatrix}$ and

$\mathrm{H}(10100101) = \mathrm{H}(1\mathrm{v}_1\mathrm{v}_1{}^{\mathrm{r}}1) = \begin{pmatrix} x^2 & 0 \\ 0 & 1+x+x^2 \end{pmatrix}$

Hence $\mathrm{H}_1(\mathrm{ov}_1\mathrm{v}_1{}^{\mathrm{r}}\mathrm{o}) \neq \mathrm{H}_1(1\mathrm{v}_1\mathrm{v}_1{}^{\mathrm{r}}1)$.

Also we have $\mathrm{H}(0\mathrm{v}_2\mathrm{v}_2{}^{\mathrm{r}}0) = \mathrm{H}(01111110) = \begin{pmatrix} 1+x+x^2 & x \\ 0 & x^2 \end{pmatrix}$ and

$\mathrm{H}(1\mathrm{v}_2\mathrm{v}_2{}^{\mathrm{r}}1) = .\mathrm{H}\ (11111111) = \begin{pmatrix} 0 & 1+x+x^2 \\ x^2 & 1+x+x^2 \end{pmatrix}$.

Hence $\mathrm{H}(0\mathrm{v}_2\mathrm{v}_2{}^{\mathrm{r}}0) \neq \mathrm{H}(1\mathrm{v}_2\mathrm{v}_2{}^{\mathrm{r}}1)$

Verification of Collision resistance in $SL_2(F_2[X]/\ x^{11}+x^2+1)$

$\mathrm{H}(\mathrm{ov}_1\mathrm{v}_1{}^{\mathrm{r}}\mathrm{o}) = \mathrm{H}(0,0,1,0,1,1,0,0,0,0,1,1,1,1,0,0,0,0,1,1,0,1,0,0)$

$$= \begin{pmatrix} 1+x^2+x^5+x^6+x^8+x^9+x^{10} & 1+x^2+x^4+x^6+x^7+x^8+x^{10} \\ x^4+x^6+x^7+x^{10} & x+x^3+x^5+x^6+x^7+x^9+x^{10} \end{pmatrix} \text{ and}$$

$\mathrm{H}(1\mathrm{v}_1\mathrm{v}_1{}^{\mathrm{r}}1) = \mathrm{H}(1,0,1,0,1,1,0,0,0,0,1,1,1,1,0,0,0,0,1,1,0,1,0,1)$

$$= \begin{pmatrix} 1+x+x^2+x^3+x^4+x^5+x^6+x^{10} & 1+x^5+x^6+x^7+x^8+x^9 \\ 1+x+x^2+x^3+x^4+x^7+x^8+x^9 & 1+x^4+x^5+x^6 \end{pmatrix}$$

Hence $\mathrm{H}(\mathrm{ov}_1\mathrm{v}_1{}^{\mathrm{r}}\mathrm{o}) \neq \mathrm{H}(1\mathrm{v}_1\mathrm{v}_1{}^{\mathrm{r}}1)$

$\mathrm{H}(\mathrm{ov}_2\mathrm{v}_2{}^{\mathrm{r}}\mathrm{o}) = \mathrm{H}((0,0,1,0,0,0,0,1,1,0,1,1,1,1,0,1,1,0,0,0,0,1,0,0)$

$$= \begin{pmatrix} x^2+x^5+x^6+x^7+x^8+x^9+x^{10} & x+x^3+x^4+x^7+x^{10} \\ x+x^2+x^3+x^4+x^7+x^{10} & 1+x^2+x^4+x^5+x^6+x^9+x^{10} \end{pmatrix}$$

$= \mathrm{H}(1,0,1,0,0,0,0,1,1,0,1,1,1,1,0,1,1,0,0,0,0,1,0,1) = \mathrm{H}(1\mathrm{v}_2\mathrm{v}_2{}^{\mathrm{r}}1) =$

$$\begin{pmatrix} 1+x+x^4+x^5+x^7+x^{10} & x^2+x^4+x^5+x^9 \\ x+x^4+x^6+x^7+x^8+x^9 & x^3+x^5+x^6+x^7 \end{pmatrix}$$

Hence $\mathrm{H}_1(0\mathrm{v}_2\mathrm{v}_2{}^{\mathrm{r}}0) \neq \mathrm{H}_1(1\mathrm{v}_2\mathrm{v}_2{}^{\mathrm{r}}1)$.

Hence the keyed hash function resists the palindrome collision. Calculations are done using “SCILAB”. The program is given in appendix 2.

IV. Conclusion

The advantages of this paper are: 1.The new hash function is not vulnerable for collision. 2. Its execution is easy. 3. It is based on the finite fields .4.Its execution is bitwise. The main limitation of this construction is that it is not up to in use as the SHA-series or such type of hash functions with so much rounds. This keyed hash function can be used in Cloud Computing, Digital Signature and the other relative areas.

Acknowledgements

I am so much grateful to University Grants Commission (Government of India) for giving me the opportunity to do the research under the faculty improvement program (FIP).

Appendix 1

Program without key

```
x= poly (0,'x'); A0=[0,1;1,x]          //matrix 1
x= poly (0,'x'); A1=[1,x+1;1,x]         //matrix 2
H=[input]
if (H(1,1)==0) then
   E=A0
else
   E=A1
end
if (H(1,2)==0) then
```

```
   D=A0
else
   D=A1
end
R=E*D
for i=3:size(H,'*')
   if (H(1,i)==0) then
      R=R*A0
   else
      R=R*A1
   end
   d11=degree(R(1,1))
   c11=coeff(R(1,1))
   for i=1:size(c11,'*')
      c11(1,i)=modulo(c11(1,i),2)
   end
   d12=degree(R(1,2))
   c12=coeff(R(1,2))
   for i=1:size(c12,'*')
      c12(1,i)=modulo(c12(1,i),2)
   end
   d21=degree(R(2,1))
   c21=coeff(R(2,1))
   for i=1:size(c21,'*')
      c21(1,i)=modulo(c21(1,i),2)
   end
   d22=degree(R(2,2))
   c22=coeff(R(2,2))
   for i=1:size(c22,'*')
      c22(1,i)=modulo(c22(1,i),2)
   end
   R(1,1)=inv_coeff(c11,d11)
   R(1,2)=inv_coeff(c12,d12)
   R(2,1)=inv_coeff(c21,d21)
   R(2,2)=inv_coeff(c22,d22)
   p=x^11+x^2+1                    //polynomial
   [res,quo]=pdiv(R,p)
   R(1,1)=res(1,1)
   R(1,2)=res(1,2)
   R(2,1)=res(2,1)
   R(2,2)=res(2,2)
   R=abs(R)
   d11=degree(R(1,1))
   c11=coeff(R(1,1))
   for i=1:size(c11,'*')
      c11(1,i)=modulo(c11(1,i),2)
   end
   d12=degree(R(1,2))
   c12=coeff(R(1,2))
   for i=1:size(c12,'*')
      c12(1,i)=modulo(c12(1,i),2)
   end
   d21=degree(R(2,1))
   c21=coeff(R(2,1))
   for i=1:size(c21,'*')
      c21(1,i)=modulo(c21(1,i),2)
   end
   d22=degree(R(2,2))
   c22=coeff(R(2,2))
```

```
    for i=1:size(c22,'*')
        c22(1,i)=modulo(c22(1,i),2)
    end
    R(1,1)=inv_coeff(c11,d11)
    R(1,2)=inv_coeff(c12,d12)
    R(2,1)=inv_coeff(c21,d21)
    R(2,2)=inv_coeff(c22,d22)
end
disp(R)
H=[]
R=[]
```

Appendix 2

Program with key

```
x=poly(0,'x');A0=[0,1;1,x]                       //matrix 1
x=poly(0,'x');A1=[1,x+1;1,x]                    //matrix 2
I=[1,0;0,1]                                     //Identity Matrix
H1=[0,1,0,0,0,0,1,1,0,1,1,1,1,0,1,1,0,0,0,0,1,0,0,0,1,0,0,0,0,1,1,0,1,1,1,1,0,1,1,0,0,0,0,1,0,0]
k0=[1,0,0,0,1]
j=1
m=1
k=k0
hs=size(H1,'*')
ks=size(k0,'*')
i=ks+1
if(ks<hs) then
     while ( m<= hs-ks)
         k(1,i)=k0(1,j)
         i=i+1
         j=j+1
         if (j>ks) then
             j=1
         end
         m=m+1
     end
 end
if (k(1,1)==0) then
    E=I
else

    if (H1(1,1)==0) then
        E=A0
    else
        E=A1
    end
end
if (k(1,2)==0)
    D=I
else
     if (H1(1,2)==0) then
        D=A0
    else
        D=A1
    end
end
R=E*D
for i=3:size(H1,'*')
     if (k(1,i)==0) then
        R=R*I
    else
        if (H1(1,i)==0) then
          R=R*A0
        else
          R=R*A1
        end
    end
    d11=degree(R(1,1))
    c11=coeff(R(1,1))
    for i=1:size(c11,'*')
        c11(1,i)=modulo(c11(1,i),2)
    end
    d12=degree(R(1,2))
    c12=coeff(R(1,2))
    for i=1:size(c12,'*')
        c12(1,i)=modulo(c12(1,i),2)
    end
    d21=degree(R(2,1))
    c21=coeff(R(2,1))
    for i=1:size(c21,'*')
        c21(1,i)=modulo(c21(1,i),2)
    end
    d22=degree(R(2,2))
    c22=coeff(R(2,2))
    for i=1:size(c22,'*')
        c22(1,i)=modulo(c22(1,i),2)
    end
    R(1,1)=inv_coeff(c11,d11)
    R(1,2)=inv_coeff(c12,d12)
    R(2,1)=inv_coeff(c21,d21)
    R(2,2)=inv_coeff(c22,d22)
    p= x^11+x^2+1                    //polynomial
    [res,quo]=pdiv(R,p)
    R(1,1)=res(1,1)
    R(1,2)=res(1,2)
    R(2,1)=res(2,1)
    R(2,2)=res(2,2)
    R=abs(R)
    d11=degree(R(1,1))
    c11=coeff(R(1,1))
    for i=1:size(c11,'*')
        c11(1,i)=modulo(c11(1,i),2)
    end
    d12=degree(R(1,2))
    c12=coeff(R(1,2))
    for i=1:size(c12,'*')
        c12(1,i)=modulo(c12(1,i),2)
    end
    d21=degree(R(2,1))
    c21=coeff(R(2,1))
    for i=1:size(c21,'*')
        c21(1,i)=modulo(c21(1,i),2)
    end
    d22=degree(R(2,2))
    c22=coeff(R(2,2))
    for i=1:size(c22,'*')
        c22(1,i)=modulo(c22(1,i),2)
    end
    R(1,1)=inv_coeff(c11,d11)
    R(1,2)=inv_coeff(c12,d12)
```

```
   R(2,1)=inv_coeff(c21,d21)
   R(2,2)=inv_coeff(c22,d22)
end
disp(R)
H1=[]
R=[]
```

Reference

[1] Bart Van Rom pay, *Analysis and Design of Cryptographic Hash Functions, MAC algorithms and Block Ciphers*, doctoral dissertation, KU Leuven2004D/2004/7515 ISBN 90-5682-527-5

[2] B. Praneel, *Analysis and Design of Cryptographic Hash Functions*, doctoral dissertation K.U .Leuven Jan. 1993

[3] J.P. Tillich and G. Zemor, *Hashing with* SL_2, *Advances in Cryptology Lecture Notes in Computer Science, vol. 839(1994), Springer-Verlag, pp. 40-49.*

[4] K.T .Joju and P.L. Lilly , Keyed Tillich-Zemor Hash Function. *Research Journal of Pure Algebra-vol 3(1),2013,www .rjpa .info ISSN 2248-9037,page:* 24-32.

[5] K.T .Joju and P.L. Lilly , Alternate form of Hashing with Polynomials. *Proceedings of the International workshop in Cyber Security*, St. Joseph's College, Irinjalakuda pp. 43-45, 2011

[6] K.T .Joju and P.L. Lilly, Tillich-Zemor Hash Function with New Generators and Analysis. *Research Journal of Pure Algebra-2(11),2012 ,www.rjpa.info ISSN 2248-9037,page:338-343.*

[7] John R Durbin, *Modern Algebra*, (John Wiley & Sons 2005).

[8] K.T .Joju and P.L. Lilly, Alternate form of Tillich - Zemor Hash Function which Resist Second Preimage(accepted) *International J. of Math. Sci. & Engg. Appls. (IJMSEA) ISSN 0973-9424, Vol. 7 No. II (March, 2013), pp.*

[9] Markus Grassl, Ivana Ilic, Spyros Magliveras, and Rainer Steinwandt, Cryptanalysis of the Tillich-Zemor hash function, *Journal of Cryptology, Volume 24 Number-1.pp 148-156.*

[10] Jill P. Mesirov and Melvin M. Sweet. Continued Fraction Expansions of Rational Expressions with Irreducible Denominators in Characteristic 2. *Journal of Number Theory, 27 (1987), pp.144-148.*

[11] Wieb Bosma, John Cannon, and Catherine Playoust, The Magma Algebra System I:The User Language. *Journal of Sympolic Computation, 24 (1997), pp.235-265.*