MRD Hashing

Rei Safavi-Naini *, Shahram Bakhtiari, and Chris Charnes

School of IT and CS, University of Wollongong, Northfields Ave
Wollongong 2522, Australia
[rei, shahram, charnes]@uow.edu.au

Abstract. We propose two new classes of hash functions inspired by Maximum Rank Distance (MRD) codes. We analyze the security and describe efficient hardware and software implementations of these schemes. In general, the system setup remains computationally expensive. However, for a special class of parameters we show that this computation can be completely avoided.

1 Introduction

A *Message Authentication Code (MAC)* is a symmetric key cryptographic primitive that ensures message integrity against active spoofing. A MAC consists of two algorithms. A *MAC generation* algorithm, G_k, that takes an arbitrary message s from a set $\mathcal{S}$ of messages and produces a *tag*, $t = G_k(s)$. The tag is appended to the message to produce an *authenticated* message, $m = s\|t$. A *MAC verification* algorithm takes an authenticated message of the form $s\|t$, and produces a *true/false* value depending on whether the message is authentic or fraudulent. The secret key k is only known to the legitimate communicants and hence a valid tag can only be computed by them. An outsider tries to generate a fraudulent message that is acceptable by the receiver.

Computationally secure MACs are usually constructed from hash functions by using a secret key during the hashing process. Secure MACs are constructed by examining the computational complexity of various attacks [10,11], and by choosing the system parameters so that the complexity of the best attack exceeds the computational resources of the enemy. Constructions of these type are always subject to revision as new attacks become available.

In provably secure MACs an intruder has a provably small chance to tamper with the message, and no limit on the computational resources of the enemy is assumed. Wegman and Carter [16] investigated unconditionally secure MACs and gave a construction for a key-efficient MAC with provable security. This construction uses an ϵ-almost strongly universal$_2$ (ϵ-ASU$_2$) class of hash functions. By Stinson's [14] composition method a new ϵ-ASU$_2$ can be created starting with a given ϵ-ASU$_2$ class and an ϵ-AU$_2$ class. Hence the question of constructing secure MACs reduces to the problem of constructing computationally efficient ϵ-AU$_2$ functions.

* Support for this project was partly provided by Australian Research Council.

S. Vaudenay (Ed.): Fast Software Encryption – FSE'98, LNCS 1372, pp. 134–149, 1998.
© Springer-Verlag Berlin Heidelberg 1998

Wegman and Carter's construction was refined by Krawczyk [6], who showed that a secure MAC only requires an ϵ-almost XOR universal$_2$ (ϵ-AXU$_2$) hash function.

In this paper we make two contributions. Firstly, we introduce two new classes (ϵ-AU$_2$ and ϵ-AXU$_2$) of hash functions, which are inspired by MRD codes and demonstrate their efficiency it terms of key size and the ease and speed of hashing. These schemes are examples of Shoup's [13] evaluation hashing, where polynomial evaluation over $GF(2^n)$ is replaced by matrix multiplication over $GF(2)$, resulting in a fast software implementation. These schemes have desirable properties such as small key size and flexibility in the block size of the hashed messages. However the system setup is in general a computationally expensive. Next, we completely describe the 2-polynomials in Galois fields of prime degree in which 2 is primitive. This result allows us to avoid the setup phase computation.

Section 2 has the preliminaries. In section 3 we describe the two classes of hash functions. Section 4 presents a MAC based on MRD hashing and compares it with one based on bucket hashing, and summarises the results on the 2-polynomials. Section 5 concludes the paper.

2 Preliminaries

Unconditionally secure MAC systems are especially important because they give efficient authentication systems that have provable security properties. With this approach, the security of a MAC system is assessed by the best chance that active spoofer has in constructing a fraudulent message-tag pair, $s\|t$, which is acceptable to the receiver, after observing a sequence of p message-tag pairs, $s_1\|t_1, \cdots s_p\|t_p$ sent over the communication channel.

Definition 1. *[16] A MAC for which the best chance of the enemy in the above described attack is at most ϵ, is called ϵ-secure.*

Wegman and Carter proposed the construction of a ϵ-secure MAC using ϵ-ASU$_2$ class of hash functions. Let Σ^m and Σ^n denote the set of binary strings of length m and n, respectively. A *hash function* h is a mapping from Σ^m to Σ^n. Let $\mathcal{H}$ denote a class of hash functions, $\mathcal{H} = \{h : \Sigma^m \to \Sigma^n\}$.

Definition 2. *[14] A class of hash functions, $\mathcal{H}$, is ϵ-AU$_2$ if for all $x \neq x' \in \Sigma^n$, we have $|\{h \in \mathcal{H} : h(x) = h(x')\}| \leq \epsilon|\mathcal{H}|$.*

Definition 3. *[14] A class of hash functions, $\mathcal{H}$, is ϵ-ASU$_2$ if the following two conditions are satisfied:*

- *for every $x \in \Sigma^m$ and $y \in \Sigma^n$, $|\{h \in \mathcal{H} : h(x) = y\}| = |\mathcal{H}| \times 2^{-n}$;*
- *for every pair $x, x' \in \Sigma^m$ and $x \neq x'$, and every pair $y, y' \in \Sigma^n$ we have $|\{h \in \mathcal{H} : h(x) = y, h(x') = y'\}| \leq \epsilon|\mathcal{H}| \times 2^{-n}$.*

These definitions can be expressed in terms of probabilities. For example, for an ϵ-AU$_2$ class hash function, $\Pr_h[h(x) = h(x')] \leq \epsilon$ for all $x, x' \in \Sigma^m$, $x \neq x'$.

Wegman and Carter's construction, which is the basis of our system is the following. Let $\mathcal{H}$ be an ϵ-ASU$_2$ class of hash function from Σ^m to Σ^n. The transmitter and the receiver share a secret key that consists of two parts. The first part identifies an element $h \in \mathcal{H}$, and the second part is a randomly generated one-time pad $r = r_1, r_2 \cdots$ of n-bit numbers. The transmitter and receiver maintain a counter, *count*, which is initialised *count* $:= 1$ and is incremented by one after each message. The tag value of the ℓ^{th} message, x_ℓ, is $h(x_\ell) + r_\ell$. The receiver can reconstruct this tag and verify the authenticity of the received message. Wegman and Carter proved that this construction is ϵ-secure and the key size is asymptotically optimal. Replacing the one-time pad with a pseudorandom sequence generator in the MAC, reduces unconditional security to computational security.

Definition 4. *[6] A class of hash functions is called ϵ-otp-secure if it is ϵ-secure in the one-time pad construction of Wegman and Carter.*

Krawczyk showed in [6] that provable security in the above sense does not really require the ϵ-ASU$_2$ property:

Definition 5. *[6] A class of hash functions is called ϵ-almost XOR universal$_2$, or ϵ-AXU$_2$, if for all $x \neq x' \in \Sigma^n$ and any $c \in \Sigma^m$, we have* $\Pr_h[h(x) \oplus h(x') = c] \leq \epsilon$.

Theorem 6. *[6] A class of hash functions is ϵ-otp-secure if and only if it is ϵ-AXU$_2$.*

Stinson proved that composition of hash functions can be effectively used to replace the construction of ϵ-ASU$_2$ by ϵ-AU$_2$ hash functions.

Proposition 7. *[14] Let $\mathcal{H}_1 = \{h : \Sigma^m \to \Sigma^n\}$ be ϵ_1-AU$_2$ and $\mathcal{H}_2 = \{h : \Sigma^n \to \Sigma^k\}$ be ϵ_2-ASU$_2$. Then $\mathcal{H}_1 \circ \mathcal{H}_2 = \{h : \Sigma^m \to \Sigma^k\}$ is $(\epsilon_1 + \epsilon_2)$-ASU$_2$.*

A similar result holds for composition of ϵ_1-AU$_2$ and ϵ_2-AXU$_2$ classes of hash functions [12].

The advantage of the composition method is that to achieve computationally efficient hashing, it suffices to construct an efficient ϵ-AU$_2$. This result has shifted the emphasis of research in the recent years to the construction of computationally efficient ϵ-AU$_2$ functions. Johanson [4], Taylor [15], Krawczyk [6,7], Rogaway [12], and Shoup [13] have investigated computationally efficient MACs that have relatively small key size. The most efficient construction is bucket hashing [12].

3 MRD-MAC

In this section we firstly introduce a class of functions $\mathcal{P}_{MRD}(t)$ from F_n to F_n which is inspired by MRD codes, and describe its properties. We describe two classes of hash functions: $\mathcal{H}^1_{MRD}(t)$ and $\mathcal{H}^2_{MRD}(t)$ which are based on $\mathcal{P}_{MRD}(t)$. Finally, we make some remarks on the implementation of the two classes.

In the rest of this paper we use the correspondence between binary strings $s = s_0 s_1 \cdots s_{n-1}$ and elements of $GF(2^n)$, represented as binary n-tuples $(s_0 s_1 \cdots s_{n-1})$.

3.1 $\mathcal{H}^1_{MRD}(t)$ and $\mathcal{H}^2_{MRD}(t)$

Maximum Rank Distance (MRD) codes were studied in [3]. They were used by Chen [2] for the purpose of identification, and Johansson [4] for arbitrated authentication. Although our proposed MAC system is inspired by MRD codes, we will give an independent presentation of these because we do not use directly any results from the theory of these codes. Throughout the section we assume a basic knowledge of the theory of finite fields. Some of the important definitions are included in Appendix 5. We refer the reader to [8] for an excellent introduction to this topic.

Let $L(x)$ be a linearized polynomial (more precisely a 2-polynomial; see [8]) $\underline{\beta} = (\beta_1, \beta_2, \ldots, \beta_n) = (\beta, \beta^{2^1} \cdots \beta^{2^{n-1}})$, denotes a normal basis for $F_n = GF(2^n)$. The n-tuple $(L(\beta_1), L(\beta_2), \ldots L(\beta_n))$, defines an $n \times n$ binary matrix C_L, where the i-th column is the binary representation of $\beta^{2^{i-1}}$. So a linearized polynomial $L(x)$ defines a mapping $f_L : F_n \to F_n$, given by $f_L(u) = C_L.u = v$, where '.' denotes matrix multiplication and $u, v \in F_n$ are $n \times 1$ binary vectors.

Proposition 8. *Let $u = (u_0 u_1 \cdots u_{n-1}) \in F_n$. Then $f_L(u)$ can be evaluated by finding the value of $L(x)$ at $u.\underline{\beta}$ where u and $\underline{\beta}$ are n-dimensional row and column vectors, respectively.*

The set of all mappings f_L defined above when degree of $L(x)$ is at most 2^t is denoted by $\mathcal{P}_{MRD}(t)$. The important properties of $\mathcal{P}_{MRD}(t)$ are stated in Lemma 9.

Define an array C whose rows are labelled by f_L, columns are labelled by the elements of F_n, and the entry in row f_L and column α is $f_L(\alpha)$. I.e., $C(f_L, \alpha) = f_L(\alpha)$, $\alpha \in F_n$. Let V_n denote the n-dimensional vector space over $GF(2)$. For a mapping $f : V_n \to V_n$, the *null space* $\mathcal{N}_f$, is the collection of vectors $v \in V_n$ such that $f(v) = 0$. $\mathcal{N}_f$ is a subspace of V_n in the case of linear mappings. Let 2^{d_α} denote the degree of the minimum linearized polynomial of α.

Lemma 9. *Array C has the following properties.*

1. *The column labelled by 0 contains only $0 \in F_n$. The column labelled by the all one vector contains only 0, or $1 \in F_n$.*
2. *If $\alpha \in F_n$ occurs in a row of C labelled by f_L, then it occurs $|\mathcal{N}_L|$ times, where $\mathcal{N}_L$ is the null space of f_L.*
3. *If $L(x)$ has even number of terms then $|\mathcal{N}_L| \geq 2$. In this case $f_L(x) = f_L(\overline{x})$ where $\overline{x}$ is the binary complement of x.*
4. *The number of zeros in a column of C labelled by $\alpha \in F_n$ is $M_\alpha = 2^{t-d_\alpha}$. Moreover if an element β occurs in this column, then it occurs M_α times.*
5. *For any two elements $x, y \in F_n$, $x \neq y$, the number of rows with $f_L(x) = f_L(y)$ is equal to M_{x+y}.*

The proof of this lemma is sketched in Appendix 5.

In Lemma 9, property 3 shows that two values that are complements of each other are mapped to the same value. This is an undesirable property for hashing. It can be removed by restricting $L(x)$ to polynomials with an odd number of terms, or alternatively, by restricting input to a subset F'_n of elements of F_n consisting of elements of the form $(\alpha_1, \alpha_2, \cdots \alpha_{n-1}, 0)$.

Let $d_{min} = \min_{\alpha \in F_n} d_\alpha$. We can assume that hash values of complements are distinct in view of the two proposed methods. In $\mathcal{P}_{MRD}(t)$, if we only consider linearized polynomials of degree less than $2^{d_{min}}$, then the following properties hold; they are direct consequences of Lemma 9.

Corollary 10. *Let $t < d_{min}$. Then array C satisfies the following properties.*

1. *In every row of C an element of F_n occurs at most once.*
2. *$\forall x \in F_n$ and $x \neq 0, 1$, $|\{f_L : f_L(x) = 0\}| = |\{f_L : f_L(x) = 1\}| = 0$. That is, a column labelled by $x \in F_n$ and $x \neq 0, 1$ does not contain 0 or $1 \in F_n$.*
3. *$\forall x, y \in F_n$ and $y \neq 0, 1 \in F_n$, $|\{f_L : f_L(x) = y\}| \leq 1$. That is, in a column of C, an element $x \in F_n$ and $x \neq 0, 1$ occurs at most once.*
4. *$\forall x, y, z \in F_n$ and $z \neq 0, 1 \in F_n$, $|\{f_L : f_L(x) \oplus f_L(y) = z\}| \leq 1$.*

Property 3 shows that every element of F_n occurs at most once in a column. Equivalently, two mappings give the same value when evaluated on the same element of the field. Property 4 suggests that $\mathcal{P}_{MRD}(t)$ is an ϵ-AXU$_2$ class of function.

However elements of $\mathcal{P}_{MRD}(t)$ are mappings from F_n to F_n and cannot compress the input, which is a basic requirement in hashing.

In the following we define two classes of hash functions based on the class $\mathcal{P}_{MRD}(t)$. We assume that the complementary property is removed by restricting the domain of elements of $\mathcal{P}_{MRD}(t)$ to F'_n.

Definition 11. *[6] $\mathcal{H} = \{h : \Sigma^m \to \Sigma^n\}$ is* $\oplus$-linear *if for all $h \in \mathcal{H}$ and all $x, x' \in \Sigma^m$, we have $h(x \oplus x') = h(x) \oplus h(x')$.*

$\mathcal{H}^{\mathbf{1}}_{\mathbf{MRD}}(\mathbf{t})$:

For a mapping $f_L \in \mathcal{P}_{MRD}(t)$, define a mapping $h^{(1)}_L : F'_n \times F'_n \to F_n$ as $h^{(1)}_L(x) = f_L(x_1) \oplus x_2$, where $x_1, x_2 \in F'_n$ and $x \in F'_n \times F'_n$. That is, the value of $h^{(1)}_L$ for a $2n$-tuple $x = x_1 \| x_2$ is obtained by applying f_L to x_1 and XOR-ing the result with x_2. We note that effectively x contains $2n - 2$ information bits

Theorem 12. *$\mathcal{H}^1_{MRD}(t)$ is a $\oplus$-linear ϵ-AU$_2$ class of hash function with $\epsilon = \frac{1}{2^{t+1}}$.* (Proof in Appendix 5.)

$\mathcal{H}^{\mathbf{2}}_{\mathbf{MRD}}(\mathbf{t})$:

For a mapping $f_L \in \mathcal{P}_{MRD}(t)$ and a binary string $s = s_0 s_1 \cdots$ of length $2^n - 2$ define the hash value $h^{(2)}_L(s) = \sum_{j=0}^{2^n-1} s_j f_L(\alpha^j)$ where α is a primitive element of F_n.

In other words $h^{(2)}_L(s)$ is a weighted sum of the values of $f_L(x)$ at all nonzero $x \in F_n$.

Theorem 13. *$\mathcal{H}^2_{MRD}(t)$ is a $\oplus$-linear ϵ-AXU_2 class of hash function with $\epsilon = \frac{1}{2^{t+1}}$.* (Proof in Appendix 5.)

3.2 Practical Considerations

$\mathcal{H}^1_{\mathbf{MRD}}(\mathbf{t})$

The compression ratio of this ϵ-AU_2 hash function is 2:1. To obtain higher compression ratios, the composition method of Proposition 7 can be used. We give the details in section 4.

Evaluating the hash values: is achieved by finding the product of a binary matrix C_L and a binary input vector. This can be efficiently implementated in hardware and software.

The key: is a randomly chosen binary vector of length d_{min}. Once a key $k_0k_1\cdots$ is chosen, a linearized polynomial $L(x) = \sum_{i=0}^{d_{min}-1} k_i x^{2^i}$ is determined and the matrix C_L is calculated. This is a fast computation.

Set-up phase: The main cost of the system is during the set-up phase in order to calculate d_{min} for a given n. The results of our experiment for $n < 19$ are given in Table 1. We have only considered prime values for n although theoretically this restriction is not necessary.

n	d_{min}	ϵ
5	4	2^{-4}
7	3	2^{-3}
11	10	2^{-10}
13	12	2^{-12}
17	8	2^{-8}

Table 1. *Parameters for $\mathcal{H}^1_{MRD}(t)$ for $n < 19$. d_{min} is the smallest degree of minimal linearized polynomials of all field elements, which is the same as the the number of key bits, and ϵ is the security parameter of the ϵ-AXU_2 class.*

It can be seen that some values like $n = 11$ and 13 give the maximum value for d_{min} (that is, $d_{min} = n - 1$), but others like $n = 17$ give small t. This table was produced by calculating the degree of the minimum linearized polynomial of the representative field elements of the conjugate groups. For higher values of n this approach becomes increasingly inefficient. But since it needs to be done only once during the life time of the system, the overhead of the required computation is acceptable for $n < 40$. For higher values of n, we require more efficient algorithms for finding d_{min}. In section 4.3 we present results which allows us to avoid this computation in extensions $GF(2^p)$, where p is a prime and 2 is a primitive mod p.

For practical applications a range of suitable values for n must be calculated and published. The user can then choose the n that gives the required level of

security and is most suitable for the message sizes considered.

$\mathcal{H}^{\mathbf{2}}_{\mathbf{MRD}}(\mathbf{t})$

This is an ϵ-AXU$_2$ with a compression ratio of $2^n - 1$ to n. To avoid the generation of fraudulent messages by appending extra zeros at the end of a message, we assume that s_{2^n-2} is always 1.

Implementation: can be efficiently done in hardware and software. Evaluating $h^{(2)}_L(s)$ could be similar to LFSR hashing used in [6]. Figure 1 gives a schematic diagram for this evaluation. The LFSR has fixed and public taps and produces a maximum length sequence (feedback polynomial is primitive). As the LFSR goes through consecutive states $\sigma_1, \sigma_2, \cdots$, the circuit c_L which implements the linear transformation defined by C_L produces a sequence of elements of F_n corresponding to the values of $f_L(\sigma_1), f_L(\sigma_2), \cdots$. The last part is an accumulator that calculates the weighted sum of $f_L(\sigma_i)$.

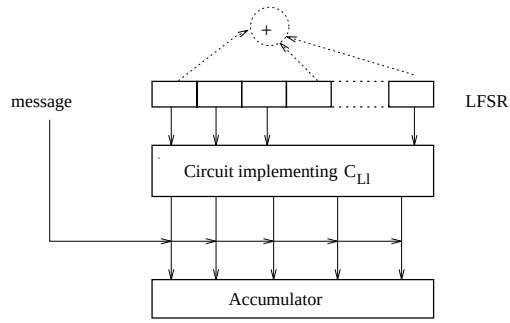

Fig. 1. *Implementation of* $\mathcal{H}^2_{MRD}(t)$

The circuit c_L can be implemented as an accumulator as shown in Figure 2.

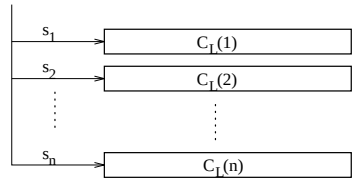

Fig. 2. *Implementation of* c_L

Each column $C_L(i), i = 1, \cdots, n$ of C_L is stored in a register and the binary n-tuple, which is the state of the LFSR, determines the subset of the registers that are XOR-ed to produce the output.

$C_L(i), i = 1, \cdots n$ form the key information (they are computable from the d_{min} key bits) and so must be kept secret.

The hashing method can also be efficiently implemented in software. Storing the consecutive outputs of the LFSR allows a parallel software evaluation of the hash function.

4 MAC from $\mathcal{H}^1_{MRD}(t)$

We use Proposition 7 to construct an ϵ-AXU$_2$ class of hash functions.

Two such classes are $\mathcal{H}_M[m,n]$ defined in [1], and $\mathcal{H}_K[m,n]$ defined in [6]. The first class maps Σ^m to Σ^n. Its elements can be described by binary $m \times n$ matrices. Let $h \in \mathcal{H}_M[m,n]$. Then $h(x) = h.x$ where h is an $m \times n$ matrix and $x \in V_n$.

Theorem 14. *[1] $\mathcal{H}_M[m,n]$ is an ϵ-AXU$_2$ class with $\epsilon = 2^{-n}$.*

The second class maps Σ^m to Σ^n. Its elements are described by irreducible polynomials of degree n over $GF(2)$. To find the hash value of $b \in \Sigma^m$ using a hash function described by $p(X)$, which is assumed to be an irreducible polynomial of degree n, a polynomial $b(X) = \sum_{i=0}^{n-1} b_i X^i$ in a formal variable X is formed. The remainder of dividing $X^n b(X)$ by $p(X)$ yields an n-tuple which is the required hash value.

Theorem 15. *[6] $\mathcal{H}_K[m,n]$ is an ϵ-AXU$_2$ class with $\epsilon = \frac{m+n}{2^{n-1}}$.*

Using proposition 7 we have the following compositions.

1. $\mathcal{H}_K[n,k] \circ \mathcal{H}^1_{MRD}(t) : F'_n \times F'_n \to \Sigma^k$ is a $(2^{1+t} + \frac{n+k}{2^{k-1}})$-AXU$_2$ class of hash function.
2. $\mathcal{H}_M[n,k] \circ \mathcal{H}^1_{MRD}(t) : F'_n \times F'_n \to \Sigma^k$ is a $(2^{1+t} + 2^{-k})$-AXU$_2$ class of hash function.

In the rest of this section we will use the second composition; in which the digest is evaluated by matrix multiplication over $GF(2)$.

4.1 A Comparison

The following comparison between the MAC described above, and the MAC obtained from Bucket Hashing (BH) gives some insight into the actual values of the parameters and application of the proposed hash functions.

In this comparison we assume the word size of 1 bit although in an efficient software implementation word size corresponds to the word size supported by the hardware. The following proposition shows that the results of our analysis are directly applicable to larger word sizes.

Proposition 16. *If $\mathcal{H} = \{h : A \to B\}$ is ϵ_1-AXU$_2$, then $\mathcal{H}' = \{h' : A^m \to B^m\}$, with $h'(x) = h(x_1)|\cdots|h(x_m)$, where $x = (x_1|\cdots|x_m)$, is ϵ-AXU$_2$.* (The proof is a straightforward extension of Proposition 2 in [12].)

We consider a typical value for BH parameters and consider an example of our scheme with a similar value for ϵ. In particular we consider a BH that maps 1024-bit messages to 140-bit digests and has a collision probability of around 2^{-31}. The family of hash function is (2^{-31})-AU_2 with $(420 \times \log_2 140)$ bits of key. Since BH is not ϵ-AXU_2, it needs to be composed with an ϵ-AXU_2 to obtain the required security (cf. Proposition 7). In our scheme we should choose $n > 32$. In Section 4.3 we show that to find a suitable value for n, one needs only to verify that 2 is a primitive element modulo the prime n. That is, 2^i, for $i = 1, \ldots, n-1$, should generate all the elements of the set $\{1, 2, \ldots, n-1\}$. Table 2 shows that the first suitable value for n is 37. This is the least prime greater than 32 for which 2 is a primitive element. Therefore in our comparison we assume $n = 37$.

n	g	n	g	n	g	n	g	n	g
5	2	7	3	11	2	13	2	17	3
19	2	23	5	29	2	31	3	37	2
41	6	43	3	47	5	53	2	59	2
61	2	67	2	71	7	73	5	79	3
83	2	89	3	97	5	101	2	103	5
107	2	109	6	113	3	127	3	131	2
137	3	139	2	149	2	151	6	157	5
163	2	167	5	173	2	179	2	181	2
191	19	193	5	197	2	199	3	211	2
223	3	227	2	229	6	233	3	239	7
241	7	251	6	257	3				

Table 2. *List of the smallest primitive elements (g) of prime numbers (n) in the range* $[5 \ldots 257]$.

Message block length: In bucket hashing, for a given level of security, the size of the digest is lower bounded and is usually larger than what is used in MAC systems. Although other hash functions can be applied to reduce the size of the digest, the size of the original message block will remain large. For example for a 140 word digest a 1024 word message size must be used. For 32 bit machines this results in a large value for the minimum size of the message. If the message length is 20 words then the digest is 7 times longer than the message, and for 32 bit machines maps 640 bits to 4480 bits. So in BH, there is a large overhead in the computation when the message length is small [13]. Several layers of hashing in Rogaway's TOY-MAC makes no sense when the message length is small and the operations on the padding of the message (to increase its length to the acceptable minimum) is wasteful.

In our method for the same value of ϵ the message block length can be as small as 74 words (when $n = 37$). In this case $d_{min} = 2^{36}$ and so the key is

$k = (\log_2 d_{min} - 1) = 35$ bits. By choosing a larger suitable n we can increase the message block length and also the required level of security.

Key length: Similarly to message lengths, our system can be used for various key lengths which can be as small as 35 bits. BH uses 420 bits which is too long when used in a computational secure model where the one-time pad is replaced with a pseudorandom generator, which typically has a key length of around 128 bits. Rogaway [12] suggested the use of a pseudorandom number generator to obtain the required key for hashing.

Digest length: In BH, the digest length is 140 bits which is much longer than the required length (32-64 bits) in a MAC. Rogaway presented a TOY-MAC in which there exist four extra hashing levels to reduce the digest length to 64 bits. These extra levels result in an overall loss of efficiency of the system (particularly when the message is not very long). In our scheme, the digest length can be as low as 37 bits.

Memory: Our scheme does not require a large memory. With $n = 37$, it only needs 172 bytes for C_L. We note that for key distribution only the n-tuple determining the polynomial $L(x)$ can be used, but for hashing C_L must be calculated and saved. The message block requires 5 bytes, and 5 bytes are needed for the digest.

4.2 Implementation

Our scheme was implemented and tested on different data. The implementation can be divided into two parts. In the fist part, the key which is the n binary coefficients of the a linearized polynomial of degree less than 2^{n-1}, is extended to an $n \times n$ binary matrix. This matrix is stored in memory and does not need to be recalculated for each MAC calculation. In the second part, this matrix is used for hashing a given message (cf. the example given in Appendix 5). Once calculated, it is stored in the system for multiple use. The complexity of this part is equivalent to the complexity of binary matrix multiplication, which has been extensively studied in the literature [5].

4.3 Setup Phase

We can calculate the minimal linearized polynomials of elements in $GF(2^n)^*$ using the σ-orbits (conjugacy groups). Since each element of an orbit has the same minimal linearized polynomial. We only state the results here, the proofs will be given elsewhere.

The Galois automorphisms of $GF(2^n)$ are generated by: $\sigma : x \to x^2$. This group acts on $GF(2^n)^*$ and partitions this set into orbits. Recall that a *modulus* M of $GF(q^n)$ is a $GF(q)$-subspace which is invariant under σ, i.e. $M^\sigma = M$. A modulus is a union of σ-orbits: $M = \Omega_1 \cup \cdots \cup \Omega_r$. A q-polynomial is a linearized polynomial $L(z)$ whose coefficients lie in the ground field $GF(q)$.

Lemma 17. *Suppose $L(x)$ is a linearized polynomial over $GF(q^n)$. The zeroes of $L(x)$ form a modulus iff $L(x)$ is a q-polynomial.*

Thus each modulus M determines a q-polynomial, which is defined as follows

$$L(x) = \prod_{\beta \in M} (x - \beta).$$

For these facts consult [8].

Let $\Omega = \{\alpha^{2^i}\}$ be a σ-orbit. We define the following $GF(2)$-span of the vectors in this orbit

$$L(\Omega) =< \alpha, \alpha^2, \dots > .$$

Proposition 18. *Let $L(\Omega) =< \alpha, \alpha^2, \dots >$ where $\Omega = \{\alpha^{2^i}\}$ is an σ-orbit, then $L(\Omega)^\sigma = L(\Omega)$, i.e. $L(\Omega)$ is a modulus.*

$L(\Omega)$ not always irreducible. It can have a 1-dimensional σ-invariant subspace namely: $V =< \alpha + \alpha^2 + \cdots >$.

Proposition 19. *$L(\Omega) \simeq V \oplus W$, where V and W are σ-invariant $GF(2)$-subspaces.*

Let $\mathrm{Tr}(\beta)$, be the absolute trace of $\beta \in GF(2^n)$ over $GF(2)$, then $\mathrm{Tr}(\beta) = 0$, or 1. Moreover, $V =< \alpha + \alpha^2 + \cdots >$ is the zero subspace if $\mathrm{Tr}(\alpha) = 0$; it is a 1-dimensional σ-invariant subspace if $\mathrm{Tr}(\alpha) = 1$.

Suppose now that p is a prime such that 2 is a primitive mod p, i.e. $2^{p-1} \equiv 1 \bmod p$, and $p - 1$ is the least power of 2 for which this is true.

Theorem 20. *Let $L(\Omega) \simeq V \oplus W$, be decomposed into σ-invariant subspaces as in Proposition 19. If 2 is a primitive* mod p*, then W is an irreducible modulus.*

An analysis of the linearized polynomials corresponding to the irreducible moduli in Theorem 20 leads to the following result.

Theorem 21. *For any prime degree extension $GF(2^p)$ in which 2 is primitive mod p, the minimal 2-polynomial of any $\beta \in GF(2^p)^*$ is $x^{2^p} + x$ if $\mathrm{Tr}(\beta) = 1$. It is $x^{2^{(p-1)}} + x^{2^{(p-2)}} + \cdots + x^2 + x$, if $\mathrm{Tr}(\beta) = 0$.*

5 Conclusions

We have introduced two new evaluation hashing schemes that have a number of important and useful properties. In particular, the evaluation of digests reduces to matrix multiplication over $GF(2)$ which can be efficiently implemented. Developing an efficient algorithm for calculating d_{min} in general requires further research. However we have characterized a class of $GF(2^n)$ which require no computation. We have also compared a MAC based on $\mathcal{H}^1_{MRD}(t)$ with one based on BH.

A Proofs

Proof of Lemma 9: *(sketch)*

1. For all $L(x)$, x is a factor and hence $L(0) = 0$. For $x = (11, \cdots, 1)$ we have $x.\underline{\beta} \in GF(2)$ (page 52, [8]).
2. In a linearized polynomial the set of zeros form a subspace $\mathcal{N}_L$ of $GF(2^n)$ (Theorem 3.50, [8]).
   Now if $L(x) = y$ for some y, then for all $z \in \mathcal{N}_L$ we have $L(x + z) = L(x) + L(z) = y$. Conversely, if $L(x) = L(u) = y$ then $L(x + u) = 0 = L(z)$ where $z \in \mathcal{N}_L$. Hence if an element occurs once, it occurs exactly $|\mathcal{N}_L|$ times.
3. If $L(x)$ has even number of terms then $L(x) = x(x + 1)f(x)$ and we have $L(0) = L(1) = 0$ and hence $L(11, \cdots, 1) = 0$. This results in the following complementary property

$$L(\overline{x}) = L(x + 11 \cdots 1) = L(x) + L(11 \cdots 1) = L(x)$$

4. Let $u \in F_n$. There is a unique minimum linearized polynomial $L(x)$ such that $L(u) = 0$. An example computation of this polynomial is given in Appendix 5. Now if $L'(x)$ is a linearized polynomial such that $L'(u) = 0$, there exists a linearized polynomial $L''(x)$ such that $L''(x) = L'(x) \otimes L(x)$ (Theorem 3.68, page 113 [8]). Let $\mathcal{M}_u$ denote the collection of linearized polynomials for which u is a root and $|\mathcal{M}_u| = M_u$. Then we have $M_u = 2^{t-d_u}$, where d_u is the degree of $L(x)$. Now if L_1 is a linearized polynomial satisfying $L_1(u) = v$ and $L_2(x) \in \mathcal{M}_u$, then $(L_1 + L_2)(u) = L_1(u) + L_2(u) = v$ and hence v occurs 2^{t-d_u} in the column labelled by u.
5. For a pair $x, y \in GF(2^n)$, the number of rows, $L(x)$, of C, such that $L(x) = L(y)$ is the same as the number of rows of C with $L(x + y) = 0$ and hence the result.

□

This completes the first part.

Proof of Theorem 12. We only need to show that,

$$\frac{|\{h^{(1)} \in \mathcal{H}^1_{MRD}(t) : h^{(1)}(M) = 0\}|}{|\mathcal{H}^1_{MRD}(t)|} \leq \frac{1}{2^{t+1}}, \quad \forall M \neq 0 \in \Sigma^{2n}.$$

Let $M = (M_1|M_2)$. Since $h^{(1)}_L(M) = f_L(M_1) \oplus M_2$ and $|\mathcal{H}^1_{MRD}(t)| = 2^{t+1}$, we have to prove that,

$$|\{f_L : f_L(M_1) = M_2\}| \leq 1.$$

This is true because of Corollary 10. Note that since $M \neq 0$, M_1 and M_2 cannot be both zero. We consider two cases.

- If $M_1 = 0$, then $f_L(M_1) = 0 \neq M_2$ and so $|\{f_L : f_L(M_1) = M_2\}| = 0 \leq 1$.
- If $M_1 \neq 0$, then based on Corollary 10 $|\{f_L : f_L(M_1) = M_2\}| \leq 1$.

This proves the theorem. □

Proof of Theorem 13. Because of the linearity of f_L we have

$$h^{(2)}(s) = \Sigma_{j=0}^{2^n-2} s_j f_L(\alpha^j) = f_L(\Sigma_{j=0}^{2^n-2} \alpha^j) = f_L(\gamma_s)$$

for $\gamma = \Sigma_{j=0}^{2^n-2} s_j \alpha^j \in F_n$. To show that the class is ϵ-AXU$_2$ we note that for arbitrary $x, y \in F_{2^n-1}$ and $z \in F_n$, we have

$$\frac{|\{h^{(2)} \in \mathcal{H}^2_{MRD}(t) : h^{(2)}(x) \oplus h^{(2)}(y) = z\}|}{|\mathcal{H}^1_{MRD}(t)|} = \frac{|\{h^{(2)} \in \mathcal{H}^2_{MRD}(t) : h^{(2)}(x \oplus y) = z\}|}{|\mathcal{H}^1_{MRD}(t)|}$$

But $h^{(2)}_L(x \oplus y) = f_L(\gamma_x \oplus \gamma_y)$ and so

$$\frac{|\{h^{(2)} \in \mathcal{H}^2_{MRD}(t) : h^{(2)}(x \oplus y) = z\}|}{|\mathcal{H}^1_{MRD}(t)|} \leq \frac{1}{2^{t+1}} .$$

□

B Hashing Example for $\mathcal{H}^1_{MRD}$

Let $n = 5$, $F_5 = GF(2^5)$, $f(x) = x^5 + x^2 + 1$ be a primitive polynomial of degree n, and $f(\alpha) = 0$ for some $\alpha \in F$. We group the elements of F_5 into conjugate groups as shown in Table 3 and find the minimum linearized polynomial of each group. Since the smallest degree of the minimum linearized polynomials is 16, therefore we can use all linearized polynomial of degree less than 16 with odd number of terms, and construct a matrix which represents our hash function.

Conjugate groups	*Minimal polynomials*	*Linearized minimal polynomials*
$\alpha, \alpha^2, \alpha^4, \alpha^8, \alpha^{16}$	$x^5 + x^2 + 1$	$x + x^2 + x^4 + x^8 + x^{16}$
$\alpha^3, \alpha^6, \alpha^{12}, \alpha^{24}, \alpha^{17}$	$x^5 + x^4 + x^3 + x^2 + 1$	$x + x^{32}$
$\alpha^5, \alpha^{10}, \alpha^{20}, \alpha^9, \alpha^{18}$	$x^5 + x^4 + x^2 + x + 1$	$x + x^{32}$
$\alpha^7, \alpha^{14}, \alpha^{28}, \alpha^{25}, \alpha^{19}$	$x^5 + x^3 + x^2 + x + 1$	$x + x^2 + x^4 + x^8 + x^{16}$
$\alpha^{11}, \alpha^{22}, \alpha^{13}, \alpha^{26}, \alpha^{21}$	$x^5 + x^4 + x^3 + x + 1$	$x + x^{32}$
$\alpha^{15}, \alpha^{30}, \alpha^{29}, \alpha^{27}, \alpha^{23}$	$x^5 + x^3 + 1$	$x + x^2 + x^4 + x^8 + x^{16}$

Table 3. *Conjugate groups with the corresponding minimal and linearized minimal polynomials. (Cf. Theorem 21.)*

One can verify that $\{\alpha^3, \alpha^6, \alpha^{12}, \alpha^{24}, \alpha^{17}\}$ are linearly independent and thus a normal basis for $GF(2^5)$. Let $L(x) = x^4$. The vector corresponding to $L(x)$ is:

$$\begin{aligned} c_L &= (L(\alpha^3),\ L(\alpha^6),\ L(\alpha^{12}),\ L(\alpha^{24}),\ L(\alpha^{17})) \\ &= ((\alpha^3)^4,\ (\alpha^6)^4,\ (\alpha^{12})^4,\ (\alpha^{24})^4,\ (\alpha^{17})^4) \\ &= (\alpha^{12},\ \alpha^{24},\ \alpha^{17},\ \alpha^3,\ \alpha^6) \\ &= (\alpha^3 + \alpha^2 + \alpha,\ \alpha^4 + \alpha^3 + \alpha^2 + \alpha,\ \alpha^4 + \alpha + 1,\ \alpha^3,\ \alpha^3 + \alpha) \end{aligned}$$

The matrix representation of the function f_L is:

$$C_L = \begin{pmatrix} 0\,0\,1\,0\,0 \\ 1\,1\,1\,0\,1 \\ 1\,1\,0\,0\,0 \\ 1\,1\,0\,1\,1 \\ 0\,1\,1\,0\,0 \end{pmatrix}$$

and $h_L^{(1)} = \begin{pmatrix} C_L \\ I_5 \end{pmatrix}$ where I_5 is the identity matrix. Now let $M = [11010\,01100]$ which results in $M_1 = [11010]^t$ and $M_2 = [01100]$. We have, $f_L(M_1) = C_L \cdot M_1 = [00011]$ and so,

$$h_L^{(1)}(M) = f_L(M_1) \oplus M_2 = [00011] \oplus [01100] = [01111].$$

C Finite Fields

Consider F_n, the finite field with 2^n elements. We consider only the binary fields, although most of the results hold for general q-ary fields.

F_n is an n-dimensional vector space over $GF(2)$. It can be constructed using an irreducible polynomial, $f(x)$, of degree n. Let α denote a root of $f(x)$. Then $1, \alpha, \alpha^2 \cdots \alpha^{n-1}$ forms a basis of F_n. An element of F_n is a *primitive element* if its powers generate all non-zero elements of F_n.

Elements of F_n are partitioned into *conjugate groups* (the σ-orbits). The conjugate group of β consists of $\{\beta, \beta^2, \cdots\}$. If the elements of $\{\beta, \beta^2, \cdots \beta^{2^{n-1}}\}$ are linearly independent then they form a basis for F_n. A *normal basis* is a basis of the form $\{\beta, \beta^2 \cdots \beta^{2^{n-1}}\}$. Every field has at least one normal basis.

Let ψ denote an element of a finite field F_n. The *minimum polynomial* of ψ is an irreducible polynomial $f(x)$ over F_n such that $f(\psi) = 0$ and for any other polynomial $g(x)$ with $g(\psi) = 0$ has $f(x)$ as a factor. It can be shown that every element of the field has a unique minimum polynomial and all the conjugate elements have the same minimum polynomial.

A polynomial of the form $L(x) = \sum_i \alpha_i x^{2^i}$ with $\alpha_i \in GF(2)$ is called a 2-polynomial; we refer to these as *linearized polynomials*.

Linearized polynomials satisfy the following two properties:

$$L(\alpha + \beta) = L(\alpha) + L(\beta), \quad \alpha, \beta \in F;$$
$$L(c\alpha) = cL(\alpha), \quad \alpha, \in F, c \in GF(2).$$

The ordinary product of linearized polynomials is not a linearized polynomial. The *Symbolic product* of two polynomials $L_1(x)$ and $L_2(x)$, defined

$$L_1(x) \otimes L_2(x) = L_1(L_2(x))$$

gives a linearized polynomial.

Let $\psi \in F$ and $f(x)$ denote its minimum polynomial. The *minimum linearized polynomial* of ψ is a linearized polynomial $L(x)$ such that $f(x)$ is a factor of $L(x)$ and any other linearized polynomial for which $L_1(\psi) = 0$ can be written as $L_1(x) = L_2(x) \otimes L(x)$. It can be shown that $L(x)$ is unique.

D How to Calculate Minimal Linearized Polynomials

Suppose that $\alpha \in F_n$ and $f(\alpha) = 0$. To find the minimum linearized polynomial $p(x)$ for α we proceed as follows. Let $q(x)$ denote the minimum polynomial of α. We have

$$p(x) = \sum_{i=0}^{n} t_i x^{2^i} = m(x)q(x) \tag{1}$$

where $m(x)$ is a polynomial over $GF(2)$. For $i = 0, \ldots, n$, let $r_i(x)$ denote the remainder of dividing x^{2^i} by $q(x)$. To satisfy equation (1) we must have

$$\sum_{i=0}^{n} t_i r_i(x) \equiv 0 \mod q(x). \tag{2}$$

Expanding (2), we obtain a set of n equations in $n+1$ variables $t_0, \ldots, t_n$. The solution that results in a polynomial with minimum degree determines $p(x)$. Note that this method requires the determination of the irreducible polynomials corresponding to the σ-orbits. This adds to the computational complexity of this algorithm, while the irreducible polynomials are not required in the final result.

References

1. J. L. Carter and M. N. Wegman, "Universal Class of Hash Functions," *Journal of Computer and System Sciences*, vol. 18, no. 2, pp. 143–154, 1979.
2. K. Chen, "A New Identification Algorithm," in *Cryptography: Policy and Algorithms Conference* (E. Dawson and J. Golic, eds.), vol. 1029 of *Lecture Notes in Computer Science (LNCS)*, (Queensland, Australia), pp. 244–249, Springer-Verlag, July 1995.
3. E. Gabidulin, "Theory of Codes with Maximum Rank Distance," *Problems of Information Transmission*, vol. 21, no. 1, pp. 1–12, 1985.
4. T. Johansson, "Authentication Codes for Nontrusting Parties Obtained from Rank Metric Codes," *Design, Codes and Cryptography*, no. 6, pp. 205–218, 1995.
5. H. Krawczyk, "The Shrinking Generator; some practical consideration," in *Proceedings of Fast Software Encryption Workshop, FSE '93*, pp. 45–46, LNCS, Springer-Verlag, 1993.
6. H. Krawczyk, "LFSR-based Hashing and Authentication," in *Advances in Cryptology, Proceedings of CRYPTO '94* (Y. G. Desmedt, ed.), vol. 839 of *Lecture Notes in Computer Science (LNCS)*, pp. 129–139, Springer-Verlag, 1994.
7. H. Krawczyk, "New Hash Functions for Message Authentication," in *Advances in Cryptology, Proceedings of EUROCRYPT '95* (L. C. Guillou and J.-J. Quisquater, eds.), Lecture Notes in Computer Science (LNCS), (Berlin), pp. 301–310, 1995.
8. R. Lidl and H. Niederreiter, *Introduction to Finite Fields and their Applications.* Cambridge Uiversity Press, 1994.
9. U. Maurer, ed., *Advances in Cryptology, Proceedings of EUROCRYPT '96*, vol. 1070 of *Lecture Notes in Computer Science (LNCS)*, (Saragossa), Springer-Verlag, 1996.
10. B. Preneel, *Analysis and Design of Cryptographic Hash Functions.* PhD thesis, Katholieke University Leuven, Jan. 1993.

11. B. Preneel and P. C. van Oorschot, "On the Security of Two MAC Algorithms," in Maurer [9], pp. 19–32.
12. P. Rogaway, "Bucket Hashing and its Application to Fast Message Authentication," in *Advances in Cryptology, Proceedings of CRYPTO '95*, Lecture Notes in Computer Science (LNCS), pp. 30–42, Springer-Verlag, 1995.
13. V. Shoup, "On Fast and Provably Secure Message Authentication Based on Universal Hashing," in Maurer [9], pp. 321–331.
14. D. R. Stinson, "Universal Hashing and Authentication Codes," *Design, Codes and Cryptography*, vol. 4, pp. 369–380, 1994.
15. R. Taylor, "Near Optimal Unconditionally Secure Authentication," in *Advances in Cryptology, Proceedings of EUROCRYPT '94 (preprints)* (W. Wolfowicz and A. de Santis, eds.), vol. 765 of *Lecture Notes in Computer Science (LNCS)*, (Perugia, Italy), pp. 245–255, may 1994.
16. M. N. Wegman and J. L. Carter, "New Hash Functions and Their Use in Authentication and Set Equality," *Journal of Computer and System Sciences*, vol. 22, pp. 265–279, 1981.