A scheme reducing the overhead costs in the transmission of message authentication code based on Reed Solomon code

Alcides Bernardo Tello

a.btello@hotmail.com

Universidad Nacional HermilioValdizán
Huánuco, Perú

Abstract: *In this paper, we depict our new scheme for transmission ofMessage Authentication Code (MAC)by making use of Reed Solomon code parity checks over Galois Field GF(2^m), whereby we reduce the overhead transmission costs.As the computation of one bit is by far cheaper than transmitting one bit, we have found that before sending any information from source to destination, we can computethe information to transform it into smaller message using encoding and decoding techniques without loosingneither integrity nor authenticity of the message. Our measurement came from a program implemented in C programming language to test our scheme.Sending fewer MAC bits, we can save energy in many applications where energy use is of great concern such as in sensor networks.*

Keywords: MAC, Galois Field, encoding, decoding, RS code, authentication.

1. Introduction

Receiving and sending messages is a critical part of our everyday lives; messages keep people connected across any of the variety of networks they alreadyutilize. The landmark idea of this study relied heavily on the fact that a message transmission could be achieved with less overhead costs if we make some computation over the stream of bits prior to transmitting them.

In every commercial and prototype communication systems, especially in sensor network community, transmitting one bit for one hop is on the order 10^5 times more expensive than computing one instruction on one bit.From the power consumption analysis [Xioa10] reveals: For most processor instructions, the energy required is $4.3*10^{-12}$ joules per bit. Multiplication requires $31.9 * 10^{-12}$ joules per bit whilst radio frequency ground communications require 10^{-7} joules per bit for 0-50 meters, and $50*10^{-6}$ joules per bit for one to ten kilometers. [Wand05] has also shown that an equivalent of 2090CPU clock cycles is required to transmit one single bit.

When sending messages, given a message *m(x)* of arbitrary length L_{msg}, it is known that in order to protect both data integrity as well as authenticity of messages, the sender runs an algorithm to generate message authentication code (*MAC*) for m(x) of fixed length L_{MAC}.*MAC(x)* $= C_K(m(x))$, where *K*is the secret key, C_K is the algorithm or function that transform *m*(*x*) into a value *MAC(x)*. Then the pair (*m(x), MAC(x)*) is sent to the destination, therefore extending the transmission cost as the new length is (L_{msg}+ L_{MAC}).

We have used reed solomon code to reduce the message transmission load by adding the 2*t* check symbols of RS code to the message instead of sending its MAC. Any application that uses MAC would get benefit from this scheme.

In the next two sections we make a general brief review of authentication code and reed solomon code. Then we present our proposed scheme for sending the RS parity checks instead of MAC. The penultimate section conveysour results and the final section presents the conclusions.

2. Message Authentication Code

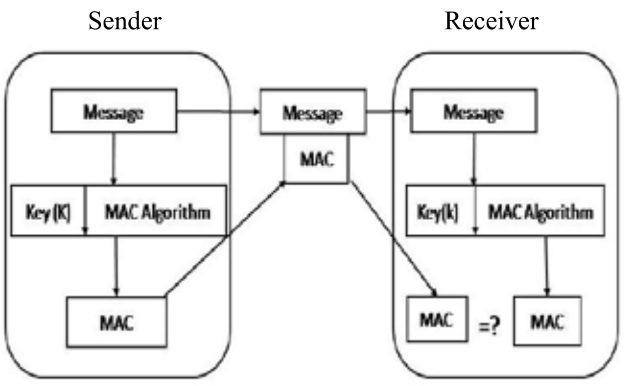

Figure 2.1. Message Authentication Code

[Paar10] provides details in this topic. In order to verify a message was not altered or tampered with during transmission either accidentallyor intentionally, there is an additional generated message MAC appended to the main message which will provide data integrity as well as authenticitybecause the access is limited to the person that possesses the secret key that can open the MAC value for verifying the data. [Mene11] shows details in Cryptography for designingsecure communication systems.

The figure 2.1 shows how the message authentication code works. An arbitrary-length message m(x) and key K are the inputs for the algorithm and it outputs a new fixed length value called MAC which clearly depends on the message and the key K.

The message together with its MAC is then sent to the receiver. Once the receiver gets the message, it in turn, recomputed the MAC by running only the message portion through the same MAC algorithm using the same key K, and hence producing a second MAC at the receiver side. Afterwards, the receiver compares the first MAC received from the sender against the second generated MAC at the receiver side. If they match, thenit is assumed that the integrity of the message is guaranteed.

3. Reed Solomon Code

From [Mceli04] and [Hump96] Reed Solomon (RS) code is a subclass of nonbinary BCH codes which is widely utilized in numerous applications due to its burst error correcting capability.

An (n, k) t-error correcting RS code with symbols from GF(2^m) has the following parameters:

Symbol length:	m bits per symbol
Block length:	n=2^m-1 symbols = m(2^m-1) bits
Data length:	k symbols

Number of parity-check: n-k=2t symbols=m(2t) bits

Minimum Distance: dmin = 2t+1.

3.1. Systematic Form Encoding of RS Codes

In this part we will regard a block code as a way of mapping some number of k symbols to another number n symbols. We will call the block of k symbols a message polynomial, and the block of n symbols a codeword polynomial.

Consider RS codes with symbols from GF(2^m) , and let α be a primitive element in GF(2^m).

The generator polynomial of a primitive t-error correcting RS-code of length2^m-1 is $g(x) = (x + \alpha)(x +$ *α2…(x+α2t)*

Let $m(x) = m_0 + m_1x + m_2x^2 + \cdots + m_{k-1}x^k$

be the message to be encoded, k=n-2t. The 2t parity-check digits are the coefficients of the remainder

$$b(x) = b_0 + b_1x + b_2x^2 + \cdots + b_{k-1}x^{k-1}$$

resulting from dividing the message polynomial $x^{2t}m(x)$ by the generator polynomial g(x).

The resulting codeword polynomial v(x) can be writing as

$$v(x) = b(x) + x^{2t}m(x)$$

3.2. Decoding RS Codes

Let $v(x) = v_0 + v_1x + v_2x^2 + \cdots + v_{n-1}x^{k-1}$

be the transmitted code polynomial and $r(x) = r + r_1x + rx^2 + \cdots + r_{n-1}x^{n-1}$the corresponding received polynomial.

Then$e(x) = e_0 + e_1x + e_2x^2 + \cdots + e_{n-1}x^{n-1}$

the error pattern added by the channel, which can also be expressed as $e(x) = r(x) - v(x)$

where in the all the forgoing polynomials the coefficients are symbols in GF(2^m).

The decoding consists of the following four steps:

- Syndrome computation
- Determination of the error-location polynomial pattern
- Determination of error value evaluator
- Error correction.

4. Proposed scheme

As shown in figure 4.1, mixing MAC and RS code together makes it possible to reduce the overhead transmission costs provided that the RS -parity checks are less in length than that of MAC length.

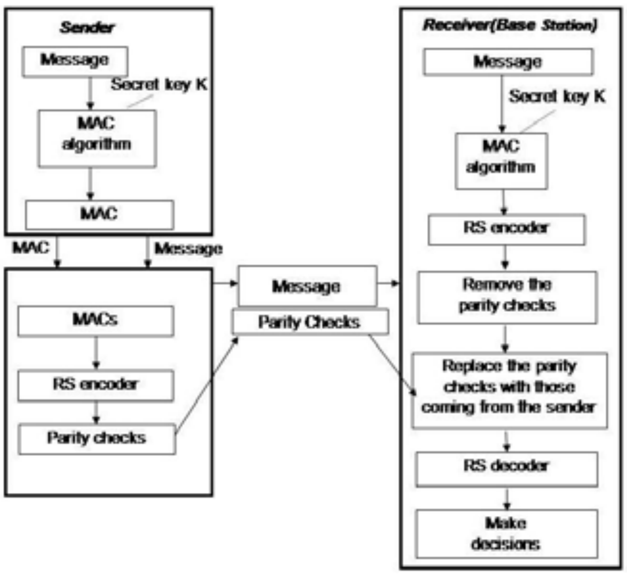

Figure 4.1. Proposed Scheme for paritycheckstransmission

The steps at the sender side are as follows:

1. Get the message m(x)
2. Run the message through the MAC algorithm which outputs the MAC(x).
3. EncodetheMACusingRS(k, t).
4. Extract the *t* parity checks from the output of RS(k, t) related to MAC(x)
5. Send the message together with its corresponding parity checks.

At the receiver side, it operates as follows:

6. Getthe message$m(x)$ coming from the sender.
7. Run the message through the MAC algorithm to produce its MAC
8. Encode the MAC using $RS(k, t)$.
9. Replace the *t* parity checks from the output of $RS(k, t)$ related to its $MAC(x)$ with the received t parity checks coming from the sender.
10. Decode the encoded MAC.
11. Make decision based on the decoded information.

As we have seen, we do not send the MAC from the sender side, we perform RS-encoding on the MAC instead and then we get the parity checks which will be sent to the receiver together with its corresponding message.

Let *Lmsg* be the total number of bits in the message m(x).

Let *Lmac* be the total number of bits in a single MAC.

The total number of bits to be sent by this new scheme is as follows:

$L = L_{msg} + 2t$

The term t is the error capability of RS code. Therefore, by this construction, our proposed scheme requires smaller transmission size.

Given a *MAC* with N bits, in order to work with elements or symbols in a finite field $GF(2^m)$, we arrange the N bits into groups of m bits. Thus, the number of symbols in each set is fixed by $Nms = N/m$.

After we have grouped the bits we cant reat each group of bits as a symbol in GF (2^m) forming a new set of symbols for each MAC.

$$M_1, M_2, \dots, M_{k,}, M_{k+1}, \dots, M_{N_{ms}}$$

Where each M_i is an element in GF (2^m)

In a typical encoding process, as a requirement for Reed Solomon code for the number of message symbols, we need k symbols out of N_{MS} symbols of each *MAC*, so that we will have *RS(n,k)* code with $2t$ parity checks where $n-k=2t$; and k can also be expressed *as* $k=2^m-1-2t$. At this time, we will have $c = N_{ms}/k$ codewords per MAC whence $N_{MS} = kc$.

After the encoding process, 2t symbols are added to each codeword as redundancy. This redundancy is exploited at the receiver to detect false message.

At the receiver end, which might be the base station, we performa teach step the same calculation as for thes end eruntil the encoder level asis shown in the figure 4.1. MAC values are generated as usual, and then they will be the input sequence to the encoder.

A given message for encoding will always produce the same parity check assuming that it uses the same encoding algorithm mas the sender side to generate the sets of parity checks

Now, before the RS decoder takes the code words there computed parity checks are replaced by those which were computed at the sender so that the decoder accepts as input the computed message code word at the receiver with its received parity checks from the sender.

This is executed in order to detect any modifications to either the message contentor in the MAC or in both. The message is accepted only if and when the syndrome is a zero vector and it is recognized as valid messages by th e machine.

5. Experiments and Results

5.1. Implementation in C.

To prove our scheme, we have implement edit in C programming language where by we computed the error location polynomial through the Berlekamp iterative algorithm following the notation and terminology in [Shu83] for RS code.

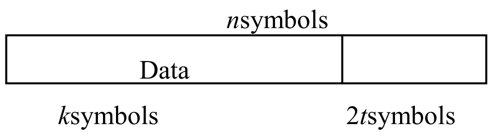

Figure 5.1 RS (*n*,*k*) code

The main functions that we have implemented are as follows:

The void GF Generator() Function.

Given the parameter M which specifies the number of bits per symbol and the irreducible polynomial, this function will generate the 2^m distinct elements of the field GF(2^m) which we labels as 0, 1, 2,..., 2^m-1, each corresponding to elements of the Galois field via a polynomial expression involving a primitive element of the field.

The Poly_Generator() Function

RS code with minimum distance *dmin= 2t+1* is a cyclic code whose generator polynomialg(x) has 2t consecutive roots in a Galois Field. This function obtains the generator polynomial of the t-error correcting Reed Solomon code from the product of ($X+\alpha^i$) where$i=1..2t$.

The encode_rs() Function

We have implemented the RS encoder in the systematic manner to produce the $2t$ parity symbols. Taking the message symbols, encoding is done by using a feedback shift register. The process is depicted in the figure 5.2

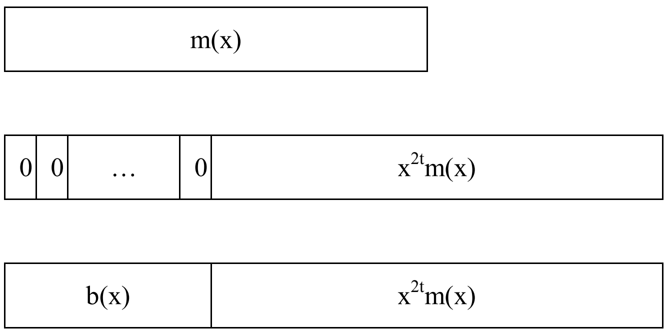

Figure 5.2. Sketch of Systematic encoding of Reed SolomonCode

The decode_rs() Function

The RS decoding is usually carried onin the following five steps:

- Compute the 2t syndromes.
- Find the error locator polynomial σ(x).

- ➢ Find the inverses of the zeros ofσ(x), then we knownthe error positions.
- ➢ Find the values of the errors by using Forney algorithm.
- ➢ Correct the errors.

There are approximately three decoding methods for finding the error locator polynomial in the RS decoding viz Berlekamp-Massey algorithm, Peterson's algorithm and Euclidean algorithm. In our program, we have implemented the Belerkamp-Massey algorithm.

5.2. Authentication Analysis

Since the number of codewords is given by$c = N_m/k$, it follows that the total number of parity checks is 2tc where each symbol has m bits. Therefore, the probability of forging a MAC is equivalent to guess the parity block of 2tcm bits. Compare to MAC transmission, in this case there is a decrease in the probability from $1/2^N$ *to* $1/22^{tcm}$

5.3. Performance Analysis

Let's assume that there is a MAC with N bits at the sender side.In what follows in this section, we evaluate our scheme in terms of transmission cost, computation cost and storage cost.

5.4. Computational cost

The computational cost in our scheme comes from the operations performed at the encoder and decoder. Although the computational cost in our scheme is larger, we have already shown the reason why computing the message is more convenient than transmitting it.

In order to know how much processor time the program uses, we got a process' CPU time by using the clock function of the language programming.

Setting the following parameters:

N=196; The number of bits per each MAC.

t = 3; Error-correction capability of the code

n = 2^m-1; Codeword length.

NumberSymbols = ceil(N/m), Number of symbols per each MAC by ceiling function.

k = n-2*t; % Message lengths for code or Number of symbols per message.

We assume that the worst case is when errors appear in every block. The best case is when there is no error in any block. By changing the value of m, we got the following time values in seconds:

Table 5.3: Elapsed time for different values of m of GF(2^m)

	4	6	8	16
Worst case	0.0470s	0.1090s	0.2810s	45.3130s
Best case	0.0408s	0.0871s	0.1840s	36.5656s

From the table 5.3, as the value m becomes larger, the elapsed time becomes longer.

If we keep m=4 as Constant value by changing the error capability we get the following time intervals in seconds:

For t=1, the elapsed time is 0.0310 s.

For t=2, the elapsed time is 0.0320s.

For t=3, the elapsed time is 0.0470s.

As the number of error capability increases, the elapsed time also increases.

5.5. Transmission cost

Our judgment is based on the size of the MAC which are transmitted from the sender to receiver.

In our scheme we only send 2t symbols instead of a MAC.

5.6. Storage cost

In our scheme an extra buffer space is required to store the intermediate variables and additional delay is introduced.

6. Conclusions and further research

6.1. Conclusions

In this study, we have made use of Reed Solomon code for reducingthe overhead message transmission costs regarding MAC transmission of messages.

The results arrived at higher authentication of messages and it demands reduced transmission costs by making use of *t*-error correcting Reed Solomon code as long as the adversary does not inject more than false messages at the time.

This is suitable for sensor applications since the energy consumption is of great concern.

To the best of our knowledge, our schemeis the first one that uses error control coding concepts to handle the authentication and reduction of transmission cost problems in message transmission.

Based on RS code itself, the scheme guarantees that the receiver can detect a false report when no more than 2t symbols are compromised, where *t* is a security threshold.

6.2. Further Research Topic and Directions

Despite the fact that we present a new scheme to take into account for any application where message transmission is involved, there is still open questions to achieve the Galois Field with large elements, 2^{196} elements for instance, so that each MAC would be an element in the finite field which will be a huge step to reduce false acceptance.

As future work, several directions are worth investigating. In particular, we may use soft decision decoding. Another topicthat we plan to address is how our scheme can be adapted for state machine concepts.

References

[Hump96] Humphreys, J., A course in group theory, *Oxford* University *Press, USA,* 1996

[Kasa84] Kasami, T. &Lin, S., On the probability yo fun detected error for the maximum distance separable codes, *Communications, IEEE Transactionson [legacy, pre-1988],* 1984, 32, 998-1006

[Mceli04] McEliece, R., The theory of information and coding, **Cambridge University***Press,* 2004

[Mceli04] McEliece, R., The theory of information and coding, **Cambridge University***Press,* 2004

[Mene11] Menezes A. J. etall, "Handbook of Applied Cryptography", CRC Press **2011**

[Paar10] Paar, C.Pelzl J. Understanding Cryptography: A Textbook for StudentsandPractitioners, **2010**, 370ps

[Reed60] Reed, I. &Solomon, G., Polynomial codes over certain finite fields, *JournaloftheSociety for Industrial andAppliedMathematics, Society for Industrial andAppliedMathematics,* 1960, 300-304

[Shu83] Shu. L.Costello, D. ErrorControlCoding: Fundamentals andApplications, EnglewoodCliffs, N.J.: Prentice-Hall, 1983, 1520ps.

[Xioa10] Xiao, Y. Chen H. Li, F. H. Handbook on sensor Networks, World Scientific Publishing Co. Pte. Ltd., 2010, 882ps.

[Wand05] Wander, A. Gura, N. Eberle, H. Gupta, V. Shantz, S.Energy analysis of public key cryptography for wireless sensor network, 2005, 5ps