Message Authentication Codes with Error Correcting Capabilities

Charles C.Y. Lam
Department of Combinatorics & Optimization
University of Waterloo
Waterloo, Ontario N2L 3G1, CANADA
cy3lam@cacr.uwaterloo.ca

Guang Gong
Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario N2L 3G1, CANADA
ggong@cacr.uwaterloo.ca

Scott Vanstone
Department of Combinatorics & Optimization
University of Waterloo
Waterloo, Ontario N2L 3G1, CANADA
savanstone@cacr.uwaterloo.ca

June 20, 2002

Abstract

In this paper, we propose classes of Message Authentication Codes (MAC) based on error correcting-codes. We introduce a new notion of error tolerant forgery of hash messages. The classes of the keyed hash functions are highly secure, and provide the capabilities of correcting errors on transmission, including burst-errors, which is a typical phenomenon in wireless communications. These classes of hash functions are easily implementable in hardware by means of simple linear feedback shift register structures.

1 Introduction

Keyed hash functions, also known as *message authentication codes* (MAC) in the literature [9], provide security service of data integrity and data origin authentication in network communications. However, due to the characteristics of transmission channels, data may be corrupted by errors occurred during the communication process. In this case, the verification for integrity check of data would not be possible. Thus, it is interesting to construct MACs which are tolerant to a few errors that may occur during transmission. The purpose of this paper is to construct MACs which allow at most 2 errors or one segment of burst error of length n to occur during communication, by means of the well known BCH codes and Reed-Solomon codes [8, 10].

Applying error correcting codes to construct MACs was first exploited by Krawczyk [7] in Crypto '94. In that paper, Krawczyk used all distinct Hamming codes of length $2^n - 1$ as a space of keys for the MAC, and the parity check bits as the hash value of a message. He showed that the functions introduced are highly secure with a failure probability of 2^{-30} authenticating 1 Gbit of information. However, the author did not make use of the error-correcting property that comes with the construction.

In our scenario, we will consider the scheme where the MAC will perform error correcting operations, which we will call an *error correcting MAC*. We will show that this scheme possess high security strength. Note that this MAC will be error-correcting, and unless the messages transmitted is not information sensitive, such a MAC is not *error tolerant*, and an incoming MAC must be error-corrected.

Traditionally, error correcting codes are used the transmit a MAC by first computing the MAC and then adding error correcting codes for transmission. However, due to the way current error correcting codes are applied, the message is first divided into many small blocks, and then error correcting codes are applied, resulting in large message expansion. In our MAC, the error correcting code is embedded into the MAC, and although not capable of correcting the amount of errors that a regular error correcting code can process, can still correct burst errors of a respectable length, without significant message expansion.

The rest of the paper is organised as follows. In Section 2, we give a construction based on BCH codes. We analyse the security in Section 3.1 and the complexity and implementation issues in Section 3.2. In Section 4, we introduce the construction using Reed-Solomon codes, and then discuss the related security, complexity and implementation issues.

To conclude this section, we will give a formal definition of classes of hash functions, and the construction of the proposed MACs given in [7].

Definition 1 *An (m,n)-family H of hash functions over a finite field $\mathbb{F}$ is a collection of functions that map the set of strings of length m over $\mathbb{F}$ into the set of strings of length n over a $\mathbb{F}$.*

For simplicity, we assume a communication scenario in which two parties communicate over an unreliable channel with a malicious adversary in the middle. The secret key used in the communication is unknown to the adversary. The parties exchange using the secret key for only one message of length m, where the secret key is a hash function h drawn randomly from the (m,n)-family of hash functions and a random pad r of length n. Let $\mathbb{Z}_2^n$ be the vector space of dimension n over GF(2). We consider the messages M to be vectors over $\mathbb{Z}_2^m$ and the hash values to be vectors over $\mathbb{Z}_2^n$. The sender sends the message M together with the tag $t = h(M) + r$, and is verified at the receiver by recalculating t.

Construction 1 *[7, Section 3.1] Let $H_{m,n}$ be an (m,n)-family of hash functions over $\mathbb{Z}_2$ defined as follows. Let $f(x)$ be an irreducible polynomial of degree n over* GF(2)*. Let M be a message of predetermined binary length m, and $M(x)$ be the polynomial over* GF(2) *with coefficients corresponding to the bits of M. That is, if $M = (s_{m-1}, s_{m-2}, \ldots, s_0)$, then $M(x) = s_{m-1}x^{m-1} + s_{m-2}x^{m-2} + \cdots + s_1x + s_0$. We associate the hash function h_f for a message M to be*

$$h_f(M) = M(x) \cdot x^n \mod f(x).$$

The construction, if considered as a codeword in error-correcting codes context for a fixed irreducible polynomial $f(x)$, is a Hamming code [8, 10] of distance 3, thus is able to correct up to 1 error upon transmission.

2 Error Correcting MAC Based on BCH Codes

In this section, we first give a definition for the error correcting MAC, then we give a detailed construction of such a MAC using binary BCH codes with designed distance 5.

Construction 2 *Let $H_{m,l}$ be an (m,l)-family of hash functions over $\mathbb{Z}_2$ defined as follows. Let C be a binary BCH code of length $2^n - 1$ with minimal* *distance $2e+1$ and $g(x)$ be the parity check polynomial of C of degree l. Let M be a message of predetermined binary length m, and $M(x)$ be the polynomial over* GF(2) *with coefficients corresponding to the bits of M. The hash function h_g associated with the polynomial g is defined as*

$$h_g(M) = M(x) \cdot x^l \bmod g(x),$$

and

$$H_{m,l} = \{h_g \mid \text{all valid parity check polynomials } g\}.$$

Let D be an error decoding algorithm for a code with minimal distance $2e+1$. D can further be decomposed into two sub-procedures: D_1 and D_2 where D_1 is the error detection algorithm (equivalently, the computation of the error locator polynomial), and D_2 is the error correction operation.

Scheme 1 (Error Correcting MAC) *The algorithm is as follows:*

Sender: To send a message M of length m,

Step 1. Select $h_g \in_R H_{m,l}$.

Step 2. Choose a random pad $u \in \mathbb{Z}_2^l$.

Step 3. Compute

$$r(x) = h_g(M) = M(x) \cdot x^l \bmod g(x)$$

and set

$$t(x) = r(x) + u(x).$$

Step 4. Convert $t(x)$ into the corresponding bit string of length l and then send the message-tag pair (M, t).

Receiver: Suppose the received message-tag pair is (M', t').

Step 1. Compute

$$h_g(M') = M'(x) \cdot x^l \bmod g(x).$$

Verify that

$$t'(x) + u(x) = h_g(M') \tag{1}$$

If (1) is true then accept the pair (M', t') as valid (we have $M' = M$ and $t' = t$), in this case no errors occurred during transmission.

Step 2. Otherwise, remove the random pad from t', let

$$r' = t' + u.$$

Apply D_1 to (M', r'). If D_1 returns $e + 1$ errors or more, then reject the pair (M', t').

Step 3. If (M', r') has at most e errors, apply D_2 to find the error locations detected in D_1. Let (M'', r'') be the resulting pair, that is,

$$(M'', r'') = D_2 \circ D_1(M', r')$$

where "$\circ$" represents the composition of the two operators D_1 and D_2. If the error location are found to be outside of the length of the message-tag pair, when the word itself is considered to be the codeword of length $2^n - 1$ in the sense of a full BCH codeword, then reject the message, otherwise accept (M'', r'') as the valid message-tag pair.

Definition 2 *Let $d(A, A')$ represent the Hamming distance of two codewords A, A' of the same length. If $A = (M, t)$ is a correct message-tag pair, and $A = (M', t')$ is another message-tag pair of the same length, such that*

$$d(A, A') \leq e,$$

then $A' = (M', t')$ is considered as an equivalent acceptable pair, corresponding to the same message-tag pair $A = (M, t)$.

Under this definition, it seems that we have violated the definition of a secure MAC. However, if we allow error correcting capability for a MAC, such a definition will be necessary. Hence, we do not consider replacing a correct message-tag pair A with an equivalent acceptable message-tag pair A' which is of Hamming distance less than e from A as a message attack, since A' is correctable to A, thus it is considered as the same message.

In the following we will discuss in detail Scheme 1 by using the BCH code with designed distance 5.

Construction 3 *Let $n \geq 7$ be an odd integer, so that $\gcd(3, 2^n - 1) = 1$. Let $H_{m,2n}$ be a $(m, 2n)$-family of hash functions defined as follows. Let α be a primitive element in $\mathrm{GF}(2^n)$ and $f_\alpha(x)$ be its minimal polynomial of degree n over $\mathrm{GF}(2)$, where α is a root over $\mathrm{GF}(2^n)$. Let $f_{\alpha^3}(x)$ be the minimal polynomial of degree n for α^3 over $\mathrm{GF}(2)$, and let $g_\alpha(x) = f_\alpha(x) f_{\alpha^3}(x)$. The hash function h_g associated with the polynomial g for the message M is defined as*

$$h_{g_\alpha}(M) = M(x) \cdot x^{2n} \bmod g_\alpha(x).$$

Scheme 2 (2-error Correcting MAC) *The algorithm is as follows:*

Sender: For a message M:

Step 1. Select in random a primitive element $\alpha \in \mathrm{GF}(2^n)$ and compute the minimal polynomials $f_\alpha(x)$ and $f_{\alpha^3}(x)$. Set

$$g_\alpha(x) = f_\alpha(x) f_{\alpha^3}(x).$$

Step 2. Select $r \in_R \mathbb{Z}_2^{2n}$

Step 3. Compute the hash value of M:

$$h_{g_\alpha}(M) = M(x) \cdot x^{2n} \bmod g_\alpha(x).$$

Receiver: For the message-tag pair (M', t'):

Step 1. Form the polynomial

$$s(x) = M'(x) \cdot x^{2n} + t'(x) + r(x).$$

Compute

$$\begin{aligned} A_1 &= s(\alpha), \\ A_3 &= s(\alpha^3). \end{aligned}$$

The receiver will accept (M', t') if $A_1 = A_3 = 0$. There are no errors occurred during transmission.

Step 2. Otherwise errors may have occurred. If $A_1 \neq 0$ and $A_3 = A_1^3$, we set the error locator polynomial as

$$\sigma(z) = 1 + A_1 z.$$

Else we set

$$\sigma(z) = 1 + A_1 z + (\frac{A_3}{A_1} + A_1^2) z^2.$$

Step 3. Solve the error locator polynomial for roots z_1 (and z_2, depending on the degree of the polynomial). Let $x_i = z_i^{-1}$. The discrete logarithm of x_i with respect to α are the locations of the errors. If the error locations are out of range then we reject the message-tag pair. Otherwise let E be the error vector and let

$$M'' = M' + E,$$

and accept the message-tag pair as valid with corrected message M''.

Remark: Note that the random pad $r \in_R \mathbb{Z}_2^{2n}$ is needed. Otherwise, a simple attack by taking the polynomial factorization $s(x)$ of a correctly transmitted s will likely lead to the factors of $g_\alpha(x)$.

Remark: Although the MAC proposed includes the parameter of the length of the message, it is noted that the length m is only used in security analysis and decoding while the hash functions are only dependent on the polynomial $g_\alpha(x)$. The length, however, is to be noted during transmission, for the reason of error correction and that otherwise forgery is much more plausible. We will discuss this issue in the following section, after security analysis.

Remark: Solving the discrete logarithm of x_i seems to be a computationally infeasible operation in the field GF(2^n) for large n. However, since the length of the received word (M, t) is known, calculating the discrete logarithm over a much smaller bounded continuous region requires much less work. We will discuss the complexity of the MAC generation, verification, and error correction in Section 3.2.

Remark: Note that our construction applies to BCH codes of larger distances. For example, in literature, there are two types of BCH codes which give a minimal distance of 7. The first one [8, 10] is to take the polynomial with designed distance 7, where

$$g(x) = f_\alpha(x) f_{\alpha^3}(x) f_{\alpha^5}(x),$$

and the other [3] is to take

$$g(x) = f_\alpha(x) f_{\alpha^d}(x) f_{\alpha^{d^2}}(x),$$

where n is odd and $d = 1 + 2^{\frac{n-1}{2}}$.

3 Security and Complexity Analysis of Error Correcting MAC through BCH Codes of Distance 5

Using the construction through BCH codes, we will show the security aspects of the class $H_{m,2n}$. The following is given by Krawczyk in [7]. The security analysis for general BCH codes is similar. The complexity and implementation issues follow.

3.1 Security Analysis

Definition 3 *A family of functions $H_{m,l}$ over $\mathbb{F}$ is linear if for all $M, M' \in \mathbb{F}^m$ we have $h(M + M') = h(M) + h(M')$, where M, $M' \in \mathbb{F}^m$ and $h(M)$, $h(M') \in \mathbb{F}^l$.*

Definition 4 *A family $H_{m,l}$ of hash functions over $\mathbb{F}$ is called ϵ-balanced if for all $M \neq 0$, $M \in \mathbb{F}^m$, $c \in \mathbb{F}^l$*

$$\mathrm{Pr}_h(h(M) = c) \leq \epsilon.$$

It can be easily shown that our class of hash functions $H_{m,2n}$ over $\mathbb{Z}_2$ for any fixed m and n is linear and ϵ-balanced. Let $w(\cdot)$ represent the hamming weight of a string.

Because of the ability of error correcting, we consider any vector of length $m + l$ within hamming distance e from a correct message-tag pair (M, t) to be in the same *equivalence class*. That is, the messages within distance e of (M, t) are all acceptable message-tag pairs corresponding to the same correct message (M, t).

Definition 5 *A hash function $h_g \in_R H_{m,l}$ over $\mathbb{F}$ is e-error-forgery-resistant with probability ϵ if and only if given any message-tag pair within distance e of a correct pair (M, t), where $M \in \mathbb{F}^m$, $t = h_g(M) + r \in \mathbb{F}_l$, $r \in_R \mathbb{F}^l$, no adversary succeeds by finding some M' of length m and t' of length l, with $M' \neq M$, $d(M', M) > e$ such that (M', t') is an acceptable pair with probability larger than ϵ.*

Theorem 1 *A family $H_{m,l}$ of hash functions is e-error-forgery-resistant with probability ϵ if and only if*

$$\forall c, M_1 \neq M_2, d(M_1, M_2) > e, \sum_{e' \in \mathbb{F}^l, w(e') \leq e} \mathrm{Pr}_h(h(M_1) + h(M_2) = c + e') \leq \epsilon.$$

The proof of the above statement is similar as in Theorem 5 in [7], except we allow errors to be introduced in the transmitted word.

Proof: We consider all the vectors within distance e from a correct message-tag pair (M, t) to be in the same equivalent class. The pair (M_1, t_1) is successfully replaced by the pair (M_2, t_2) if and only if for the secret function h and the random vector r we have $t_1' = h(M_1') + r$ and $t_2 = h(M_2) + r + e'$, where M_1' and t_1' are the error-free message-tag pair associated with M_1 and

t_1, and $e' \in \mathbb{F}^l$ with $0 \leq w(e') \leq e$. Note that the error e' already includes the possibility of having errors in the message part. Equivalently, we have $c = t_1' + t_2 + e' = h(M_1' + M_2)$. Hence, the probability of success for an adversary is bounded by

$$p = \sum_{e' \in \mathbb{F}^l, w(e') \leq e} \max_{M_1, M_2, c} \Pr_h(h(M_1) + h(M_2) \in \mathcal{C}_c),$$

where $\mathcal{C}_c$ is the equivalent class of vectors within distance e of c. Notice that this success probability is achievable whenever the transmitted message is one of the messages in which the maximum is attained, by replacing (M_1', t_1') by $(M_2, t_1' + c)$. Since the correct words (M, t) are distance at least e apart, we must have

$$p \leq \epsilon.$$

■

Similar to [7], we can draw the following consequence.

Theorem 2 *If $H_{m,l}$ is linear then $H_{m,l}$ is e-error-forgery-resistant with probability ϵ if and only if $H_{m,l}$ is ϵ-balanced.*

Theorem 3 *For any values of m and n, the family of hash functions $H_{m,2n}$ in Construction 3 is linear and ϵ-balanced for*

$$\epsilon \leq \frac{6(m+2n)\ln\ln(2^n-1)}{2^n-1},$$

and 2-error-forgery-resistant with probability ϵ.

Proof: The family of hash functions is obviously linear. Now given any polynomial $g_\alpha(x) = f_\alpha(x) f_{\alpha^3}(x)$ where α is a primitive element in $\mathrm{GF}(2^n)$, any non-zero message M of length m and any string c of length n,

$$\begin{aligned} h_g(M) = c \quad &\Longleftrightarrow \quad M(x) \cdot x^{2n} \bmod g_\alpha(x) = c(x) \\ &\Longleftrightarrow \quad g_\alpha(x) | (M(x) \cdot x^{2n} - c(x)). \end{aligned}$$

Note that $q(x) = M(x) \cdot x^{2n} - c(x)$ is a non-zero polynomial of degree at most $m + 2n$. Thus there are at most $\frac{m+2n}{n}$ factors of primitive polynomials of degree n in $q(x)$. Therefore, there are at most $\frac{m+2n}{n}$ possible hash function candidates. Let $\phi(\cdot)$ be the Euler Phi function. While there are $\frac{\phi(2^n-1)}{n}$ possible hash functions in this family, we have the probability

$$\Pr(h_g(M) = c) \leq \frac{(m+2n)/n}{\phi(2^n-1)/n} = \frac{m+2n}{\phi(2^n-1)}.$$

Using the estimate

$$\phi(2^n - 1) > \frac{2^n - 1}{6 \ln \ln (2^n - 1)},$$

the result follows. Note that if the received message-tag pair is error-prone within distance 2 of a correct code word, the factorisation of $q(x)$ will not contain all of the factors of $g_\alpha(x)$, thus lowering the probability value ϵ. ■

For $\epsilon < 2^{-80}$, $n \geq 93$ is required. Below is a table for the values of m and n which guarantee a probability $\epsilon < 2^{-80}$.

	$\epsilon < 2^{-80}$
$n = 100$	$m < 41031$
$n = 120$	$m < 4.14 \times 10^{10}$
$n = 140$	$m < 4.19 \times 10^{16}$

For $\epsilon < 2^{-160}$, $n \geq 174$ is required. Below is a table for the values of m and n which guarantee a probability $\epsilon < 2^{-160}$.

	$\epsilon < 2^{-160}$
$n = 180$	$m < 35849$
$n = 200$	$m < 3.71 \times 10^{10}$
$n = 220$	$m < 3.82 \times 10^{16}$

In practice, two communicating parties would exchange multiple messages, one particular hash function h_g can be used for the multiple messages, as long as a different random pad r is used for each message. In fact, since a random pad is applied, it is useless that the adversary sees many message-tag pairs.

In the practice of error correcting codes, the error correcting property also mean the capability to be able to *detect* errors without correcting. However, in this MAC, the procedure of error correcting *must* be applied. The reason being that the security of the MAC is dependent on the length of the message-tag pair. If error detection is done without error-correcting, this would allow a message of a different length (typically much longer one) of short Hamming distance to be accepted as a valid message-tag pair. In other words, it is highly likely that the any pair $(M, t) \in \mathbb{Z}_2^{m+2n}$ such that when considered over a codeword of length $2^n - 1$ under the normal sense of the BCH codeword $C_1 \in \mathbb{Z}_2^{2^n-1}$, where

$$C_1 = (\underbrace{0, 0, \cdots 0}_{2^n-1-m-2n}, \underbrace{M, t}_{m+2n})$$

is within distance 2 of a codeword such as

$$C_2 = (\underbrace{0,\cdots 0,\times,0,\cdots 0,\times,0,\cdots 0}_{2^n-1-m-2n},\underbrace{M,t}_{m+2n}) \in \mathbb{Z}_2^{2^n-1}.$$

However, the restriction of the length of the message limits the probability (defined by ϵ) that such an attack can happen. This also illustrates the reason for the upper limit on the length m of the message for any fixed ϵ and n.

Remark: In Construction 3, we propose to use $g_\alpha(x) = f_\alpha(x) f_{\alpha^3}(x)$. There are in fact more choices $g(x) = f_\alpha(x) f_{\alpha^k}(x)$ which give a BCH code of minimal distance at least 5. While there are no results on which values of k or the proportion of such polynomials that give codes of distance at least 5, any such results would greatly improve the probability value ϵ, and thus increase the possible number of hash functions in the class.

3.2 Complexity and Implementation

In a typical communication session, such a MAC algorithm requires a predetermined choice of a primitive element $\alpha \in \mathrm{GF}(2^n)$, together with a pseudorandom bit string r of length $2n$, which together forms our secret key. The polynomial $g_\alpha(x)$ is therefore $f_\alpha(x) f_{\alpha^3}(x)$. The message-tag pair (m,t) is then computed and sent over an error-prone channel. The receiver calculates the tag using $g_\alpha(x)$. If the tag does not match, the receiver may first attempt to error correct the message-tag pair by considering $(m, t+r)$ as a codeword with limited length $m+2n$. If the error correcting is successful, the receiver will accept the message-tag pair. Note that if the error location is out of the range of the $m+2n$ bits, then the error correction fails, thus the receiver may reject the message-tag pair.

The generation of the message-tag pair requires a multiplication by x^{2n} followed by an operation of the division algorithm. It is easy to see that the division algorithm takes $2mn$ bit operations over $\mathrm{GF}(2)$. The implementation of this operation is simple over software. In hardware, such an operation can be implemented through a linear feedback shift register (LFSR) with architecture allowing input of a variable $g_\alpha(x)$.

The more computationally heavy operation is the error-correcting part. We are required to first solve a quadratic equation over $\mathrm{GF}(2^n)$ and then compute at most two discrete logarithms of type $x = \log_\alpha \beta$ where α is a primitive element of $\mathrm{GF}(2^n)$ (specified by $g_\alpha(x)$) and $0 \le x \le m+2n-1$ (recall that if x is outside of this range then we reject the message). Finding the roots of a quadratic equation is easy with an algorithm outlined in IEEE P1363 [5]. As for the part of solving the discrete logarithm problem, an exhaustive search for the discrete logarithms for a short length message-tag pair is feasible as such an operation requires only $m+2n$ multiplication operations in $\mathrm{GF}(2^n)$, which is relatively fast by choosing an appropriate basis representation. If, say, a baby-step-giant-step algorithm is used for decoding on a device with abundant memory storage, such an operation requires $\sqrt{m+2n}$ storage of group elements, $2\sqrt{m+2n}$ multiplications, and at most $\sqrt{m+2n}$ table lookups.

The key size of the hash functions in the use as a MAC is $3n$, including n bits for the element α and $2n$ bits for the pseudorandom pad r.

4 Error Correcting MAC Based on Reed-Solomon Codes

In this section, we give another construction by applying Reed-Solomon Codes. Let $\alpha \in \mathrm{GF}(2^n)$. A Reed-Solomon code with designed distance δ has generator polynomial

$$g(x) = (x-\alpha^b)(x-\alpha^{b+1})\cdots(x-\alpha^{b+\delta-2}).$$

The resulting code has a distance of at least δ over $\mathrm{GF}(2^n)$.

4.1 Construction

We can map a binary message of length $m = kn$ where $k \in \mathbb{Z}$ into a polynomial over $\mathrm{GF}(2^n)$ by dividing up the message into n-bit blocks and consider each as an element in $\mathrm{GF}(2^n)$. For example, if a polynomial basis is used, then each bit in the n-bit block can be considered as the coefficient to the linear combination of the ordered basis elements, and the whole n-bit block is one element represented by the linear combination. Padding is necessary if the message length is not a multiple of n. However, caution must be taken in the method for padding.

Construction 4 *Let $m = kn$ and $H_{k,\delta-1}$ be a family of hash functions defined over $\mathrm{GF}(2^n)$ as follows. Let α be a primitive element in $\mathrm{GF}(2^n)$. Let $1 \le b \le 2^n - 1$ where $\gcd(b, 2^n-1) = 1$. Let $\delta = 2e+1$ for some positive integer e. Let*

$$g_{\alpha,b,\delta}(x) = (x-\alpha^b)(x-\alpha^{b+1})\cdots(x-\alpha^{b+\delta-2}).$$

Let $M(x)$ be the polynomial over $\mathrm{GF}(2^n)$ of degree k associated with the message M of binary length m by a bijective mapping as described above. We associate a hash function h_g for the message M to be

$$h_g(M) = M(x) \cdot x^{\delta-1} \bmod g_{\alpha,b,\delta}.$$

The above construction leads us to a hash function capable of correcting *burst errors.* We can consider the message-tag pair not only as a codeword over $\mathrm{GF}(2^n)$ but also over $\mathbb{Z}_2$ by considering each element represented over a certain basis and the coefficients over $\mathbb{Z}_2$ of the linear combination as message bits (using the same bijective mapping as described). The result is an $H_{m,(\delta-1)n}$ family of hash functions. The following is a standard result from coding theory.

Definition 6 *A cyclic burst of length t is a vector whose non-zero components are within t successive entries (cyclically), and which the first and the last entry are non-zero.*

Fact 1 *The resulting message-tag pairs (M, t) from Construction 4 is capable of correcting up to e errors when considered as a word over $\mathrm{GF}(2^n)$, and correcting a burst error of length up to $n(e-1)+1$ when considered as a word over $\mathbb{Z}_2$.*

Definition 7 *A hash function is e-burst-error-forgery-resistant with probability ϵ if and only if given any message-tag pair within a burst error of length e of a correct pair (M, t), where M is of length m, $t = h_g(M) + r$ of length $(\delta-1)n$, $h_g \in_R H_{m,(\delta-1)n}$, $r \in_R \mathbb{Z}_2^{(\delta-1)n}$, no adversary succeeds by finding some M' of length m and t' of length $(\delta-1)n$, with $M' \neq M$, and $(M', t') - (M, t)$ being a burst error vector of length $> e$, such that (M', t') is an acceptable pair with probability larger than ϵ.*

4.2 Security Analysis

Theorem 4 *The family of hash functions $H_{k,n}$ in Construction 4 is*

1. *linear*
2. *ϵ-balanced where*

$$\epsilon \leq \frac{\begin{pmatrix} k+\delta-1 \\ \delta-1 \end{pmatrix}}{(\phi(2^n-1))^2}.$$

3. *e-error-forgery-resistant over* $\mathrm{GF}(2^n)$ *with probability* ϵ.

4. $n(e-1)+1$*-burst-error-forgery-resistant over* $\mathbb{Z}_2$ *with probability* ϵ *(when considered as the family* $H_{kn,n(\delta-1)}$*).*

Proof: The proof is similar to Theorem 3 except now that

$$q(x) = M(x) \cdot x^{\delta-1} - c(x)$$

may split into linear factors. Since our choice of α can be any primitive element in $\mathrm{GF}(2^n)$ and the choice of b is any integer relatively prime to $2^n - 1$, we have $(\phi(2^n-1))^2$ different functions with respect to any given δ. ■

Below is a table for the values of n and $k = \frac{m}{n}$ which guarantee a probability $\epsilon < 2^{-80}$.

	$\delta = 5$	$\delta = 7$	$\delta = 9$
$n = 80$	$k < 1.52 \times 10^6$	$k < 23301$	$k < 3111$
$n = 120$	$k < 1.39 \times 10^{12}$	$k < 2.20 \times 10^8$	$k < 2.99 \times 10^6$
$n = 160$	$k < 1.67 \times 10^{18}$	$k < 2.48 \times 10^{12}$	$k < 3.27 \times 10^9$

4.3 Complexity and Implementation Issues

The implementation of decoding Reed-Solomon codes have been discussed thoroughly in literature [1, 2, 8, 10, 6, 12].

Due to the similar reason as in the MACs using BCH codes, it is necessary to compute the error locations during decoding. However, for applications to less information-sensitive transmissions, such as voice and image signals, a few errors occurred during transmission is not crucial. Therefore, the step to correct the errors may not be required and we can simply take the received message M as the accepted message if error locations are found to be within the range of the length of M.

The major computation problem here we are faced with, again, is the solution of the discrete logarithm with base α over a limited region of length $k + \delta - 1$ for the location of error. The computational aspect is similar as in the BCH case.

However, if only burst errors are corrected, there is a much simpler algorithm which requires only $k + \delta - 1$ clock cycles of operations. This is discussed in the following section.

4.4 Error-Trapping Method for Burst Errors Decoding

The method is described in [10], here we give a brief explanation of the algorithm.

Let $e(x)$ be the error polynomial. Let

$$s_j(x) \equiv x^{-j}e(x) \pmod{g_{\alpha,b,\delta}(x)}.$$

If $s_j(x)$ is a polynomial of burst length e or less, then

$$e(x) = x^j s_j(x)$$

is the error polynomial.

The implementation of this decoding method is done by calculating

$$s_j(x) \equiv x^{-j}r(x) \pmod{g_{\alpha,b,\delta}(x)}$$

starting from $j = 0$. Note that for each $j \geq 1$, $s_j(x)$ can be obtained from a shift operation of the polynomial from $s_{j-1}(x)$, plus an addition of a polynomial over GF$(2^n)[x]$ of degree δ (which is the inverse of x with respect to $g_{\alpha,b,\delta}(x)$). We stop when there are no such bursts within $s_0(x), \ldots, s_{k+\delta-1}(x)$. Hence this operation takes $k + \delta - 1$ clock cycles of simple XOR operations in hardware.

5 Conclusion

We have provided an error correcting MAC which can provide integrity check for corrupted data which corrects at most 2 errors or an $n(e-1)+1$-bit burst error during transmission. In the process, we introduced the notion of error-tolerant forgery of hash messages. The security of such MACs remain high, and can be applied to less information-sensitive wireless transmissions such as voice communications. It remains open research problems to increase the size of key space, to efficiently decode BCH and Reed-Solomon codes over a large field, and to solve the discrete logarithm problem over a short continuous interval. It is also desirable to find classes of error correcting codes that locate errors without having to solve the discrete logarithm problem.

References

[1] E.R. Berlekamp, *Algebraic coding theory*, McGraw-Hill, New York, 1968.

[2] R.E. Blahut, *Theory and Practice of Error Control Codes*, Addison-Wesley Publishing Co., May 1984.

[3] A. Chang, P. Gaal, S.W. Golomb, G. Gong, T. Helleseth, P.V. Kumar, *On a Conjectured Ideal Autocorrelation Sequence and a Related Triple-Error Correcting Cyclic Code*, IEEE Trans. Information Theory, Vol. 46, No. 2, March 2000.

[4] S.W. Golomb, *Shift Register Sequences,* Holden-Day, Inc., 1967. Revised edition, Aegean Park Press, 1982.

[5] IEEE P1363, *Standard Specifications For Public Key Cryptography*, 2000.

[6] J.H. Jeng and T.K. Truong, *on Decoding of Both Errors and Erasures of a Reed-Solomon Code Using an Inverse-Free Berlekamp-Massey Algorithm*, IEEE Transactions on Communications, Vol. 47, No. 10, October 1999, 1488-1494.

[7] H. Krawczyk, *LFSR-based Hashing and Authentication*, Advances in Cryptology–Crypto '94, Lecture Notes in Computer Science, LNCS 839(1994), Springer-Verlag, 129-139.

[8] F.J. MacWilliams, N.J.A. Sloane, *The Theory of Error-Correcting Codes*, North-Holland Publishing Company, 1977.

[9] A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

[10] S.A. Vanstone, P.C. van Oorschot, *An Introduction to Error Correcting Codes with Applications*, Kluwer Academic Publishers, 1989.

[11] E. Bach, J. Shallit, *Algorithmic Number Theory*, MIT Press, Cambridge, Massachusetts, 1996.

[12] T.K. Truong, J.H. Jeng, I.S. Reed, *Fast Algorithm for Computing the Roots of Error Locator Polynomials up to Degree* 11 *in Reed-Solomon Decoders*, IEEE Transactions on Communications, Vol. 49, No.5, May 2001, 779-783.