Abstract—In this paper we consider authentication of multiple messages \( m_1, m_2, \ldots, m_L \), where each message \( m_i \) consists of \( s \) bits. We propose a scheme for the computation of a message authentication code (MAC) tag \( t \), of \( m_1, m_2, \ldots, m_L \) that takes constant time (time that is independent of \( L \)) and has a tag length that is constant or independent of \( L \). The verification time of the proposed scheme is also constant. Current schemes result in tag computation and verification times proportional to \( L \), and are hence less efficient than the proposed scheme. The proposed scheme uses a modification of division by an irreducible polynomial over \( GF(2) \) in order to compress the \( L \) messages. The compressed result is then input to a pseudorandom function \( F_t(.) \) to obtain a secure tag \( t \). We prove the security of the proposed MAC scheme. The proposed scheme has applications in sensor networks where many messages having one tag can reduce the number of bits being transmitted by a sensor node, thereby reducing the power consumption at a sensor node. Another application is in multimedia authentication, where a large multimedia data file can be split up into smaller segments whose MAC tag can be computed at high speed using the proposed scheme.

Keywords—Message Authentication Codes (MACs); Multiple Input Shift Register (MISR); provably secure MAC; pseudorandom function

I. INTRODUCTION

Message Authentication Codes (MACs) are widely used in communication networks in which the parties share a secret key and the channels are assumed to be insecure. In many such networks the communicated messages are lengthy which in turn necessitates the existence of fast MACs. Moreover, in many networks, such as sensor networks, messages are sent very frequently. This fact stresses the need for MACs that take multiple messages at the same time and generate a single tag for all the messages in efficient time.

In this paper we consider the problem of authentication of several messages simultaneously using a single, short tag. Note that all the messages are available simultaneously for tag computation. Such situations occur in sensor nodes when many messages generated by one sensor have to be transmitted to another. Instead of generating a tag for each message it would be more efficient to generate a short tag for all the messages at once. Assume that a message has \( s \) bits and the tag for each message has \( n \) bits. Then if a sensor wants to transmit \( L \) messages with \( L \) tags, it would have to transmit a total number of \( L(s+n) \) bits. Our scheme generates one tag of length \( n \) for all the \( L \) messages. Thus the sensor would have to transmit only \( sL+n \) bits, hence a reduction of \( n(L-J) \) bits. Another possible application of our method is when long messages have to be authenticated like in multimedia. A long multimedia data file can be split into \( L \) shorter messages of length \( s \) and a single tag \( t \) can be computed for the \( L \) messages. In this case the time needed to compute the tag and the verification time are reduced by a factor of \( L \).

The proposed method can be understood easily by first considering the generation of a tag for one message, \( m \), as follows. The binary sequence \( m \) is converted to a polynomial \( m(x) \) whose coefficients are equal to the binary sequence \( m \). For example 1011 is \( x^3 + x + 1 \) (each bit from the rightmost is multiplied by successive powers of \( x \) and the sum of all these powers of \( x \) is the polynomial). The tag can be computed by first computing \( r(x) = m(x) \mod g(x) \), where \( g(x) \) is an irreducible polynomial over \( GF(2) \) [1]. The coefficients of \( r(x) \) form a binary sequence \( r \) that is then input to a pseudorandom function \( F \) to obtain the tag as follows, \( t=F_0(r) \), where \( k \) is an \( n \)-bit key and \( r \) is an \( n \)-bit binary sequence, if \( g(x) \) is of degree \( n \). The operation \( m(x) \mod g(x) \) can be easily implemented by a Linear Feedback Shift Register (LFSR) with input \( m \) as shown in Fig. 1. The contents of the flip-flops 1 through 5 after all the bits of \( m \) have entered the LFSR are equal to \( r[2] \).

The message \( m \) can be split up into five messages of equal length, \( m_1, m_2, \ldots, m_5 \). These messages can be input into the LFSR of Fig. 1 to obtain the MISR (Multiple Input Shift Register) of Fig. 2. The contents of the flip-flops after the five messages have been input into the MISR are a sort of remainder, \( R \), when the five messages are divided by the characteristic polynomial of the LFSR, \((1+x^2+x^4+x^5)\). The tag of the five messages can be computed as \( F_4(R) \). Thus the tag computation of \( m \) has a speed-up of 5 when MISR is used instead of LFSR.

Figure 1. A Linear Feedback Shift Register with one input \( m \), five D-flip-flops (1 through 5) that performs division by \((1+x^2+x^4+x^5)\)
The proposed scheme will be shown to be secure in Section IV. The main advantage of our scheme is that the tag length, tag computation time, and tag verification time for as many messages as the degree of the irreducible polynomial $g(x)$ are the same as those for a single message.

The rest of the paper is organized as follows: In Section II, a number of the most important existing MAC schemes are briefly introduced and compared. Section III gives a formal definition of MAC and its security which we will use for representing our scheme. Section IV describes the proposed scheme in detail, gives a proof of its security, and explores its advantages and applications. Finally, Section V concludes the paper.

II. RELATED WORKS

Good MAC algorithms have been known since the mid-70s. At that time, MACs were most commonly constructed out of block ciphers like DES. The most popular in this genre is the CBC MAC, analyzed in [3], [4]. In 1979 Carter and Wegman published the idea of a Universal Hash Family [5], but did not mention MACs. In 1981, the same authors published another paper in which they proposed the use of Universal Hash Families for message authentication [6]. Their idea was quite novel: instead of applying some cryptographic primitive to the message $M$ to be MACed, they would first hash $M$ down to a smaller size using a hash function which had only a combinatorial property (rather than a cryptographic one). They then would apply a cryptographic primitive – one-time pad encryption – to the smaller resulting string. In 1983, Brassard [7] proposed using this approach within a complexity-theoretic cryptographic context: despite Wegman and Carter who had encrypted the output using a onetime pad, Brassard used a block cipher (e.g. DES) for producing a MAC.

In 1994, Krawczyk pursued Carter and Wegman’s approach to construct a MAC. In his paper ([8]), he described the “Division Hash”, a method similar to CRC computations. He also described a matrix-multiplication method called “LFSR-Based Toeplitz Hash.” Both of these methods focus on exploiting a Linear Feedback Shift Register (LFSR) to allow efficient hardware implementations.

In 1996 Shoup studied and compared a few variants of the Division Hash family [9]. Shoup’s families had shorter outputs and were therefore possibly more practical. NMAC and HMAC constructions which are based on “collision-resistant hash functions” [10] were introduced by Bellare et al. in 1996 [11]. Both NMAC and HMAC constructions can be used for variable-length messages.

In [12], [13] hash schemes were proposed that performed a division by a random irreducible polynomial. In 2000, Black gave a formal proof for the security of a MAC construction in which a pseudorandom function is applied to the output of a universal hash function [14].

Recently, there has been a surge of interest in aggregate MACs [15], [16]. The goal in this line of research is to reduce the number of tags routed in a network in which many nodes send messages to a single destination node and communication is an expensive resource. The proposed solution is to combine the tags of multiple messages together, such that the resulting tag is verifiable by the destination party. Katz and Lindell were the first to propose a formal proof for the security of aggregate MACs [15]. In these methods, a short tag may be produced but the tag generation and verification times are proportional to the number of messages. Our method on the other hand produces a short tag in constant time and has a constant verification time.

As we showed by an example in Section I the proposed scheme has a significant improvement in terms of tag length and generation/verification time when compared to the schemes which use LFSR. We also analyzed the amount of speedup when compared to HMAC (using SHA-1 with 160 bits output as the cryptographic hash function). We estimate that our proposed scheme is at least two orders of magnitude faster with respect to tag generation/verification time.

III. MESSAGE AUTHENTICATION CODES

Two users who want to communicate in an authenticated manner generate and share a secret key $k$ in advance of their communication. When one of the parties decides to send a message $m$ to the other one, she computes a tag, $t \leftarrow \text{MAC}_k(m)$ and transmits $(m,t)$. The tag is computed by a tag-generation algorithm, called Mac. Upon receiving $(m,t)$, the second party verifies if $t$ is a valid tag for message $m$. This is done using a verification algorithm, called Vrfy, which takes $k$, $t$, and $m$ as input and outputs 1 if $t$ is a valid tag for $m$.

A. Defining Message Authentication Codes

Formally, a message authentication code (MAC) is a tuple of probabilistic polynomial-time algorithms $(\text{Gen},\text{Mac},\text{Vrfy})$ in which Gen is the key generation algorithm, Mac is the tag generation algorithm, and Vrfy is the verification algorithm.

It is required that for every $n$ (a security parameter that usually specifies the length of key $k$), every $k$ output by Gen, and every $m$, it holds that $\text{Vrfy}(m, \text{MAC}_k(m)) = 1$. If $\Pi=(\text{Gen},\text{Mac},\text{Vrfy})$ is such that Mac is only defined for messages of a certain length, then the MAC scheme is said to be fixed-length, otherwise it is called variable-length MAC [10].

B. Security of Message Authentication Codes

There is a unique generally-accepted definition of security for message authentication codes. In simple words, no polynomial-time adversary should be able to generate a valid tag for a message which was not previously authenticated. Note that the adversary is allowed to request MAC tags for any messages of its choice. Toward the formal definition, consider the following experiment for a message authentication code $\Pi = (\text{Gen},\text{Mac},\text{Vrfy})$, adversary $A$, and security parameter $n$. 

978-1-4244-5637-6/10/$26.00 ©2010 IEEE
The Message Authentication Experiment Mac-forge\textsubscript{L}D(n):
1. Run Gen(n) to obtain a random key $k$.
2. The adversary $A$ is given oracle access to Mac($\cdot$). The adversary eventually outputs a pair $(m,t)$. Let $Q$ denote the set of all queries that $A$ asks its oracle.
3. The output of the experiment is defined to be 1 if and only if $(1)$ $\text{Vrfy}_k(m,t) = 1$ and $(2) \ m \not\in Q$ [10].

Security Definition:
A message authentication code $\Pi = (\text{Gen}, \text{Mac}, \text{Vrfy})$ is said to be existentially unforgeable under an adaptive chosen-message attack, or just secure, if for all probabilistic polynomial-time adversaries $A$, there exists a negligible function $\text{negl}(\cdot)$ such that $\Pr[\text{Mac-forge}_{L,D}(n) = 1] \leq \text{negl}(n)$.

A negligible function is defined as following: A function $f$ is negligible if for every polynomial $p(\cdot)$ there exists an $N$ such that for all integers $n > N$ it holds that $f(n) < \frac{1}{p(n)}$.

IV. PROPOSED SCHEME
Any binary string of length $d = sL$ representing a message $M$ can be split up into $L$ messages, $m_1, \ldots, m_L$ all of equal length $s$. These messages can be input into an MISR. The contents of the flip-flops after $s$ bits of the $L$ messages have been input into the MISR form a sort of remainder, $R$, when $L$ messages are divided by the characteristic polynomial of the MISR. The tag of the $L$ messages can be computed as $F_k(R)$, where $F_k$ is a pseudorandom function. A pseudorandom function is defined below.

Definition of Pseudorandom Function:
Let $F:\{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ be an efficient (polynomial-time), length-preserving, keyed function. We say that $F$ is a pseudorandom function if for all probabilistic polynomial-time distinguishers $D$, there exists a negligible function $\text{negl}(\cdot)$ such that:

$$\Pr\left[D^{F_k}(n) = 1\right] - \Pr\left[D^{F_{\perp}(n)} = 1\right] \leq \text{negl}(n),$$

where $k \leftarrow \{0,1\}^n$ is chosen uniformly at random and $f$ is chosen uniformly at random from the set of functions mapping $n$-bit strings to $n$-bit strings.

Remark: $F_k$ is chosen from one of $2^n$ distinct functions (one for each value of the $n$-bit key $k$). $f$ is chosen from the set of all $2^n$ functions with $n$-bit input and output, where $y = n2^n$. $D$ is given oracle access to some function (either $F_k$ or $f$, denoted $D^F$ or $D^D$) and its goal is to determine if this function is $F_k$ or $f$. $F_k(n)$ denotes the fact that $D$ is successful when given oracle access to $F_k$. If no probabilistic polynomial-time distinguisher $D$, can tell which function $D$ has oracle access to, then $F_k$ is a pseudorandom function. Note that $D$ can query the oracle function a polynomial number of times. Thus, even if $x_1$ and $x_2$ differ in only a single bit $F_k(x_1)$ and $F_k(x_2)$ look completely uncorrelated. Block ciphers such as AES can be used as pseudorandom functions [10].

A. Construction II
- **Gen**: On input $n$ (the security parameter), choose $k \leftarrow \{0,1\}^n$ uniformly at random, and a secret irreducible polynomial $g(x)$ over $GF(2)$ of degree $n$. Note that this step is done only once and therefore the key is $(k, g(x))$.
- **Mac**: On input $k \in \{0,1\}^n$, irreducible polynomial $g(x)$ over $GF(2)$ of degree $n$, and a message $M$ of length $d = sL$, $(M \in \{0,1\}^d)$ where $L \leq n$, split $M$ into $L$ parts each of length $s$, such that $M = (m_1, \ldots, m_L)$. Input all $m_i$’s into the MISR with characteristic polynomial $g(x)$. The contents of the MISR’s flip-flops after $s$ cycles is called $\text{MISR}(M) = R$ $( R \in \{0,1\}^s )$. Output the tag $t := F_k(R)$.
- **Vrfy**: On input $k \in \{0,1\}^n$, an irreducible polynomial $g(x)$ of degree $n$ over $GF(2)$, a message $M \in \{0,1\}^d$, and a tag $t \in \{0,1\}^s$, split $M$ to $L$ parts each of length $s$ $(m_1, \ldots, m_L)$, where $L \leq n$. Input all $m_i$’s into the MISR with characteristic polynomial $g(x)$. Output 1 if and only if $t = F_k(\text{MISR}(M))$.

Remark: The above MAC construction is for fixed length messages with $d$ bits. However $d$ can be changed initially because $s$ can be chosen to any value and fixed initially. The above definition can also be changed to a variable length MAC but we do not consider this extension in this paper.

**Theorem 1**: If $F_k(\cdot)$ is a pseudorandom function, then construction II is a MAC for messages of length $d$ that is existentially unforgeable under an adaptive chosen-message attack.

The next subsection is dedicated to the proof of Theorem 1.

B. Proof of Security
In order to prove security, the probability that adversary $A$ forges the scheme should be computed. We begin by examining the MISR values produced from $A$’s queries and her attempted forgery; consider the set $H = \{\text{MISR}(M_1), \text{MISR}(M_2), \ldots, \text{MISR}(M_{\ell})\}$. Where $\text{MISR}(M)$ denotes the contents of $n$ flip-flops of MISR with characteristic polynomial $g(x)$ and input $M = m_1||m_2||\ldots||m_L$ $(L \leq n, |M| = Ls)$ after $s$ cycles. The adversary, $A$, queries the MAC oracle with messages $Q = \{M_1, M_2, \ldots, M_{\ell}\}$ to get their tags $(t_1, t_2, \ldots, t_{\ell})$, and finally outputs $(M_{\ell+1}, t_{\ell+1})$ as a forgery.

The proof is carried out in two steps. In the first step we make use of the fact that the function MISR(\cdot) is a collision resistant function and in the next step we make use of the fact that it is negligibly probable to distinguish between a pseudorandom function and a random function.

We condition on the event that two or more of the elements in $H$ have the same value, i.e. $\text{MISR}(M_i) = \text{MISR}(M_j)$ where
\( i \neq j \). We call this event COLLISION and we represent it by \(\text{COLL} \) in the rest of the paper (\(\overline{\text{COLL}}\) denotes the complement of event COLL).

We don’t know necessarily how the adversary might use this to her advantage, but certainly some information is being given away if two of the elements in \(H\) have the same MISR output. So we will count any collision a bad event. Note that in the case when MISR(\(M_A\))=MISR(\(M_{A+1}\)) for \(1 \leq w \leq q\), (i.e. \(M_w\) is one of the queries and \(M_{w+1}\) is the final output of adversary) it is obvious that the scheme can be broken by probability 1 because the adversary can simply output \((M_{w+1}, t_w)\) as a forgery.

The probability of adversary \(A\) succeeding in experiment \(\text{Mac-forge}_{A,R}(n)\) can now be stated as:

\[
\text{Pr}[\text{Mac-forge}_{A,R}(n)=1]= \text{Pr}[\text{Mac-forge}_{A,R}(n)=1|\text{COLL}].\text{Pr}[\text{COLL}] + \text{Pr}[\text{Mac-forge}_{A,R}(n)=1|\text{COLL}].\overline{\text{Pr}[\text{COLL}]} \tag{1}
\]

In [17] it is proven that the probability of COLL happening is a negligible function of the degree of the MISR’s characteristic polynomial, \(n\), because the function MISR(.) is a collision-resistant function. Thus, we have:

\[
\text{Pr}[\text{COLL}] \leq \text{negl}(n) \tag{2}
\]

Clearly, \(\text{Pr}[\text{Mac-forge}_{A,R}(n)=1|\text{COLL}] \leq 1\), Thus, we will have:

\[
\text{Pr}[\text{Mac-forge}_{A,R}(n)=1|\text{COLL}] . \text{Pr}[\text{COLL}] \leq \text{Pr}[\text{COLL}] \tag{3}
\]

From (2) and (3):

\[
\text{Pr}[\text{Mac-forge}_{A,R}(n)=1|\text{COLL}].\text{Pr}[\text{COLL}] \leq \text{negl}(n) \tag{4}
\]

From (1) and (4):

\[
\text{Pr}[\text{Mac-forge}_{A,R}(n)=1] \leq \text{negl}(n) + \text{Pr}[\text{Mac-forge}_{A,R}(n)=1|\overline{\text{COLL}}].\text{Pr}[\overline{\text{COLL}}] \tag{5}
\]

Since \(\text{Pr}[\overline{\text{COLL}}] \leq 1\), we have:

\[
\text{Pr}[\text{Mac-forge}_{A,R}(n)=1|\overline{\text{COLL}}].\text{Pr}[\overline{\text{COLL}}] \leq \text{Pr}[\text{Mac-forge}_{A,R}(n)=1|\overline{\text{COLL}}] \tag{6}
\]

Finally, from (1) and (6):

\[
\text{Pr}[\text{Mac-forge}_{A,R}(n)=1] \leq \text{negl}(n) + \text{Pr}[\text{Mac-forge}_{A,R}(n)=1|\overline{\text{COLL}}] \tag{7}
\]

Next we show that \(\text{Pr}[\text{Mac-forge}_{A,R}(n)=1|\overline{\text{COLL}}] \leq \text{negl}(n)\) by using the fact that it is not possible for an adversary to distinguish between a pseudorandom function and a random function in polynomial-time. This then implies that \(\text{Pr}[\text{Mac-forge}_{A,R}(n)=1] \leq \text{negl}(n)\) and our MAC construction is secure.

This is accomplished by first analyzing the security of the proposed scheme using a truly random function, and then we will consider the result of replacing the truly random function with a pseudorandom one. Let \(A\) be a probabilistic polynomial-time adversary and define:

\[
e(n) = \text{Pr}[\text{Mac-forge}_{A,R}(n)=1|\overline{\text{COLL}}] \tag{8}
\]

Consider a message authentication code \(\Pi=(\text{Gen}',\text{Mac}',\text{Vrfy}')\) which is the same as \(\Pi=(\text{Gen},\text{Mac},\text{Vrfy})\) in our proposed construction except that a truly random function \(f\) is used instead of the pseudorandom function \(F_k\). That is, \(\text{Gen}'(n)\) works by choosing a random function \(f ← \text{func}_n\) (\(\text{func}_n\) is the set of all functions with \(n\)-bit input and output), and \(\text{Mac}'\) computes a tag just as \(\text{Mac}\) does except that \(f\) is used instead of \(F_k\). It is straightforward to see that

\[
\text{Pr}[\text{Mac-forge}_{A,R}(n)=1|\overline{\text{COLL}}] \leq \frac{1}{2^n} \tag{9}
\]

This is because for the MISR output \(R\) corresponding to any message \(M\notin Q\), (Recall that \(Q\) is the set of queries) the value of \(t = f(R)\) is uniformly distributed in \(\{0,1\}^n\) from the point of view of the adversary \(A\).

We now construct a polynomial-time distinguisher \(D\) that is given oracle access to the MAC that could either use a pseudorandom function \(F_k(\cdot)\) or a random function \((\cdot,\cdot))\), and whose goal is to determine whether this function is pseudorandom (i.e. equal to \(F_k(R)\) for randomly-chosen \(k ← \{0,1\}^n\) ) or random (i.e. equal to \(f(R)\) for \(f ← \text{func}_n\) ). To do this, \(D\) emulates the message authentication experiment for \(A\) in the manner described below, and observes whether \(A\) succeeds in outputting a valid tag on a “new” message. If so, \(D\) guesses that its oracle must be a pseudorandom function; otherwise, \(D\) guesses that its oracle must be a random function.

**Distinguisher D:**

\(D\) is given input \(n\) and access to an oracle \(O\): \(\{0,1\}^* \rightarrow \{0,1\}^n\) and works as follows:

1. Run \(A(n)\). Whenever \(A\) queries its MAC oracle on a message \(M\), answer this query in the following way:
   - Query \(O\) with \(M\) and obtain response \(t\), return \(t\) to \(A\).

2. When \(A\) outputs forgery \((M,t)\) at the end of the execution, do:
   - Query \(O\) with \(M\) and obtain response \(\tilde{t}\).
   - If (1) \(\tilde{t} = t\) and (2) \(A\) never queried its MAC oracle on \(M\), then output 1; otherwise, output 0.

It is clear that \(D\) runs in polynomial time. Notice that if \(D\)’s MAC oracle uses pseudorandom function \(F_k(R)\), then the view of \(A\) when run as a sub-routine by \(D\) is distributed identically to the view of \(A\) in experiment \(\text{Mac-forge}_{A,R}(n)\). Furthermore, \(D\) outputs 1 exactly when \(\text{Mac-forge}_{A,R}(n)\). Therefore, considering (8) we have:
Pr[\(D^{f(\cdot)}(n)=1\)] = Pr[Mac-Forge\(_{A,II}(n) = 1\)] - Pr[COLL(\(m\)) = \epsilon(n) \text{ (10)},
\]
where \(k \leftarrow \{0,1\}^n\) is chosen uniformly at random. If \(D\)'s MAC oracle uses random function \(f(R)\), then the view of \(A\) when run as a sub-routine by \(D\) is distributed identically to the view of \(A\) in experiment Mac-Forge\(_{A,II}(n)\), and again, \(D\) outputs 1 exactly when Mac-Forge\(_{A,II}(n)\) = 1. Therefore considering (9) we have:

\[
Pr[\(D^{f(\cdot)}(n)=1\)] = Pr[Mac-Forge\(_{A,II}(n) = 1\)] \leq \frac{1}{2^n} \text{ (11)},
\]
where \(f \leftarrow \text{func}\_9\) is chosen uniformly at random.

Thus, from (10) and (11):

\[
Pr[\(D^{f(\cdot)}(n)=1\)] - Pr[\(D^{f(\cdot)}(n)=1\)] \geq \epsilon(n) - \frac{1}{2^n}
\]

Since \(F_k\) is a pseudorandom function, there exists a negligible function \(\text{negl}(n)\) such that \(\epsilon(n) - \frac{1}{2^n} \leq \text{negl}(n)\). We then have \(\epsilon(n) \leq \text{negl}(n) + \frac{1}{2^n}\) and therefore \(\epsilon\) is also negligible. Thus, from definition (8), and equation (7) \(Pr[\text{Mac-Forge}_{A,II}(n) = 1] \leq \text{negl}(n)\) and our MAC construction is secure. This concludes the proof that our proposed construction is secure under adaptive chosen-message attack.

V. CONCLUSION

The message authentication code proposed in this paper can be easily extended to the case of variable length messages. Our method is useful when long messages have to be authenticated or many messages have to be authenticated at once. Such requirements occur in multimedia communication or wireless sensor networks. Since our method uses MISRs, the hardware implementation of the proposed MAC is very simple and efficient.

The proof of security can also be based on the fact that a pseudorandom function is a secure fixed length MAC. In this case, it can be shown that the event \(\text{COLL}\) results in a forgery for the MAC in which the tag of \(m\) is computed as \(F_k(m)\). The proposed MAC is therefore secure and efficient with respect to tag computation time, verification time and tag length.

REFERENCES