An Integrated ECC-MAC Based on RS Code

Jaydeb Bhaumik and Dipanwita Roy Chowdhury

Indian Institute of Technology
Kharagpur-721302, India

Abstract. This paper presents a message authentication code (MAC) with error-correcting capabilities which can be used for wireless transmission. Also the paper introduces a new nonlinear mixing function '*Nmix*' which is cryptographically strong compared to other existing method and secured against linear, differential and other conventional cryptanalysis. This nonlinear function is used to compute proposed MAC from check symbols of Reed-Solomon (RS) code. Our MAC is shown to be secured even if a fixed pad is used in MAC generation.

Keywords: Error correcting code, Message authentication code, Nonlinear mixing, Reed-Solomon code, ECC-MAC.

1 Introduction

A MAC is a symmetric-key method that is used to protect a message from unauthorized alteration. MAC is a function of the message and the secret key that produces a fixed length value, which serves as the authenticator. It is a tag that is appended to the message or encrypted message by the sender. At the receiver the tag is recomputed from the received message and the secret key which is known to both the sender and receiver. Message is accepted, when computed MAC matches with the received one. If the received message is corrupted either by random noise or by intentional forgeries, then the computed MAC does not match the received one and the message is rejected.

For an unreliable and insecure communication channel between a sender and receiver, Error Correcting Codes (ECC) are used to transmit a MAC by first computing the MAC and then error correcting code is added for transmission. However, this increases overhead. An integrated scheme combining MAC and ECC can be a better solution. This combined scheme is appealing, especially in applications where latency is a concern or resources are limited. Thus, it is interesting to construct MAC which can correct a few errors that may occur during transmission.

Applying error correcting codes to construct MAC was first exploited by Krawczyk [6]. Lam *et al.* have proposed error correcting MAC based on BCH and RS code in [12]. A noise tolerant MAC or NTMAC based on cyclic redundancy check (CRC) called CRC-NTMAC has been proposed in [15]. But the tag itself is not error tolerant. BCH-NTMAC for noisy message authentication has been published in [16]. This MAC has much lower false acceptance probability compared to CRC-NTMAC. Both BCH-NTMAC [16] and MAC proposed by Lam *et al.* [12] are error tolerant. Moreover as Lam *et al.*'s MAC generation scheme is a LFSR based, so it is easy to implement. However, Lam *et al.*'s

M.L. Gavrilova et al. (Eds.): Trans. on Comput. Sci. IV, LNCS 5430, pp. 117–135, 2009.

scheme requires truly random pad of size equal to the size of the hash output for each transmitted message. But in most practical applications, the successive pad r is generated using a pseudorandom generator out of a secret seed shared by the parties. In this case the security of the authentication scheme reduces to the security of the pseudorandom generator.

In this paper, we proposed a new integrated scheme for message authentication and error correction using t-symbol error correcting RS code. Also a new nonlinear mixing function called *Nmix* is proposed which is used to produce MAC from check symbols of RS code. The proposed function reduces the bias of linear approximations exponentially. Also differential property of the proposed key mixing is better than other existing methods. The *Nmix* function improves the security against linear, differential cryptanalysis and hence MAC is secured even if, fixed pad with *Nmix* is used.

The rest of this paper is organized as follows. In the next section, a brief overview of Lam *et al.*'s MAC generation scheme is given. Section 3 discusses the proposed ECC-MAC scheme. The new non linear function *Nmix* which is used to construct the proposed MAC is given in section 4. In section 5, security issues are described. Section 6 describes the performance evaluation of the proposed MAC and finally the paper is concluded in section 7.

2 Overview of Lam *et al.*'s Scheme

Lam *et al.* have proposed two schemes for error correcting MAC [12], one of which is based on BCH code while the other is based on RS code. Both the schemes can provide an integrity check for corrupted data and can correct few transmission errors. They have shown that these MACs are secure. It is suggested that the MAC can be applied to less information- sensitive wireless transmission such as voice and image signals. In this section, the MAC generation scheme [12] based on RS code is reproduced.

Authors have considered a typical communication scenario in which two parties communicate over an unreliable channel with a malicious adversary in the middle. The communicating parties share a secret key unknown to the adversary. For simplicity at the beginning it is assumed that the parties exchange only one message M of length m using that secret key. The secret key is used to draw a hash function randomly from the $H_{m,l}$ family of hash functions and a random pad r of length l over Z_2^l. The message M is a vector over Z_2^m and the hash value is a vector over Z_2^l. The message sender sends M together with a tag $= h_g(M) + r$, and is verified at the receiver by recalculating the tag. The hash function h_g is associated with the polynomial g of degree l. The $H_{m,l}$ family of hash functions are defined as

$$h_g(M) = x^l M(x) \mod g(x)$$

$H_{m,l} = \{h_g | \forall \textit{ valid generator polynomial } g\}$ Therefore, the MAC of the message M in polynomial form is

$$MAC(x) = x^l M(x) \mod g(x) + r(x) \tag{1}$$

where $g(x)$ is a generator polynomial of degree l and $r(x)$ is the random pad in polynomial form. It is assumed that the adversary knows the family of hash

functions, but not the particular value of h_g or the random pad r. In the typical scenario where the parties exchange multiple messages, the hash function h_g can be reused for different messages but for each new message a different random pad will be used for encryption of hash value. Next section introduces our integrated ECC-MAC scheme based on RS code.

3 Proposed Integrated ECC-MAC

In this section, we describe our MAC generation scheme which is based on RS code having t- symbol error correction capability. The message data string is subjected to a padding process both in sender and receiver. The padded string is a bit string of length an integer multiple of n-bit, where n is the length in bits of a symbol of an RS code and the codeword length is $2^n - 1$. RS encoder adds extra $2t$-check symbol with a block of message symbols to correct t-error. Therefore, maximum size of the message block is $2^n - 1 - 2t$ symbols. Our aim is to develop an integrated scheme where this $2t$-check symbol for error correction can also be used to authenticate the message.

The generator polynomial of a t-error correcting RS code [2] is

$$g_k(x) = (x + \alpha^i)(x + \alpha^{i+1})(x + \alpha^{i+2})...(x + \alpha^{i+2t-1})$$

where α is a primitive element of $GF(2^n)$. The generator polynomial depends on primitive element α and the parameter i. In this scheme, both α and i are selected by the secret key k. In GF(2^n), the number of primitive elements is $p = \Phi(2^n - 1)$, where ϕ is Euler's totient function. The number of possible generator polynomials that can be formed using a single primitive element for a t-symbol error-correcting code is $q = 2^n - 1$ [4]. Therefore, total number of possible generator polynomials that can be constructed using all primitive elements is given by

$$N = q \times p = (2^n - 1) \times \Phi(2^n - 1) \tag{2}$$

For $n = 40$ and $t = 4$, the number of possible generator polynomials is given by $N \approx 2^{79}$.

In our scheme, any one of the N generator polynomials is selected by the key to provide security and error correction capability. But the generator polynomial is fixed and known to everybody for a system where RS code is used only for error correction. The selected generator polynomial is used to calculate $2t$-check symbols. Since RS code is linear, the computed bare check symbols are vulnerable to several attacks when it is also used to provide security. Therefore, a sequence is mixed with the check symbols using a non-linear operator. In the proposed algorithm, the secret key of length is four times symbol length. Out of the $4n$ bits, 1^{st}, 2^{nd} n-bit values are used to select primitive element α, the index i respectively and last $2n$ bits are used as an initial seed of the pseudo random sequence generator. In the following subsection, our integrated scheme for message authentication and error correction is described.

3.1 Algorithm

The algorithm of t-symbol error correcting MAC is as follows.

MACGEN: MAC GENeration Algorithm

1. pad the message to make its length equal to an integer multiple of n bits.
2. partition the padded message into symbols of n-bit each.
3. Select a generator polynomial $g_k(x)$
4. Select $r_k \in {Z_2}^{2t.n}$; sequence of dimension $2t.n$ over $GF(2)$, the sequence is a function of secret key.
5. Compute the Check symbols using $P(x) = x^{2t}M(x)\ mod\ g_k(x)$
6. Compute MAC using $c = F(P,\ r_k)$, where F is the nonlinear function.
7. Send Message MAC pair (M, c)

MACVEC: MAC Verification and Error Correction Algorithm
Suppose the received message MAC pair is (M', c').

1. Compute $r_k \in {Z_2}^{2t.n}$; same sequence of dimension $2t.n$ over $GF(2)$.
2. Compute Check symbols using $P' = G(c', r_k)$, where G is the nonlinear function.
3. Compute Syndromes from padded M' and P'.
4. Verify if all syndromes are zero or not. If all are zero, go to step 7.
5. For non-zero syndromes, verify whether the number of errors that have occurred is greater than the error correction capability of the code or not. If yes, reject the message and go to step 8.
6. Correct the errors.
7. Accept the message.
8. Stop.

Next subsection explains the operation of sender and receiver with the help of suitable block diagrams. It also compares the overhead needed for error correcting code (ECC) embedded into the MAC and first MAC then ECC scheme.

3.2 Sender and Receiver

As depicted in Fig.1, sender block diagram consists of an RS encoder, a sequence generator, a nonlinear mixer and an appending block. On the other hand, Fig. 3 shows that the receiver consists of a nonlinear mixer, a sequence generator and an

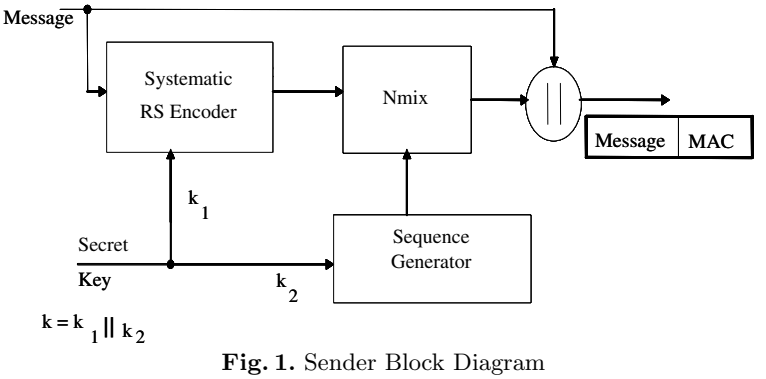

Fig. 1. Sender Block Diagram

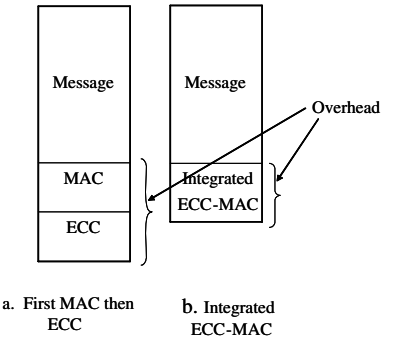

Fig. 2. Overhead for Conventional and Integrated Scheme

Table 1. Comparison of Overhead

# of symbol errors to be corrected	Overhead in Bits			
	First MAC then ECC			Integrated ECC-MAC
	For MAC	For ECC	Total	Total
1	320	80	400	320
2	320	160	480	320
3	320	240	560	320
4	320	320	640	320

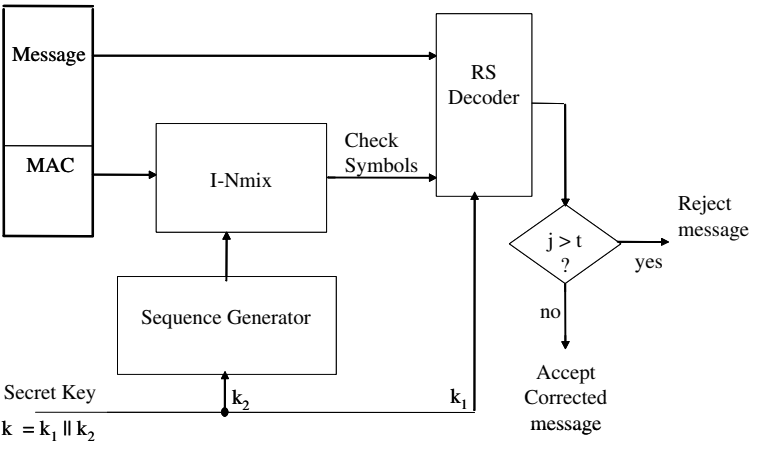

Fig. 3. Receiver Block Diagram

RS decoder. Here j is the number of symbol errors occurred and t is the number of correctable symbol errors. Figure 2 shows that the conventional (first MAC then ECC) scheme needs more over head compared with integrated scheme. A comparison of overhead between the proposed scheme and first MAC then ECC scheme (It is assumed that MAC value computation and error correcting check symbol computation is done in same layer although MAC computation is done in higher layer) is given in Table 1. Here we have considered a MAC length of 320

bits, RS code of symbol length is 40 bits. Table shows that overhead is double for first MAC then ECC scheme compared to integrated ECC-MAC scheme for 4-symbol error correction. Next section introduces new non-linear mixing function *Nmix* which is used to compute the proposed MAC.

4 Nonlinear Mixing Function: *Nmix*

In this section, a new nonlinear mixing function called '*Nmix*' and inverse mixing function '*I-Nmix*' is introduced.

Sequence Mixing (*Nmix*): Assume $X = (x_{n-1}\ x_{n-2}\ ...\ x_0)$ be an $n-$bit data, $R = (r_{n-1}\ r_{n-2}\ ...\ r_0)$ be an $n-$bit sequence and $Y = (y_{n-1}\ y_{n-2}\ ...\ y_0)$ be the $n-$bit output after mixing X with R. Then each output bit is related to the input bits by the following relationship

$$y_i = x_i \oplus r_i \oplus c_{i-1}\ ;\ c_i = \bigoplus_{j=0}^{i} x_j \cdot r_j \oplus x_{i-1}x_i \oplus r_{i-1}r_i \tag{3}$$

where $0 \leq i < n$, $c_{-1} = 0$, $x_{-1} = 0$, $r_{-1} = 0$ and c_i is the carry term propagating from i^{th} bit position to $(i+1)^{th}$ bit position. The end carry c_{n-1} is neglected. We use the notation $Y = (X \dagger R)\ mod\ 2^n = F(X, R)$, where $\dagger$ is the *Nmix* operator. Each y_i is balanced function for all i $(0 \leq i < n)$.

Inverse Sequence Mixing (*I-Nmix*): In inverse sequence mixing, the mixer takes an n-bit data $Y = (y_{n-1}\ y_{n-2}\ \ldots\ y_0)$ and an $n-$bit sequence $R = (r_{n-1}\ r_{n-2}\ \ldots\ r_0)$ as inputs and produces an $n-$bit output $X = (x_{n-1}\ x_{n-2}\ \ldots\ x_0)$. Inverse sequence mixing operation can be defined as

$$x_i = y_i \oplus r_i \oplus d_{i-1}\ ;\ d_i = \bigoplus_{j=0}^{i} x_j \cdot r_j \oplus x_{i-1}x_i \oplus r_{i-1}r_i \tag{4}$$

where $0 \leq i < n$, $d_{-1} = 0$, $x_{-1} = 0$, $r_{-1} = 0$, and d_i is the carry term propagating from i^{th} bit position to $(i+1)^{th}$ bit position. The end carry d_{n-1} is neglected. Here we use the notation $X = (Y * R)\ mod\ 2^n = G(Y, R)$, where $*$ is the *I-Nmix* operator. Each x_i is balanced function for all i $(0 \leq i < n)$. Next, we summarize the results related to properties of *Nmix* function. The proof of the theorems are given in appendix A.

Theorem 1. The function G is the inverse function of F.

Fact 1. For an n-bit Number X, $F(X, X) \neq 0$.

Fact 2. The function F is commutative, i.e. $F(X, R) = F(R, X)$, where X and R are two n-bit numbers.

Theorem 2. The function F does not hold associative law, i.e. $F(F(X, Y), Z) \neq F(X, F(Y, Z))$, where X, Y and Z are three n-bit numbers.

Theorem 3. Output difference (XOR) of the function F is not equal to input difference (XOR) when single input changes, i.e. $F(X, R) \oplus F(Y, R) \neq X \oplus Y$.

Theorem 4. If X, Y and R are the three n-bit numbers then
$F(F(X,R),F(Y,R)) \neq F(X,Y)$.

Theorem 5. Keeping one input fixed of the two input function F, modulo-2 addition of three outputs for three different inputs is not equal to output of the function F when input is modulo-2 addition of three individual inputs.

i.e. $F(X,R) \oplus F(Y,R) \oplus F(Z,R) \neq F((X \oplus Y \oplus Z),R)$.

Theorem 6. If X, Y, Z and R are the four n-bit numbers then
$F(F(F(X,R),F(Y,R)),F(Z,R)) \neq F(F(F(X,Y),Z),R)$

Next we discuss the hardware requirements for implementing a n-bit key mixer based *Nmix* operator. The logic diagram of *Nmix* and *I-Nmix* are given in appendix B.

4.1 Hardware Requirement

Table 2 shows the gate counts for nonlinear functions addition modulo 2^n, *Slash* modulo 2^n [19] and *Nmix* modulo 2^n for forward as well as reverse transformation. As the linear XOR is mostly used key mixing function, so the Table 2 includes XOR also. It is observed that *Nmix* requires more hardware but provides more security which is discussed in section 5.

Table 2. Comparison of Gate Counts

Transformation	Mixing Function	Number of Logic Gates			
		XOR	OR	AND	NOT
Forward	Bit wise XOR	n			
	Addition modulo2^n	$2n+1$	$n-2$	$2(n-2)$	
	Slash modulo2^n	$3(n-1)$		$n-1$	
	Nmix modulo 2^n	$5n-7$		$3n-5$	
Reverse	Bit wise XOR	n			
	Subtraction modulo2^n	$2n+1$	$n-2$	$2(n-2)$	$n-2$
	Reverse Slash modulo2^n	$3(n-1)$		$2n-3$	$2(n-1)$
	I-Nmix modulo 2^n	$5n-7$		$3n-5$	

4.2 Logic for Using *Nmix*

In this section, we explain the logic for using *Nmix* instead of XOR. In [12], different random pads are used to compute MAC from check symbols of different messages. But random pad is not practical for real life applications and, on the other hand fixed pads are not secured. So, our aim is to use one nonlinear function in addition with fixed pad to make it secure. It can be shown that if a fixed generator polynomial and fixed pad is used for MAC generation for many messages with XOR as mixing function then the scheme is vulnerable to following two attacks.

Attack-1: Generator Polynomial Recovery
In this attack, it is shown that computation of generator polynomial is possible from few correctly received message-MAC pairs. The generator polynomial $g_k(x)$

and the pad $r_k(x)$ are fixed for a sender and receiver pair because the secret key (k) is fixed. Assume $M_1(x)$ be one message polynomial. Then the corresponding check symbol polynomial $P_1(x)$ for a t-symbol error correcting RS code is given by

$$P_1(x) = x^{2t} M_1(x) \mod g_k(x) \tag{5}$$

If the code polynomial is $d_1(x)$ then

$$d_1(x) = x^{2t} M_1(x) + P_1(x) = q_1(x)\ g_k(x) \tag{6}$$

where $d_1(x)$ is the code polynomial which is multiple of generator polynomial and $q_1(x)$ is a quotient polynomial. The corresponding message-MAC polynomial $C_1(x)$ is as follows

$$C_1(x) = d_1(x) + r_k(x) = q_1(x)\ g_k(x) + r_k(x) \tag{7}$$

Assume $M_2(x)$ be another message polynomial and corresponding message-MAC polynomial $C_2(x)$ is given by

$$C_2(x) = q_2(x)\ g_k(x) + r_k(x) \tag{8}$$

From equations (7) & (8) we get

$$C_1(x) + C_2(x) = [q_1(x) + q_2(x)]\ g_k(x) \tag{9}$$

So attacker can form few equations like equation (9). Then by taking the GCD of few such newly formed equations attacker can determine the generator polynomial.

Attack-2: MAC Forgery
It can be shown that an attacker can calculate the MAC of a message which is combination of three or more odd number of correctly received messages. This attack is possible only when all messages are of equal length. Assume $M_1(x), M_2(x)$ and $M_3(x)$ be the three message polynomials and the corresponding MAC polynomials are $MAC_1(x), MAC_2(x)$ and $MAC_3(x)$ then

$$MAC_1(x) = x^{2t} M_1(x) \mod g_k(x) + r_k(x) \tag{10}$$

$$MAC_2(x) = x^{2t} M_2(x) \mod g_k(x) + r_k(x) \tag{11}$$

$$MAC_3(x) = x^{2t} M_3(x) \mod g_k(x) + r_k(x) \tag{12}$$

From (10),(11) and (12) we get

$$MAC(x) = x^{2t}[M(x)] \mod g_k(x) + r_k(x) \tag{13}$$

where $M_1(x)+M_2(x)+M_3(x) = M(x)$ and $MAC_1(x)+MAC_2(x)+MAC_3(x) = MAC(x)$. So an attacker can produce the MAC of a message (M) which is combination of three messages M_1, M_2 and M_3, provided the original messages are of equal length. This is also valid for more than three odd numbers of messages. But the generated message M may not be a meaningful message always. If it is a garbage message then there is no risk for the sender and receiver. Only adversary can take up the channel by sending such garbage message with a valid

MAC and consumes the computational resources of other node. For preventing these attacks, we proposed a scheme which employs the function *Nmix*. Next we show how it is possible to thwart the above two attacks using *Nmix*.

Nonlinearity has been introduced in our proposed scheme to compute MAC which makes over all scheme nonlinear. So we can not write the message-MAC polynomial as in equation (7). Let $C_1'(x)$ be the message-MAC polynomial in the proposed scheme then

$$C_1'(x) = x^{2t} M_1(x) + F(P_1(x),\ r_k(x)) \tag{14}$$

$$or\ C_1'(x) \neq q_1'(x)\ g_k(x) + r_k(x) \tag{15}$$

Also from Theorem 3 of the nonlinear function $F(X,R) \oplus F(Y,R) \neq X \oplus Y$, where X, Y, and R are three n-bit numbers. Therefore, addition of two message-MAC polynomial is not a multiple of generator polynomial. So, attacker can not find out the generator polynomial by taking GCD.
The new MAC can also resist the attack 2. In our proposed scheme, let any message-MAC polynomial be $C_1'(x)$ then

$$C_1'(x) \neq q_1'(x)\ g_k(x) + r_k(x) \tag{16}$$

Theorem 5 says that $F(X,R) \oplus F(Y,R) \oplus F(Z,R) \neq F(X \oplus Y \oplus Z, R)$, where X, Y, Z and R are four $n-$bit numbers. Since MAC generation scheme is non-linear, so we can conclude using theorem 5 that modulo-2 addition of three correctly received MACs is not equal to the MAC of the message which is modulo-2 addition of these three messages. Therefore an attacker can not produce the MAC of a message which is combination of three messages M_1, M_2 and M_3.

The proposed non-linear function increases the over all nonlinearity of the scheme without affecting the error correcting performance of the integrated scheme. In key mixing by XOR, single bit error in a MAC affects only single bit in the check symbol after inverse operation at the receiver. But in nonlinear mixing error may propagate from LSB to MSB i.e. single bit error in MAC may produce multiple bit errors in the check symbol. In our proposed scheme each check symbol is processed separately i.e. there is no carry propagation from one symbol to the next adjacent symbol. Since RS code can correct symbol errors, so error correcting performance will not be affected by nonlinear mixing.

5 Security Analysis

In this section, we establish that our proposed scheme is secure against some conventional attacks and robustness of the proposed scheme.

Brute force attack: In brute force attack, the level of effort require to attack on a MAC algorithm can be expressed as $\min(2^k, 2^l)$, where k is the key length and l is the MAC length. In our proposed scheme, MAC length is 320 bits and key length is 160 bits. So the level of effort required is 2^{160}, which is computationally infeasible.

Birthday Attack: Our proposed scheme produces 320 bits MAC output, so the effort required for birthday attack is 2^{160}, which is computationally infeasible.

MAC guessing: The attacker selects a message and simply guess the correct MAC value. The probability that the guess will be correct is 2^{-l}, where l is the number of bits in the MAC. In our proposed scheme, the probability that the guess will be correct is 2^{-320}.

Divide and Conquer Attack: In our proposed scheme, we used 80-bit secret information to select a particular $g(x)$ from the set of all possible generator polynomials and another 80-bit as the seed of the sequence generator. So this type of attack is not possible.

Extension Attack: From the definition of F, it can be shown that $F(X, R) \oplus Y \neq F((X \oplus Y), R)$ and $F(F(X, R), Y) \neq F((X \oplus Y), R)$. So we may infer that the proposed construction is robust against extension attack.

Linear Cryptanalysis of *Nmix*: Linear cryptanalysis (LC) tries to take advantage of high probability of occurrences of linear expressions involving message bits, key bits and MAC bits. The approach in LC is to determine linear expressions of the form which have a low or high probability of occurrence. Since the coding scheme is linear, therefore a nonlinear sequence mixing is introduced to resist LC. In this section, we derive the probability that each MAC bit can be expressed linearly in terms of check bits and sequence bits. Assuming that two n-bit inputs are X, R and the corresponding n-bit output is Y, where $Y = F(X, R)$. The following result shows that *Nmix* is a good nonlinear mixing function.

Theorem 7. The bias of best linear approximation of *Nmix* is 2^{-i}, where $2 \leq i < n$

Proof: If two $n-$bit numbers $X = (x_{n-1} \ldots x_0)$ and $R = (r_{n-1} \ldots r_0)$ generate an n-bit number $Y = (y_{n-1} \ldots y_0)$ such that $Y = F(X, R)$, then one need to prove that the bias of best linear approximation of y_i is $2^{-i}, where\ 2 \leq i < n$. From the definition of F, it is evident that in the output $y_i = x_i \oplus r_i \oplus c_{i-1}$, where c_{i-1} is the carry input into the i^{th} bit position and is the only nonlinear term of the equation. Therefore, nonlinearity of y_i is same as that of nonlinearity of c_{i-1}, which can be expressed as

$$c_{i-1} = x_0 r_0 \oplus \ \ldots \ \oplus x_{i-1} r_{i-1} \oplus r_{i-1} r_{i-2} \oplus x_{i-1} x_{i-2} \tag{17}$$

If we assume that $p_j = x_j \oplus r_{j+1}$ and $q_j = r_j \oplus x_{j+1}$ then c_{i-1} can be expressed as

$$c_{i-1} = x_0 r_0 \oplus \ \ldots \ \oplus x_{i-3} r_{i-3} \oplus p_{i-2} q_{i-2} \tag{18}$$

where p_j and q_j are statistically independent if x_j and r_j are statistically independent and uniformly chosen. Expression of c_{i-1} shows that it is a function of $2(i-1)$ variables and it is in the form of bent function. Therefore, non-linearity of c_{i-1} is $2^{2i-3} - 2^{i-2}$, where $2 \leq i < n$. So, number of matches in the best linear approximation is $2^{2i-2} - 2^{2i-3} + 2^{i-2}$ and hence probability of matches is $\frac{1}{2} + 2^{-i}$. Therefore, bias of best linear approximation is 2^{-i}, where $2 \leq i < n$. Since, $y_0 = x_0 \oplus r_0$, so bias of best linear approximation is $\frac{1}{2}$. For $y_1 = x_1 \oplus r_1 \oplus x_0 r_0$, the bias is $\frac{1}{4}$. Therefore, except the first two bits *Nmix* posses high nonlinearity at all other bits.

Table 3. Comparison of Linear Probability Bias

Mixing by	Bias of best linear Approx. of					
	y_0	y_1	y_2	y_3	y_4	y_5
Addition modulo 2^n	0.50	0.25	0.25	0.25	0.25	0.25
Slash modulo 2^n	0.50	0.25	0.125	0.0625	0.0313	0.0156
Nmix modulo 2^n	0.50	0.25	0.25	0.125	0.0625	0.0313

Table 4. Comparison of Bias of Best Linear Approximation of $y_i \oplus y_{i+1}$

Mixing by	Bias of best linear Approx. for				
	$y_0 \oplus y_1$	$y_1 \oplus y_2$	$y_2 \oplus y_3$	$y_3 \oplus y_4$	$y_4 \oplus y_5$
Addition modulo2^n	0.25	0.25	0.25	0.25	0.25
Slash modulo 2^n	0.25	0.25	0.25	0.25	0.25
Nmix modulo 2^n	0.25	0.125	0.0625	0.0625	0.0625

Table 3 shows the comparison of bias for best linear approximation for few terms. It is observed that the bias of best linear approximation of y_i decreases exponentially with i. Next we compute the bias value of best linear approximation for $y_i \oplus y_{i+1}$.

It is observed that nonlinearity of $y_i \oplus y_{i+1}$ depends solely on the nonlinearity of $c_i \oplus c_{i-1}$. From the definition, $c_i \oplus c_{i-1}$ can be expressed as

$$c_i \oplus c_{i-1} = x_i r_i \oplus x_{i-1}(x_i \oplus x_{i-2}) \oplus r_{i-1}(r_i \oplus r_{i-2}) \tag{19}$$

where $2 \leq i < n$. If we assume $a_i = x_i \oplus x_{i-2}$ and $b_i = r_i \oplus r_{i-2}$, then $c_i \oplus c_{i-1} = x_i r_i \oplus x_{i-1} a_i \oplus r_{i-1} b_i$. Therefore, $c_i \oplus c_{i-1}$ is a function of six variables and it is in the form of bent function. So, nonlinearity is 28 and bias of best linear approximation is 0.0625, where $2 \leq i < n$. Since $y_0 \oplus y_1 = x_0 \oplus x_1 \oplus r_0 \oplus r_1 \oplus x_0 r_0$, so nonlinearity is 1 and bias is 0.25. The nonlinearity of $y_1 \oplus y_2$ is 6 and bias is 0.125. Table 4 shows that bias of the linear approximation of $y_i \oplus y_{i+1}$ has significant value 0.25 for addition modulo 2^n and *Slash* modulo 2^n but bias is negligible 0.0625 in *Nmix* modulo 2^n. Therefore, *Nmix* is cryptographically stronger than addition modulo 2^n. Recently, *Slash* [19] function is used to prevent crossword puzzle attack [18] on stream cipher NLS [14]. The function modular *Slash* is nonlinear, reversible and has a strong resistance against linear cryptanalysis. However the Boolean functions like modular addition, *Slash* have the demerit that the bias of XOR of consecutive bit positions in the output is held constant at $\frac{1}{4}$. The proposed function *Nmix* eliminates above disadvantage.

Differential Cryptanalysis of *Nmix*: In our proposed scheme, sequence mixing method using nonlinear function offers differential resistance. But the sequence mixing which is done by XOR operator, does not provide any differential resistance as always $\Delta y = \Delta x$, which is independent of the sequence. In the proposed scheme nonlinearity exists only in the sequence mixing so differential resistance of the scheme is same as that of mixing part. Next we calculate the bias of linear approximation of the difference term. Assuming that

Table 5. Comparison of Bias of Best Linear Approximation of Δy_i

Mixing by	Bias of best linear Approx. of				
	Δy_0	Δy_1	Δy_2	Δy_3	Δy_4
Addition modulo2^n	0.5	0.25	0.1875	0.1719	0.1680
Slash modulo2^n	0.5	0.25	0.125	0.0625	0.0313
Nmix modulo 2^n	0.5	0.25	0.125	0.0625	0.0313

$Y = (y_{n-1}\ y_{n-2}...y_0)$ is an n-bit MAC, $R = (r_{n-1}\ r_{n-2}...r_0)$ is the key dependent sequence inputs to the non linear operator, $X = (x_{n-1}\ x_{n-2}...x_0)$ is the another n-bit input such that $Y = F(X, R)$ and c_i represents the carry from the i^{th} level. If $Y' = (y'_{n-1}\ y'_{n-2}\ \ldots\ y'_0)$ be the another n-bit MAC, $R = (r_{n-1}\ r_{n-2}\ ...r_0)$ be the key dependent sequence inputs to the nonlinear operator, $X' = (x'_{n-1}\ x'_{n-2}\ \ldots\ x'_0)$ be the other input such that $Y' = F(X', R)$ and c'_i represents the carry from the i^{th} level. Then $\Delta y_i = (y_i \oplus y'_i) = \Delta x_i \oplus \Delta c_{i-1}$, where $\Delta x_i = x_i \oplus x'_i$ and $\Delta c_{i-1} = c_{i-1} \oplus c'_{i-1}$ Therefore sequence mixing using nonlinear operator offers differential resistance as the probability distribution of Δy_i for given Δx_i is identical to the probability distribution of Δc_{i-1}. From equation (2) it can be shown that

$$\Delta c_{i-1} = x_{i-2}x_{i-1} \oplus x'_{i-2}x'_{i-1} \bigoplus_{j=0}^{i-1} r_j(x_j \oplus x'_j) \tag{20}$$

It is found from Table 5 that bias of best linear approximation of Δy_i is also decreases exponentially. But in case of addition modulo 2^n, rate of decrease of bias relatively flat compared to *Nmix* modulo 2^n. From (20), it can be shown that $\Delta y_i = \Delta x_i \oplus x_{i-2}x_{i-1} \oplus x'_{i-2}x'_{i-1} \bigoplus_{j=0}^{i-1} r_j \Delta x_j$, where $\Delta y_i = y_i \oplus y'_i$ and $\Delta x_i = x_i \oplus x'_i$, X, X' are the two different inputs to the function and corresponding outputs are Y, Y' for a fixed R and $0 \leq i < n$. So it is observed that the probability of a particular output difference ΔY occurs given a particular input difference ΔX is function of all x_i and x'_i for a fixed R. The probability $Pr(\Delta Y|\Delta X)$ is always much less than 1. Therefore the proposed function provides differential resistance.

The following section describes the performance of the proposed scheme as a hashing algorithm.

6 Evaluation of the Proposed Scheme

In this section, we discuss the computational cost of the proposed scheme as well as the conventional scheme (first MAC then ECC). The proposed MAC algorithm is also evaluated against the metrics: (i) Bit-variance test and (ii) Entropy test proposed in [10].

6.1 Computational Cost

In our MAC generation scheme, total computational cost is equal to the computation cost of a systematic RS encoder with programmable generator polynomial

and *Nmix* function. The systematic RS encoder [2] requires $2tN_m$ number of $GF(2^n)$ multiplication and $2tN_m$ number of $GF(2^n)$ addition, where N_m is the total number of message symbols in the code word, t is the error correction capability of the code and n is the size of each symbol. From Table 2 it can be shown that for *Nmix*, computational cost is $2t(5n-7)$ number of XOR operation and $2t(3n-5)$ number of AND operation. Computational cost of proposed MAC verification scheme consists of computational cost of *I-Nmix* and a systematic RS decoder with programmable generator polynomial. Computational cost of *I-Nmix* is same as that of *Nmix* and is equal to $2t(5n-7)$ number of XOR operation and $2t(3n-5)$ number of AND operation. It has been shown in [4] that for decoding RS code with programmable generator polynomial require extra four Galois field multipliers and two registers of size same as that of symbol size. Therefore, computational cost of the stated RS decoder is equal to computational cost of a non programmable RS decoder and $2(N_m+N_c)$ number of $GF(2^n)$ multiplication, where N_m is the number of message symbols in the code word and N_c is the code word length in symbols.

In case of conventional scheme, first MAC is computed then ECC is computed over message and MAC. In sender, the total computational cost is equal to computational cost of MAC generator and RS encoder. If any HMAC algorithm is used to compute MAC then it mainly requires two hash operations. If SHA-1 is used then for each 512-bit message block round operation is processed 80 times. Each round operation performs four addition modulo 2^{32}, two circular right shift and one logical operation. In the receiver, computational is sum of the computational cost due to RS decoder and MAC generator. Therefore, the computational cost of the integrated ECC-MAC is much less than the computational cost of conventional scheme.

6.2 Evaluation of Our Scheme as a Hash Function

1. Bit Variance Test: Bit variance test shows the impact on the MAC bits by changing the input message bits. Bits of an input message are changed and the corresponding MACs (for each changed input) are calculated for a fixed secret key. Then the difference between the original MAC (corresponding to the original message) and the changed MACs are calculated. Finally, the average probability of 1 and 0 are calculated from all the differences of MACs. For this test, it is difficult to check for all possible bit changes on the input message. Therefore, only changes involving 1 bit and 2 bits have been considered.

Table 6. Result of Bit Variance Test

Message	Probability of change for the variation of							
	One message bit and mixing by				Two message bits and mixing by			
	XOR		Nmix		XOR		Nmix	
	p_0	p_1	p_0	p_1	p_0	p_1	p_0	p_1
M_1	0.499	0.500	0.501	0.498	0.498	0.501	0.501	0.498
M_2	0.499	0.500	0.503	0.496	0.498	0.501	0.499	0.500
M_3	0.499	0.500	0.499	0.500	0.498	0.501	0.505	0.494

Cellular Automata based RS encoder [20] with some modification in the algorithm, is used for our experiment with symbol size 40 bits, message length 10000 bits, and generated MAC length 320 bits. The experiments were performed for three different messages. Results are given in Table 6. It shows that for both the mixing function XOR and *Nmix* the condition $p_0 \approx p_1 = 0.5$ is satisfied, where p_0 and p_1 are the average probability of 0 and 1.

2. Entropy Test: Entropy is a measure of uncertainty. The entropy is maximum when all the MACs are equally likely. Entropy test addresses the question about the actual probability to find a message with given MAC or two messages with the same MAC. Since, it is infeasible to calculate the probability of the individual MACs , an approximate entropy assessment method is used.

Approximate Entropy: Approximate entropy is a measure of the logarithmic frequency with which blocks of length i that are close together remain close together for blocks augmented by one position. By comparing the actual frequency of groups of digits to their expected frequency, approximate entropy of the sequence is determined which is a measure of it's randomness. A random sequence should have equal number of all possible groups. A small value of approximate entropy implies strong regularity. For this test, it is difficult to check for all possible groups. Therefore, only 2 bytes block has been considered.

Approximate Entropy Assessment Method: Assume the MAC is composed of blocks where the length of each block is 1 byte. By taking all possible combinations of byte pairs, a set of 16 bits numbers $(0 - 65535)$ are obtained. For a large number of MAC if the frequencies of these numbers $(0 - 65535)$ are equal, then the approximate entropy of the 16 bit long sub sequence is maximum and the value of this entropy is 16. For approximate entropy calculation we took 260796 samples from the produced MACs with each samples of length 2-byte. The calculated approximate entropy for 16 bits subsequence is 15.8001, which is slightly less than maximum entropy 16.

7 Conclusions

In this correspondence, we have proposed a MAC algorithm based on RS code having t-symbol error correcting capability. In our proposed MAC generation algorithm, we have used a function *Nmix* for mixing the sequence with the error correcting check symbols. The proposed scheme is shown to be secure even if same pad is used for more than one MAC generation. Also the proposed MAC is found to be a good choice for Keyed-Hash technique and has been evaluated successfully for Bit-Variance and Entropy test. The proposed function can be used effectively as a key mixing function in hardware based block ciphers.

References

1. Rothaus, O.S.: On "Bent" Functions. Journal of Combinatorial Theory 20(A), 300–305 (1976)
2. Lin, S., Costello, D.J.: Error control coding: Fundamentals and Applications. Prentice-Hall, Englewood Cliffs (1983)

3. Mceliece, R.J., Swanson, L.: Decoder Error Probability for Reed-Solomon Codes. IEEE Transaction on Information Theory IT-32(5), 701–703 (1986)
4. Shayan, Y.R., Le-Ngoc, T.: Decoding Reed-Solomon Codes Generated by any Generator Polynomial. Electronics Letters 25(18), 1223–1224 (1989)
5. Ohtal, K., Matsui, M.: Differential attack on Message Authentication Codes. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 200–211. Springer, Heidelberg (1994)
6. Krawczyk, H.: LFSR-based Hasing and Authentication. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 129–139. Springer, Heidelberg (1994)
7. Bakhtiari, S., Safavi-Naini, R., Pieprzyk, J.: Cryptographic Hash Functions: A Survey. Technical Report 95-09. Department of Computer Science, University of Wollongong (1995), `http://www.citeseer.ist.psu.edu/bkhtiari95cryptographic.html`
8. Bellare, M., Canettiy, R., Krawczykz, H.: Keying Hash Functions for Message Authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
9. Preneel, B.: Cryptanalysis of Message Authentication Code. In: Okamoto, E. (ed.) ISW 1997. LNCS, vol. 1396, pp. 55–65. Springer, Heidelberg (1998)
10. Karras, D.A., Zorkadis, V.: A Novel Suite for Evaluation One-Way Hash Functions for Electronic Commerce Applications. In: Proc. of 26th EUROMICRO 2000, Netherlands, pp. 464–468 (2000)
11. Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography, 5th edn. CRC Press, Boca Raton (2001)
12. Lam, C.C.Y., Gong, G., Vanstone, S.: Message Authentication Codes with Error Correcting Capabilities. In: Deng, R.H., Qing, S., Bao, F., Zhou, J. (eds.) ICICS 2002. LNCS, vol. 2513, pp. 354–366. Springer, Heidelberg (2002)
13. Sarkar, P., Mitra, S.: Construction of Nonlinear Resilient Boolean Functions Using "Small" Affine functions. IEEE Transactions on Information Theory 50(9), 2185–2193 (2004)
14. Rose, G., Hawkes, P., Paddon, M., De Vries, M.W.: Primitive specification for nls (2005), `http://www.ecrypt.eu.org/stream/nls.html`
15. Liu, Y., Boncelet, C.G.: The CRC-NTMAC for Noisy Messages Authentication. IEEE Transaction on Information Forensics and Security 1(4), 517–523 (2006)
16. Liu, Y., Boncelet, C.G.: The BCH-NTMAC for Noisy Messages Authentication. In: Proceedings of CISS (2006)
17. Contini, S., Yin, Y.L.: Forgery and Partial Key-Recovery Attacks on HMAC and NMAC Using Hash Collisions. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 37–53. Springer, Heidelberg (2006)
18. Cho, J.Y., Pieprzyk, J.: Crossword Puzzle Attack on NLS. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 249–265. Springer, Heidelberg (2007)
19. Bhattacharya, D., Mukhopadhyay, D., Saha, D., Roy Chowdhury, D.: Strengthening NLS against Crossword Puzzle Attack. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 29–44. Springer, Heidelberg (2007)
20. Bhaumik, J., Roy Chowdhury, D., Chakrabarti, I.: An Improved Double Byte Error Correcting Code using Cellular Automata. In: Umeo, H., Morishita, S., Nishinari, K., Komatsuzaki, T., Bandini, S. (eds.) ACRI 2008. LNCS, vol. 5191, pp. 463–470. Springer, Heidelberg (2008)

Appendix

A. Proof of the *Nmix* Properties

Theorem 1. The function G is the inverse function of F.

Proof: If X, Y and R be three n-bit data such that $Y = F(X, R)$ and function G is defined as $X = G(Y, R)$,then G is the inverse function of F. From the definition of F, bitwise expressions of Y are as follows

$y_0 = x_0 \oplus r_0$
$y_1 = x_1 \oplus r_1 \oplus x_0 r_0$
$y_2 = x_2 \oplus r_2 \oplus x_1 r_1 \oplus x_0 r_0 \oplus x_0 x_1 \oplus r_0 r_1$

similarly it can be shown that

$y_{n-1} = x_{n-1} \oplus r_{n-1} \oplus x_0 r_0 \oplus \ \ldots \ \oplus x_{n-3} x_{n-2} \oplus r_{n-3} r_{n-2}$

Let $P = G(Y, R)$, where $P = (p_{n-1} \ \ldots \ p_0)$ is an n-bit number. From the definition of G, the bitwise expressions of P after simplification are given below

$p_0 = y_0 \oplus r_0 = x_0 \oplus r_0 \oplus r_0 = x_0$
$p_1 = y_1 \oplus r_1 \oplus p_0 r_0 = x_1 \oplus r_1 \oplus x_0 r_0 \oplus r_1 \oplus x_0 r_0 = x_1$
$p_2 = y_2 \oplus r_2 \oplus p_1 r_1 \oplus p_0 r_0 \oplus p_0 p_1 \oplus r_0 r_1 = x_2$

similarly it can be shown that

$p_{n-1} = y_{n-1} \oplus r_{n-1} \oplus \ \ldots \ \oplus y_{n-3} y_{n-2} \oplus r_{n-3} r_{n-2} = x_{n-1}$

Since, $p_i = x_i$ for all $0 \leq i \leq n-1$ therefore G is the inverse function of F.

Theorem 2. The function F does not hold associative law, i.e. $F(F(X, Y), Z) \neq F(X, F(Y, Z))$, where X, Y and Z are three n-bit numbers.

Proof: Let A and B be the two n-bit numbers such that $A = F(F(X, Y), Z)$ and $B = F(X, F(Y, Z))$. From the definition of F, bitwise expressions of A are as follows

$a_0 = x_0 \oplus y_0 \oplus z_0$
$a_1 = x_1 \oplus y_1 \oplus z_1 \oplus x_0 y_0 \oplus y_0 z_0 \oplus x_0 z_0$
$a_2 = x_2 \oplus y_2 \oplus z_2 \oplus x_1 y_1 \oplus y_1 z_1 \oplus z_1 x_1 \oplus x_0 y_0 \oplus \ \ldots \ \oplus x_0 y_1 \oplus x_1 y_0 \oplus z_1 (x_0 y_0 \oplus z_0)$

similarly it can be shown that

$a_{n-1} = x_{n-1} \oplus y_{n-1} \oplus z_{n-1} \oplus \ \ldots \ \oplus z_{n-2} (x_{n-3} x_{n-4} \oplus y_{n-3} y_{n-4})$

Similarly bitwise expression of B can be written as

$b_0 = x_0 \oplus y_0 \oplus z_0$
$b_1 = x_1 \oplus y_1 \oplus z_1 \oplus x_0 y_0 \oplus y_0 z_0 \oplus z_0 x_0$
$b_2 = x_2 \oplus y_2 \oplus z_2 \oplus x_1 y_1 \oplus y_1 z_1 \oplus z_1 x_1 \oplus x_0 y_0 \oplus \ \ldots \ \oplus y_0 z_1 \oplus y_1 z_0 \oplus x_1 (y_0 z_0 \oplus x_0)$

similarly it can be shown that

$b_{n-1} = x_{n-1} \oplus y_{n-1} \oplus z_{n-1} \oplus \ \ldots \ \oplus x_{n-2} (y_{n-3} y_{n-4} \oplus z_{n-3} z_{n-4})$

From the bit level expressions of A and B, it is observed that $a_0 = b_0$, $a_1 = b_1$ but $a_i \neq b_i$ for all $2 \leq i < n$. Hence, F does not follow associative property.

Theorem 3. Output difference (XOR) of the function F is not equal to input difference (XOR) when single input changes, i.e. $F(X,R) \oplus F(Y,R) \neq X \oplus Y$.

Proof: Assume $X = (x_{n-1}\ x_{n-2}\ ...\ x_0)$, $Y = (y_{n-1}\ y_{n-2}\ ...\ y_0)$, $R = (r_{n-1}\ r_{n-2}\ ...\ r_0)$, $U = (u_{n-1}\ u_{n-2}\ ...\ u_0)$ and $V = (v_{n-1}\ v_{n-2}\ ...\ v_0)$ be the five $n-$bit numbers, such that $U = F(X,R)$ and $V = F(Y,R)$, then one need to prove $U \oplus V \neq X \oplus Y$.

According to the definition of nonlinear function F, the bitwise expressions of $U \oplus V$ are as follows

$u_0 \oplus v_0 = x_0 \oplus y_0$
$u_1 \oplus v_1 = (x_1 \oplus y_1) \oplus r_0(x_0 \oplus y_0)$
$u_2 \oplus v_2 = (x_2 \oplus y_2) \oplus r_1(x_1 \oplus y_1) \oplus r_0(x_0 \oplus r_0) \oplus (x_0x_1 \oplus y_0y_1)$

Similarly it can be shown that

$u_{n-1} \oplus v_{n-1} = x_{n-1} \oplus y_{n-1} \oplus\ ...\ \oplus (x_{n-2}x_{n-3} \oplus y_{n-2}y_{n-3})$

Hence, it is proved that $U \oplus V \neq X \oplus Y$.

Also it can be shown that

Theorem 4. If X, Y and R are the three n-bit numbers then

$F(F(X,R),F(Y,R)) \neq F(X,Y)$.

Proof: Let $X = (x_{n-1}\ x_{n-2}\ ...\ x_0)$, $Y = (y_{n-1}\ y_{n-2}\ ...\ y_0)$, $R = (r_{n-1}\ r_{n-2}\ ...\ r_0)$, $U = (u_{n-1}\ u_{n-2}\ ...\ u_0)$, $V = (v_{n-1}\ v_{n-2}\ ...\ v_0)$, $W = (w_{n-1}\ w_{n-2}\ ...\ w_0)$ and $P = (p_{n-1}\ p_{n-2}\ ...\ p_0)$ be the seven $n-$bit numbers, such that $U = F(X,R)$, $V = F(Y,R)$, $P = F(U,V)$ and $Q = F(X,Y)$, then it is required to prove that $P \neq Q$.

Representing P at bit level and after simplification we get

$p_0 = u_0 \oplus v_0 = x_0 \oplus y_0$
$p_1 = u_1 \oplus v_1 \oplus u_0v_0 = x_1 \oplus y_1 \oplus x_0y_0 \oplus r_0$
$p_2 = u_2 \oplus v_2 \oplus u_1v_1 \oplus u_0v_0 \oplus u_1u_0 \oplus v_1v_0$
or $p_2 = x_2 \oplus y_2 \oplus x_1y_1 \oplus x_0y_0 \oplus\ \ldots\ \oplus r_0r_1(x_0 \oplus y_0)$

similarly it can be shown that

$p_{n-1} = x_{n-1} \oplus y_{n-1} \oplus\ \ldots\ \oplus r_{n-3}r_{n-4}(x_{n-3}x_{n-4} \oplus y_{n-3}y_{n-4})$

From the definition of F, bitwise expressions of Q are as follows

$q_0 = x_0 \oplus y_0$
$q_1 = x_1 \oplus y_1 \oplus x_0y_0$
$q_2 = x_2 \oplus y_2 \oplus x_1y_1 \oplus x_0y_0 \oplus x_0x_1 \oplus y_0y_1$

similarly it can be shown that

$q_{n-1} = x_{n-1} \oplus y_{n-1} \oplus\ \ldots\ \oplus x_{i-2}x_{i-1} \oplus y_{n-2}y_{n-1}$

It is noted that $p_0 = q_0$ but $p_i \neq q_i$, for all $1 \leq i < n$. Hence, it is proved that $F(F(X,R),F(Y,R)) \neq F(X,Y)$

Theorem 5. Keeping one input fixed of the two input function F, modulo-2 addition of three outputs for three different inputs is not equal to output of the function F when input is modulo-2 addition of three individual inputs.

i.e. $F(X,R) \oplus F(Y,R) \oplus F(Z,R) \neq F((X \oplus Y \oplus Z), R)$.

Proof: Assume $X = (x_{n-1}\ x_{n-2}\ ...\ x_0)$, $Y = (y_{n-1}\ y_{n-2}\ ...\ y_0)$, $Z = (z_{n-1}\ z_{n-2}\ ...\ z_0)$, $R = (r_{n-1}\ r_{n-2}\ ...\ r_0)$, $U = (u_{n-1}\ u_{n-2}\ ...\ u_0)$, $V = (v_{n-1}\ v_{n-2}\ ...\ v_0)$ and $W = (w_{n-1}\ w_{n-2}\ ...\ w_0)$ be the seven $n-$bit numbers, such that $U = F(X,R)$, $V = F(Y,R)$, $W = F(Z,R)$ then the theorem says that $F(X,R) \oplus F(Y,R) \oplus F(Z,R) \neq F((X \oplus Y \oplus Z), R)$. Assuming that $A = U \oplus V \oplus W$ and $B = F((X \oplus Y \oplus Z), R)$, then the bitwise expressions of A are as follows

$a_0 = (x_0 \oplus y_0 \oplus z_0) \oplus r_0$
$a_1 = (x_1 \oplus y_1 \oplus z_1) \oplus r_1 \oplus r_0(x_0 \oplus y_0 \oplus z_0)$
$a_2 = (x_2 \oplus y_2 \oplus z_2) \oplus r_2 \oplus\ \ldots\ \oplus r_0 r_1 \oplus (x_0 x_1 \oplus y_0 y_1 \oplus z_0 z_1)$

Similarly it can be shown that

$a_{n-1} = x_{n-1} \oplus y_{n-1} \oplus z_{n-1} \oplus ... \oplus r_{n-2} r_{n-3} \oplus (x_{n-2} x_{n-3} \oplus y_{n-2} y_{n-3} \oplus z_{n-2} z_{n-3})$

The bitwise expressions of B are given below

$b_0 = (x_0 \oplus y_0 \oplus z_0) \oplus r_0$
$b_1 = (x_1 \oplus y_1 \oplus z_1) \oplus r_1 \oplus r_0(x_0 \oplus y_0 \oplus z_0)$
$b_2 = (x_2 \oplus y_2 \oplus z_2) \oplus r_2 \oplus\ \ldots\ \oplus r_0 r_1 \oplus (x_0 \oplus y_0 \oplus z_0)(x_1 \oplus y_1 \oplus z_1)$

Similarly it can be shown that

$b_{n-1} = x_{n-1} \oplus y_{n-1} \oplus z_{n-1} \oplus ... \oplus (x_{n-3} \oplus y_{n-3} \oplus z_{n-3})(x_{n-2} \oplus y_{n-2} \oplus z_{n-2})$

From the bitwise expressions of A and B, it is observed that $a_0 = b_0$ and $a_1 = b_1$ but $a_i \neq b_i$ for $2 \leq i < n$. Hence it is proved that $F(X,R) \oplus F(Y,R) \oplus F(Z,R) \neq F((X \oplus Y \oplus Z), R)$. Also it can be shown that

Theorem 6. If X, Y, Z and R are the four n-bit numbers then $F(F(F(X,R),F(Y,R)),F(Z,R)) \neq F(F(F(X,Y),Z),R)$

Proof: Let $P = (p_{n-1} \ldots p_0)$ and $Q = (q_{n-1} \ldots q_0)$ be the two n- bit numbers such that $P = F(F(F(X,R),F(Y,R)),F(Z,R))$ and $Q = F(F(F(X,Y),Z),R)$. From the definition of F, the bitwise expressions of P are given below

$p_0 = x_0 \oplus y_0 \oplus z_0 \oplus r_0$
$p_1 = x_1 \oplus y_1 \oplus z_1 \oplus r_1 \oplus r_0(x_0 \oplus y_0 \oplus z_0) \oplus x_0 y_0 \oplus y_0 z_0 \oplus z_0 x_0$
$p_2 = x_2 \oplus y_2 \oplus z_2 \oplus\ \ldots\ \oplus r_0(x_1 \oplus y_1 \oplus x_1 y_0 \oplus x_0 y_1 \oplus x_0 y_0 \oplus x_1 z_0 \oplus y_1 z_0)$

similarly it can be shown that

$p_{n-1} = x_{n-1} \oplus y_{n-1} \oplus z_{n-1} \oplus\ \ldots\ \oplus r_{n-3} z_{n-3}(x_{n-4} x_{n-5} \oplus y_{n-4} y_{n-5})$

Similarly bitwise expressions of Q are

$q_0 = x_0 \oplus y_0 \oplus z_0 \oplus k_0$
$q_1 = x_1 \oplus y_1 \oplus z_1 \oplus r_1 \oplus r_0(x_0 \oplus y_0 \oplus z_0) \oplus x_0 y_0 \oplus y_0 z_0 \oplus z_0 x_0$
$q_2 = x_2 \oplus y_2 \oplus z_2 \oplus\ \ldots\ \oplus r_0(x_0 \oplus y_0 \oplus z_0)$

similarly it can be shown that

$$q_{n-1} = x_{n-1} \oplus y_{n-1} \oplus z_{n-1} \oplus \ \ldots \ \oplus r_{n-2} z_{n-3}(x_{n-4} x_{n-5} \oplus y_{n-4} y_{n-5})$$

From the bitwise expressions of P and Q, it is observed that $p_0 = q_0$, $p_1 = q_1$ but $p_i = q_i$ for all $2 \leq i < n$. Hence, it is proved that $F(F(F(X,R),F(Y,R)),F(Z,R)) \neq F(F(F(X,Y),Z),R)$.

B. Logic Diagram of *Nmix* and *I-Nmix*

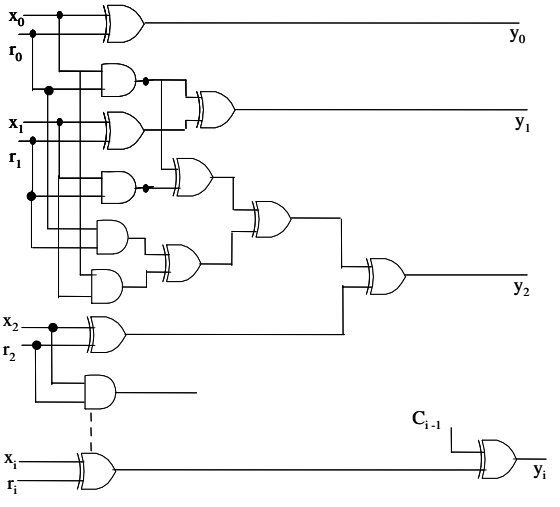

Fig. 4. Logic Diagram for *Nmix*

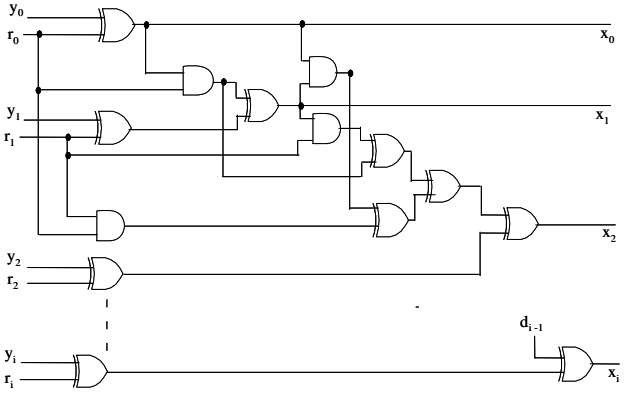

Fig. 5. Logic Diagram for *I-Nmix*