SANDIA REPORT

SAND2004-5113
Unlimited Release
Printed October 2004

Manticore and CS Mode: Parallelizable Encryption with Joint Cipher-State Authentication

Erik Anderson, Cheryl Beaver, Timothy Draelos,
Richard Schroeppel, and Mark Torgerson
Cryptography and Information Systems Surety Department

Russell Miller
Digital Microelectronics Department

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94-AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.doe.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/ordering.htm

SAND2004-5113
Unlimited Release
Printed October 2004

Manticore and CS Mode: Parallelizable Encryption with Joint Cipher-State Authentication

Erik Anderson, Cheryl Beaver, Timothy Draelos,
Richard Schroeppel, and Mark Torgerson
Cryptography and Information Systems Surety Department

Russell Miller
Digital Microelectronics Department

Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-0785

Abstract

We describe a new mode of encryption with inexpensive authentication, which uses information from the internal state of the cipher to provide the authentication. Our algorithms have a number of benefits: 1) the encryption has properties similar to CBC mode, yet the encipherment and authentication can be parallelized and/or pipelined, 2) the authentication overhead is minimal, and 3) the authentication process remains resistant against some IV reuse. We offer a Manticore class of authenticated encryption algorithms based on cryptographic hash functions, which support variable block sizes up to twice the hash output length and variable key lengths. A proof of security is presented for the MTC4 and Pepper algorithms. We then generalize the construction to create the Cipher-State (CS) mode of encryption that uses the internal state of any round-based block cipher as an authenticator. We provide hardware and software performance estimates for all of our constructions and give a concrete example of the CS mode of encryption that uses AES as the encryption primitive and adds a small speed overhead (10-15%) compared to AES alone.

Keywords: Authenticated Encryption, Encryption Mode, Cipher-State Mode, Inexpensive Authentication, Luby-Rackoff, Feistel, Middletext, Hash, Cipher, Manticore, MTC4, Pepper

Contents

Appendix

Figures

Tables

Manticore and CS Mode: Parallelizable Encryption with Joint Cipher-State Authentication

1 Introduction

When choosing a cipher, its mode of operation, and method of authentication, one needs to consider the security, speed, size, and functionality required by the application. Data security schemes have typically relied on combining an encryption step (with a mode of cipher operation) and a message authentication mechanism. These separate processes lead to undesirable computational costs.

One would like to speed up the process by using information from the encryption step for authentication. Recent research has considered authenticated encryption schemes that are more efficient than current standards and practices of separate data encryption and data authentication [11, 12, 24, 26]. In particular, OCB [24] is parallelizable and offers CBC-like authenticated encryption with only two extra block cipher invocations over that needed for encipherment alone. CCM [26], as specified in NIST Draft Pub 800-38C, offers authenticated encryption with associated data (AEAD) for AES, which accommodates a combination of secret and non-secret data by authenticating all data and encrypting only secret data.

We strive to take a new and different approach - to examine using a cipher's internal state as inputs for an authentication mechanism. We use the term 'Manticore' to reflect the combination of cryptographic functions resulting from this encryption method with joint cipher-state authentication. Our approach is also parallelizable, yet exhibits many of the practical benefits of CBC mode. The authentication adds minimal cost to the encryption process. The presented methods also offer security in the face of initialization vector (IV) reuse, which, to our knowledge, existing authenticated encryption mechanisms do not. Finally, given the landscape of cryptographic algorithms, we have chosen to not pursue patents on these new algorithms so as not to contribute to the current patent minefield. This report covers a superset of our work that was published in [1].

We examine block ciphers comprised of $2r$ rounds. The authentication tag is a function of the encryption state after r rounds. Our first construction, MTC4, is based on a four round Feistel network with cryptographic hash functions as round functions. Many of the components necessary for security can be added into the round functions because of the hash algorithm's ability to accept arbitrary length inputs. In Section 2.1, the MTC4 algorithm is shown to be secure with respect to both privacy and integrity, under general security assumptions.

Extensive research has been conducted on the security and construction of low round Feistel ciphers. In [16], Luby and Rackoff show how to construct $2n$-bit pseudo-random permutations using a Feistel network. Their constructions are secure against any adversary who has combined adaptive chosen plaintext and ciphertext attacks. In [13], Knudsen provides a nice survey and analysis of the security bounds for low round Feistel constructions. Much research has also been done finding practical instantiations of low round Feistel networks using cryptographic hashes as round functions. In [2, 17], the authors examine three round ciphers, while Lim [15] looks at four round constructions. Naor and Reingold [19] and Patel, Ramzan, and Sundaram [21] examine replacing some of the hash functions used in the various rounds with less expensive function calls.

Section 3 presents a reduced computation version of MTC4, RMTC4, which uses a more efficient function for rounds 1 and 4. In Section 4, we present a 2-round version of the Manticore class of algorithms called Pepper and provide a proof of security. Pepper is faster than MTC4, but it is not resistant to IV reuse.

For our last construction, discussed in Section 5, we show how to take an arbitrary round-based cipher and extend it to provide inexpensive authentication. As with the hash-based construction, the general version exhibits encryption properties similar to CBC mode, is parallelizable, and the authentication adds minimal overhead and provides security against some IV reuse. We use AES as a concrete example and have submitted the CS (Cipher State) mode to NIST as a proposed mode of operation for AES [20]. Sections 6 and 7 respectively present software and hardware implementation results of our authenticated encryption constructions.

2 Manticore4 (MTC4)

As its namesake implies, our Manticore constructions comprise a number of common elements. The basic cipher elements use cryptographic hash functions in a n-round Feistel network and can be viewed as a variant of [15]. One attractive feature is the ability of the hash to accept arbitrary sized inputs. This allows us to insert an IV, round counter, and block counters into the round inputs in a simple fashion, which provides properties similar to CBC mode, yet each block can be computed in parallel. This construction can be used to create a block cipher of any bit length up to twice the hash size. Of course, the hash may be truncated to produce shorter block sizes. The key size is adjustable and impacts performance only when the input block size of the hash function is exceeded.

The MTC4 algorithm is a 4-round ManTiCore construction. Let H be a cryptographically strong hash function mapping an arbitrary number of bits to n bits. Let K be a k-bit key and IV be a v-bit initialization vector. Let $M = m_1, m_2, \cdots, m_{2j}$ be the message to be encrypted, where each m_i is h bits in length. We assume the

message M is padded with some suitable padding scheme, if necessary, so it is a multiple of $2n$ bits in length.

The following is the MTC4 algorithm, which is depicted in Figure 1.

MTC4
INPUT $(IV, M), K$
OUTPUT $(IV, C, AUTH)$
Set $CS \leftarrow 0$
For i from 1 to $2j - 1$ by 2 do
    Set $x \leftarrow m_i \oplus H(K, IV, 0, i, m_{i+1})$
    Set $y \leftarrow m_{i+1} \oplus H(K, IV, 1, i, x)$
    Set $CS \leftarrow CS \oplus x \oplus y$
    Set $c_{i+1} \leftarrow x \oplus H(K, IV, 2, i, y)$
    Set $c_i \leftarrow y \oplus H(K, IV, 3, i, c_{i+1})$
Set $AUTH = H(K, IV, 0, 0, CS)$
RETURN $(IV, C, AUTH)$

To enable a security proof, the fields presented to the hash must be aligned to ensure independence. Hence the sizes of the round and block counters must be consistent. At a minimum, we need at least 2 bits to represent the round number and $log_2(j)$ bits for the block number. Let u represent a round counter of fixed length and z a block counter of fixed length. Although cryptographic hash functions accept arbitrary-length inputs, typically they process a block of b bits at a time. For instance, SHA-1 [8] operates on 512-bit blocks and outputs 160 bits. From an efficiency standpoint, one should limit the parameter size so the arguments fit in one input block, that is $k + v + u + z + n \leq h$, where h is the input block size of the hash function H. Given that they do, the expected speed of MTC4-SHA-1 is on the order of the $160/512 * 1/2 = 5/32$ times as fast as SHA-1. The cost to authenticate the entire message is essentially that of having to hash only a single block of data. In addition, both the encryption and authentication for each message block can be computed in parallel, leaving a single hash of the combined pre-authenticators, CS, to complete the process.

2.1 Security Considerations

When viewed strictly as a block cipher, MTC4's security relies on the pseudorandomness of a four-round Feistel network and the properties of a cryptographic hash. For a wide range of key and block sizes, the non-inversion properties of a cryptographic hash satisfy the definition for an ideal function as described in [13]. Thus, in MTC4 and like ciphers, the work to mount a key recovery attack, even given inputs and outputs of the round functions, is the minimum of 1) exhaustion of the key space or 2) inversion of the hash. So if $k \leq n$, the best method to recover the key is exhaustion. In practice, the inputs and outputs of the round function are not given to an

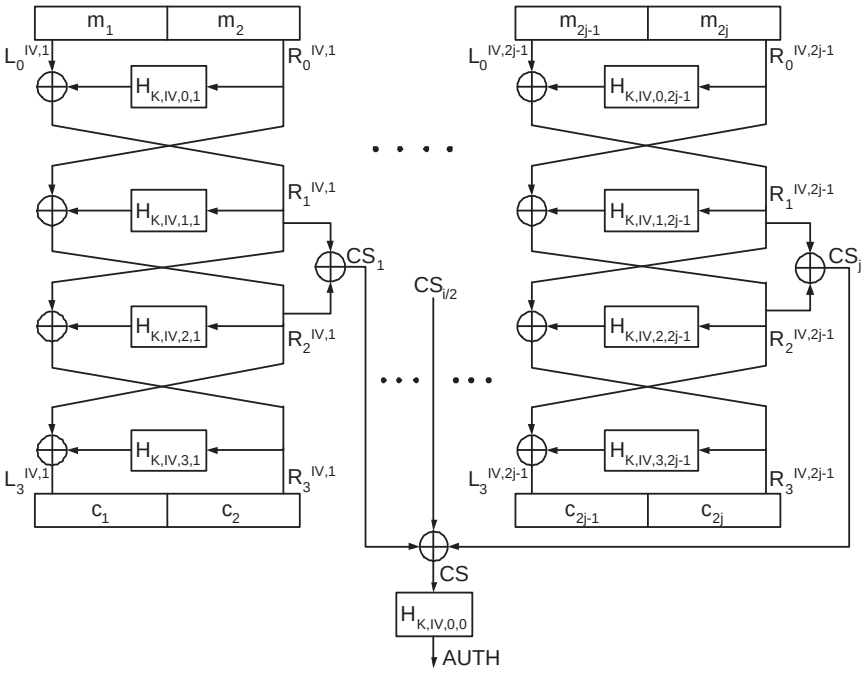

Figure 1. Block diagram for the MTC4 algorithm.

adversary, so a non-exhaustive key recovery method is harder than inverting the hash directly.

The strength of the authentication mechanism is tied to the collision resistance property of the cryptographic hash. If one were to hold the IV fixed and publish the secret key, and one were to assume that it takes work less than $2^{\frac{h}{2}}$ to construct a collision in the hash, then one would be able to push a known differential down to the end of the second round with that same amount of work. The combining function for the pre-authenticator, CS, is a simple XOR so one may combine several differentials to create a collision in the pre-authenticator CS. There are a couple of things to note. The collision resistance property of a cryptographic hash removes the possibility of pushing differentials down to the second round. The fact that the key is indeed secret means that one may use a function that is not perfectly collision resistant. What is needed to foil the attack above is a function that, including a k-bit key, requires at least $2^{\frac{h}{2}}$ work to find a collision in the output. In the remainder of this section, we take a more formal look at the security of MTC4.

There are two notions of security to consider. The first is message privacy, which looks at the security of the encryption and decryption processes. The second is ciphertext integrity, which measures the ability to force an authentication. For notational purposes, we first provide a quick review of message privacy.

A symmetric scheme $\mathcal{SE} = (\mathcal{K}, E, D)$ consists of three algorithms, a key gener-

ation algorithm $\mathcal{K}(n)$ that takes a security parameter n and returns a random key K, and encryption and decryption algorithms E_K, D_K respectively. The encryption algorithm takes a message m and key K and returns a ciphertext $C \xleftarrow{R} E_K(m)$, either randomly or based on some predetermined state. The decryption algorithm D_K takes a ciphertext C and returns deterministically either a message m such that $(D_K \circ E_K)(m) = m$ or the symbol $\perp$, reflecting that C was not a valid ciphertext.

Definition (Indistinguishability of a Symmetric Encryption Scheme [6]) Let $b \xleftarrow{R} \{0,1\}$ and (m_0, m_1) denote two equal length plaintexts. Following [4], define the left-or-right encryption oracle $E_K(\mathcal{LR}(\cdot,\cdot,b))$, as the oracle taking queries of the form (m_0, m_1) and returning either $E_K(m_0)$ if $b = 0$ or $E_K(m_1)$ if $b = 1$. A symmetric encryption scheme is said to be secure against chosen-plaintext attacks if for any adversary A_{cpa} with oracle access to $E_K(\mathcal{LR}(\cdot,\cdot,b))$, denoted $A_{\text{cpa}}^{E_K(\mathcal{LR}(\cdot,\cdot,b))}$, the advantage in determining the correct value of b is negligible.

More formally, let $\mathcal{SE} = (\mathcal{K}, E, D)$ be a symmetric encryption scheme and consider the following experiment.

$\mathbf{Exp}^{\text{ind-cpa-b}}_{\mathcal{SE},A_{\text{cpa}}}$

1. $K \xleftarrow{R} \mathcal{K}$
2. $d \leftarrow A_{\text{cpa}}^{E_K(\mathcal{LR}(\cdot,\cdot,b))}$
3. Return d

The advantage obtained by the adversary in distinguishing the correct value of b is defined as

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE},A_{\text{cpa}}} := \Big|\Pr[\mathbf{Exp}^{\text{ind-cpa-1}}_{\mathcal{SE},A_{\text{cpa}}} = 1] - \Pr[\mathbf{Exp}^{\text{ind-cpa-0}}_{\mathcal{SE},A_{\text{cpa}}} = 1]\Big|.$$

The advantage of the encryption scheme $\mathcal{SE}$ over all adversaries A_{cpa} making q queries totaling $\leq \mu$ bits is defined as

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE}}(q,\mu) := \max_{A_{\text{cpa}}}\{\mathbf{Adv}^{\text{ind-cpa}}_{\mathcal{SE},A_{\text{cpa}}}\}.$$

Proposition 1. *Let MTC4 denote the 4-round Manticore encryption scheme with round function f, where f is a random function taking $n+v+u+z$-bits to n-bits. Then for any q queries totaling $\leq \mu$ bits, the distinguishing advantage $\mathbf{Adv}^{\text{ind-cpa}}_{MTC4}(q,\mu) = 0$ in the IND-CPA model.*

Proof: Let A_{cpa} be any distinguishing algorithm and Γ the set of all transcripts σ such that $A_{\text{cpa}}(\sigma) = 1$. Let T_{MTC4} denote the transcript generated by A_{cpa} given

oracle access to $MTC4$. The advantage may be rewritten as

$$\begin{aligned}\mathbf{Adv}^{\text{ind-cpa}}_{MTC4,A_{\text{cpa}}}(q,\mu) &= \Big|\Pr[\mathbf{Exp}^{\text{ind-cpa-1}}_{MTC4,A_{\text{cpa}}}=1]-\Pr[\mathbf{Exp}^{\text{ind-cpa-0}}_{MTC4,A_{\text{cpa}}}=1]\Big| \\ &= \Big|\sum_{\sigma\in\Gamma}\big(\Pr_{MTC4}[T_{MTC4}=\sigma \mid b=1]- \\ &\qquad\qquad \Pr_{MTC4}[T_{MTC4}=\sigma \mid b=0]\big)\Big|.\end{aligned}$$

Notice that for each round in the encryption block, the IV/counter value pairs are unique. This implies the output of the each round, including the final authentication tag $AUTH$, are random. Therefore for every transcript σ,

$$\begin{aligned}\Pr_{MTC4}[T_{MTC4}=\sigma \mid b=1] &= \Pr_{MTC4}[T_{MTC4}=\sigma \mid b=0] \\ &= \frac{1}{2^{n\mu}}\cdot\frac{1}{2^{nq}}.\end{aligned}$$

where q and μ are the number of different IV's and bits queried. Therefore it follows,

$$\mathbf{Adv}^{\text{ind-cpa}}_{MTC4}(q,\mu)=0. \qquad \square$$

Since MTC4's message security and authentication are integrated, one must be cautious that neither leaks enough information to allow an adversary to mount an attack. To ensure that tapping the internal state of the cipher does not compromise security, we need to show that the ciphertext integrity is protected.

Definition (Integrity Awareness [6]) Let $\mathcal{SE}=(\mathcal{K},E,D)$ be a symmetric encryption scheme and A_{ctxt} an adversary with access to two oracles, E_K and V_K. The oracle V_K takes a ciphertext C and returns 1 if there is a plaintext m satisfying $E_K(m)=C$ and 0 otherwise. If after $C_1,\ldots,C_q$ oracle replies to E_K, A_{ctxt} can produce a ciphertext C different from C_i, $i=1,\ldots,q$, satisfying $V_K(C)=1$, then we say A_{ctxt} was successful. The adversary's success is defined as

$$\mathbf{Adv}^{\text{int-ctxt}}_{\mathcal{SE},A_{\text{ctxt}}} := \Pr[V_K(C)=1].$$

The ciphertext integrity for the symmetric encryption scheme $\mathcal{SE}$ over all adversaries A_{ctxt} is defined as

$$\mathbf{Adv}^{\text{int-ctxt}}_{\mathcal{SE}}(q,\mu) := \max_{A_{\text{ctxt}}}\{\mathbf{Adv}^{\text{int-ctxt}}_{\mathcal{SE},A_{\text{ctxt}}}\}.$$

Proposition 2. *Suppose $MTC4$ has round function f, where f is a random function taking $n+v+u+z$ bits to n bits. Then for any q queries totaling $\leq \mu$ bits, the advantage in forging an authentication is*

$$\mathbf{Adv}^{\text{ind-ctxt}}_{MTC4}(q,\mu)=\frac{1}{2^n}+\frac{1}{2^n}\Big(1-\frac{1}{2^n}\Big).$$

Proof: Let A_{ctxt} be any algorithm as defined in the Integrity Awareness model and let σ denote the transcript generated by A_{ctxt}'s plaintext query/ciphertext replies. We will use the word ciphertext loosely to define the ciphertext/authentication tag pair. We observe that the final integrity check query made by A_{ctxt} must satisfy one of the following four cases:

Case 1: A_{ctxt} queries a ciphertext using an IV that it has not seen before.

Case 2: A_{ctxt} queries a ciphertext/IV pair using a previously seen IV, but the ciphertext is longer than the one contained in σ.

Case 3: A_{ctxt} queries a ciphertext/IV pair that is either shorter or the same length as that contained in σ, but at least one of the ciphertext blocks is changed by A_{ctxt}.

Case 4: A_{ctxt} queries a truncated version of a ciphertext/IV pair contained in σ.

Recall the advantage A_{ctxt} achieves in forging an authentication is defined as

$$\mathbf{Adv}^{\text{int}-\text{ctxt}}_{MTC4,A_{\text{ctxt}}} := \Pr[V_K(C) = 1].$$

If we let Ψ denote the set of all ciphertext such that $V_K(C) = 1$, the above equation can be rewritten as

$$\mathbf{Adv}^{\text{int}-\text{ctxt}}_{MTC4,A_{\text{ctxt}}} = \sum_{\sigma} \Pr[\, \Psi \ni C \leftarrow A_{\text{ctxt}} \mid T_{MTC4} = \sigma] \cdot \Pr[T_{MTC4} = \sigma].$$

If we can show for arbitrary transcripts σ that A_{ctxt}'s advantage is bounded above by some negligible factor for each of the four cases, then our claim will follow. We now analyze each case separately.

Case 1 is straightforward since the authenticator has never seen an input with this type of IV. Hence $\Pr[\, \Psi \ni C \leftarrow A_{\text{ctxt}} \mid T_{MTC4} = \sigma] = 1/2^n$. In Case 2, the output of both $R_1^{IV,i_{\text{new}}}$ and $R_2^{IV,i_{\text{new}}}$ for each new counter value i_{new} is random, since they are both independent of the transcript σ. Algorithm A_{ctxt} may choose to keep the same authentication tag $Auth_{IV}$ in its final query or replace it with another. It is not difficult to show the best choice for A_{ctxt} is not to change $Auth_{IV}$. Therefore the probability of success is no larger than

$$\begin{aligned} &\Pr[\, \Psi \ni C \leftarrow A_{\text{ctxt}} \mid T_{MTC4} = \sigma] \qquad (1)\\ &\leq \Pr\big[CS^{IV}_{new} = CS^{IV}_{old} \text{ or } CS^{IV}_{new} \neq CS^{IV}_{old} \text{ and} \\ &\qquad\qquad f\big(IV.0.0.CS^{IV}_{new}\big) = Auth_{IV} \mid T_{MTC4} = \sigma\big] \\ &= \frac{1}{2^n} + \frac{1}{2^n}\Big(1 - \frac{1}{2^n}\Big). \end{aligned}$$

For Case 3, there are two different types of attacks that A_{ctxt} may choose from. Without loss of generality, we will assume that only one ciphertext block is changed and all the rest remain untouched. Changing several ciphertext blocks does not give A_{ctxt} an advantage. Given this assumption, the two cases are:

(3.a) For some i, the left hand output $L_3^{IV,i}$ is changed, but the right hand $R_3^{IV,i}$ remains the same.

(3.b) For some i, the right hand output $R_3^{IV,i}$ is changed.

For case 3.a, observe the value $R_2^{IV,i}$ must be different from the original, since $L_3^{IV,i}$ has been modified. The output $f(IV.2.i.R_2^{IV,i})$ must be random and hence following Equation 2, A_{ctxt}'s success can be no larger than $1/2^n + 1/2^n(1 - 1/2^n)$. Case 3.b follows the same reasoning, except now the probability that $CS_{new}^{IV} = CS_{old}^{IV}$ is no longer $1/2^n$. In particular, we have

$$\begin{aligned}
&\Pr[CS_{new}^{IV} = CS_{old}^{IV} \mid T_{MTC4} = \sigma] \\
&= \Pr[R_{2,new}^{IV,i} = R_{2,old}^{IV,i} \text{ and } R_{1,new}^{IV,i} = R_{1,old}^{IV,i} \text{ or} \\
&\qquad R_{2,new}^{IV,i} \neq R_{2,old}^{IV,i} \text{ and } R_{2,new}^{IV,i} \oplus R_{1,new}^{IV,i} = R_{2,old}^{IV,i} \oplus R_{1,old}^{IV,i}] \\
&= 0 + \frac{1}{2^n}\Big(1 - \frac{1}{2^n}\Big).
\end{aligned}$$

Therefore,

$$\Pr[\,\Psi \ni C \leftarrow A_{\text{ctxt}} \mid T_{MTC4} = \sigma] = \frac{1}{2^n}\Big(1 - \frac{1}{2^n}\Big) + \frac{1}{2^n}\Big(1 - \frac{1}{2^n}\Big(1 - \frac{1}{2^n}\Big)\Big).$$

For the fourth and final case, suppose A_{ctxt} returns a query to a truncated version of an IV with m_{IV} blocks. Let $m'_{IV} < m_{IV}$ denote the number of blocks in the truncated ciphertext. Since,

$$CS_{new}^{IV} = CS_{old}^{IV} \quad \Leftrightarrow \quad \sum_{i=m'_{IV}+1}^{m_{IV}} \big(f(IV.0.i.R_0^{IV,i}) \oplus f(IV.1.i.R_1^{IV,i}) \oplus L_0^{IV,i} \oplus R_0^{IV,i}\big) = 0$$

$$\begin{aligned}
\text{and } T_{MTC4} = \sigma \quad \Leftrightarrow \quad & \text{for each IV and } i = 1, \ldots, m_{IV} \\
& (L_0^{IV,i}, R_0^{IV,i}) \text{ input} \\
& f(IV.0.i.R_0^{IV,i}) \oplus f(IV.2.i.R_2^{IV,i}) = L_0^{IV,i} \oplus R_3^{IV,i} \\
& f(IV.1.i.R_1^{IV,i}) \oplus f(IV.3.i.R_3^{IV,i}) = R_0^{IV,i} \oplus L_3^{IV,i} \\
& f(IV.0.0.CS^{IV}) = Auth_{IV}
\end{aligned}$$

are independent events, it follows that

$$\begin{aligned}
&\Pr[\,\Psi \ni C \leftarrow A_{\text{ctxt}} \mid T_{MTC4} = \sigma] \\
&\leq \Pr[CS_{new}^{IV} = CS_{old}^{IV} \mid T_{MTC4} = \sigma] + \\
&\quad \Pr[CS_{new}^{IV} \neq CS_{old}^{IV} \text{ and} \\
&\qquad f(IV.0.0.CS_{new}^{IV}) = Auth_{IV} \mid T_{MTC4} = \sigma] \\
&= \frac{1}{2^n} + \frac{1}{2^n}\Big(1 - \frac{1}{2^n}\Big).
\end{aligned}$$

In each of the four cases, the probability that A_{ctxt} successfully returns an authenticated ciphertext after seeing an arbitrary transcript σ is bounded above by $1/2^n + 1/2^n(1 - 1/2^n)$. Therefore,

$$\begin{aligned}
&\mathbf{Adv}^{\text{int-ctxt}}_{MTC4,A_{\text{ctxt}}}(q,\mu) \\
&= \sum_{\sigma} \Pr[\, \Psi \ni C \leftarrow A_{\text{ctxt}} \mid T_{MTC4} = \sigma] \cdot \Pr[T_{MTC4} = \sigma] \\
&\leq \sum_{\sigma} \Big(\frac{1}{2^n} + \frac{1}{2^n}\Big(1 - \frac{1}{2^n}\Big)\Big) \cdot \Pr[T_{MTC4} = \sigma] \\
&= \frac{1}{2^n} + \frac{1}{2^n}\Big(1 - \frac{1}{2^n}\Big).
\end{aligned}$$

Equality holds when A_{ctxt} chooses Case 2, 3.a, or 4 for its final query. Hence,

$$\mathbf{Adv}^{\text{int-ctxt}}_{MTC4}(q,\mu) = \frac{1}{2^n} + \frac{1}{2^n}\Big(1 - \frac{1}{2^n}\Big). \qquad \square$$

It is important to note that in each of the above propositions we assumed our cryptographic primitive was perfectly random. Similar security results follow whenever we assume our primitive is a pseudorandom function.

2.2 Initialization Vector Considerations

In a typical cipher design, the codebook mode of operation is undesirable, since repeats in plaintext give repeats in ciphertext. CBC mode overcomes this to some extent, since repeats in the plaintext do not generally produce repeats in the ciphertext. Further, if two identical messages have different IVs, then they encrypt to different values. However, given a repeated IV, if two messages agree on the first few blocks of plaintext, then CBC mode will return ciphertexts that agree in the same positions.

Because of the counters, MTC4 has some CBC-like properties. In particular, repeated plaintext blocks in the same message encrypt to different values, and given identical messages with different IVs, the correlation between the two ciphertexts is negligible. However, given a repeated IV, two messages that have identical plaintext blocks in identical positions will produce identical ciphertext in that position. This is a little weaker than what occurs in CBC mode.

The assumption of unique IVs, counters, nonces and the like are often used in cryptographic designs to allow proofs of security in various adversarial models. The fact that when IVs are repeated, plaintext blocks in equal positions give equal ciphertext implies that the cipher can be distinguished from random. This is also true of most cryptographic designs that rely on unique message nonces to attain the desired level of security. Unfortunately, many of these other designs also have easily exploited weaknesses whenever an IV is repeated. For instance, the authentication

mechanisms of both XORMAC [5] and OCB [24] are trivially broken with a few messages processed with the same IV.

Since security under nonce reuse is difficult, the solution is often to insist the implementation never reuse nonces and so pass responsibility to the implementors. However, nonce reuse is a practical concern and may result from natural or malicious causes. This must be addressed widely from the management down through the hardware. For instance, if the particular hardware supporting an algorithm is rebooted, often sequence numbers and the like simply start over.

Our goal is to offer a scheme that addresses a pragmatic set of system-wide security issues, including security under nonce reuse, that has measured degradation in security when various suppositions are not met, rather than a more brittle approach where it is disastrous to reuse an IV. To this end, the inputs of our authentication designs are key dependent and never exposed. Even if an adversary has multiple messages processed with the same IV, the advantage in foiling the authentication mechanism is limited.

3 Reduced Manticore4 (RMTC4)

The MTC4 algorithm incorporates very attractive qualities of integral authentication with the positive aspects of CBC encryption using a single cryptographic primitive. The ability to execute the algorithm in parallel for multiple-block messages offers the potential for high-performance security. Additional efficiency options to speed up the construction are also possible. One option is to devise and use a faster cryptographic hash function. It is important to note that noninvertibility is a necessary attribute of the hash function used in the Manticore algorithms, but the requirement for collision resistance can be weakened. Only collision resistance of the keyed hash, with unknown key, is required. This observation offers fertile research ground for modifying existing or devising new hash functions suitable for MTC4.

Another option of speeding up MTC4 is to follow the approach of [21, 22] and require less computation in the first and last round functions. The following algorithm is a straightforward application of the ideas of Patel, et al [21] to MTC4. Reduced Manticore4 (RMTC4) is a four-round Feistel construction, where the second and third round functions are cryptographic hash functions, but the first and last rounds are not.
Let H be a cryptographically strong hash function mapping an arbitrary number of bits to h bits. Let K be a k-bit key and IV be a v-bit initialization vector. Let $M = m_1, m_2, \cdots, m_{2j}$ be the message to be encrypted, where each m_i is h bits in length. We assume the message M is padded with some suitable padding scheme, if necessary, so it is a multiple of $2h$ bits in length.

RMTC4
INPUT $(IV, M), K$
OUTPUT $(IV, C, AUTH)$
Set $k_1 \leftarrow H(K, 1)$, $k_2 \leftarrow H(K, 2)$, $k_3 \leftarrow H(K, 3)$, $k_4 \leftarrow H(K, 4)$
Set $CS \leftarrow 0$
For i from 1 to $2j - 1$ by 2 do
    Set $x \leftarrow m_i \oplus F_{k_1,k_2}(m_{i+1})$
    Set $y \leftarrow m_{i+1} \oplus H(K, IV, 0, i, x)$
    Set $CS \leftarrow CS \oplus x \oplus y$
    Set $c_{i+1} \leftarrow x \oplus H(K, IV, 1, i, y)$
    Set $c_i \leftarrow y \oplus F_{k_3,k_4}(c_{i+1})$
Set $AUTH = H(K, IV, 2, 0, CS)$
RETURN $(IV, C, AUTH)$

Depending on the relative speed of F verses H, RMTC4 encryption is up to twice as fast as MTC4. Naor and Reingold [19] give two alternatives for F to retain security of the cipher. The first is the notion of pairwise independence. They give the following as an example. Let G be a finite field. Then $F_{a,b}(x) \stackrel{def}{=} ax + b$, where $a \neq 0$, $b \in G$ are uniformly distributed and pairwise independent. In particular, one may use the field $G = GF(2^n)$, which can be efficiently implemented in hardware. See Sections 6 and 7 for results of our implementations of this linear function defined over prime fields with primes 160-bits and 128-bits in size.

4 2 Round Manticore (Pepper)

In this section, we present a version of Manticore with two rounds that we call Pepper because of its increased speed. Pepper is a very fast authenticated encryption scheme that uses the internal state of the cipher for authentication and is provably secure whenever the underlying primitives are pseudorandom. It is based on a 2 round Luby-Rackoff block cipher, which is the minimal number of rounds needed for cryptographic security. The running pre-authenticator is an XOR of the left-hand plaintext input and right-hand ciphertext output of each block cipher. This method provides the best cryptographic security out of any 2 round Luby-Rackoff based schemes that use an XOR of the internal state for the pre-authenticator. The authenticated encryption scheme Pepper is described as follows.

Let $M = m_1, m_2, \cdots, m_{2j}$ be the message to be encrypted, where each m_i is n bits in length. We assume the message M is padded with some suitable padding scheme, if necessary, so it is a multiple of $2n$ bits in length. Let u represent a 1-bit round counter of and z a block counter of fixed length. Let F_K be a pseudorandom function mapping $v+u+z+n$-bits to n-bits and IV a v-bit message specific initialization vector.

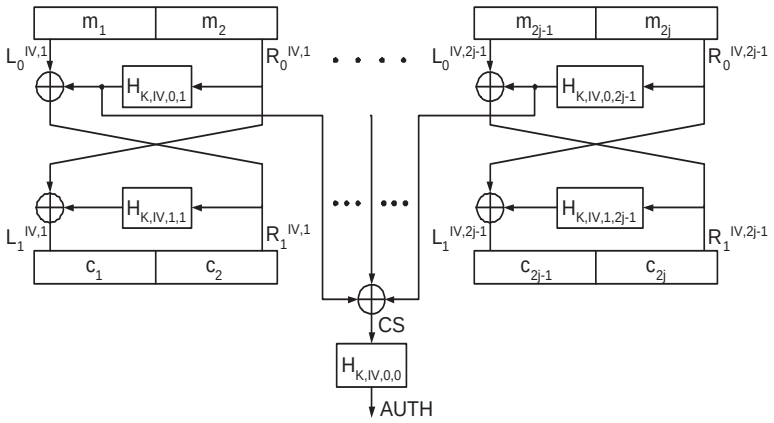

Figure 2. Block diagram for the Pepper Algorithm.

Pepper
INPUT $(IV, M), K$
OUTPUT $(IV, C, AUTH)$
Set $CS \leftarrow 0$
For i from 1 to $2j-1$ by 2 do
    Set $CS \leftarrow CS \oplus F_K(IV, 0, i, m_{i+1})$
    Set $c_{i+1} \leftarrow m_i \oplus F_K(IV, 0, i, m_{i+1})$
    Set $c_i \leftarrow m_{i+1} \oplus F_K(IV, 1, i, c_{i+1})$
Set $AUTH = F_K(IV, 0, 0, CS)$
RETURN $(IV, C, AUTH)$

4.1 Security of Pepper

Refer to Section 2.1 for security definitions. We assume a unique IV is used per encryption.

Proposition 3. *Let the authenticated encryption scheme Pepper have round function f, where f is a random function taking $n + v + u + z$-bits to n-bits. Then for any q queries totaling $\leq \mu$ bits, the distinguishing advantage $\mathbf{Adv}^{\mathrm{ind-cpa}}_{Pepper}(q, \mu) = 0$ in the IND-CPA model.*

Proof: Let A_{cpa} be any distinguishing algorithm and Γ the set of all transcripts σ such that $A_{\mathrm{cpa}}(\sigma) = 1$. Without loss of generality we will assume A_{cpa} is deterministic and chooses queries that maximizes its advantage. Let T_{Pepper} denote the transcript

generated by A_{cpa} given oracle access to *Pepper*. The advantage may be rewritten as

$$\begin{aligned}\mathbf{Adv}^{\text{ind-cpa}}_{Pepper,A_{\text{cpa}}}(q,\mu) &= \Big|\Pr[\mathbf{Exp}^{\text{ind-cpa-1}}_{Pepper,A_{\text{cpa}}}=1]-\Pr[\mathbf{Exp}^{\text{ind-cpa-0}}_{Pepper,A_{\text{cpa}}}=1]\Big| \\ &= \Big|\sum_{\sigma\in\Gamma}\big(\Pr[T_{Pepper}=\sigma \mid b=1]- \\ &\qquad\qquad \Pr[T_{Pepper}=\sigma \mid b=0]\big)\Big|.\end{aligned}$$

Notice that for each round in the encryption block, the IV/counter value pairs are unique. This implies the output of the each round, including the final authentication tag $AUTH$, are random. Therefore for every transcript σ,

$$\begin{aligned}\Pr[T_{Pepper}=\sigma \mid b=1] &= \Pr[T_{Pepper}=\sigma \mid b=0] \\ &= \frac{1}{2^{n\mu}}\cdot\frac{1}{2^{nq}}.\end{aligned}$$

where q and μ are the number of different IV's and bits queried. Therefore it follows,

$$\mathbf{Adv}^{\text{ind-cpa}}_{Pepper}(q,\mu)=0. \qquad \square$$

Proposition 4. *Suppose Pepper has round function f, where f is a random function taking $n+v+u+z$-bits to n-bits. Then for any q queries totaling $\leq \mu$ bits, the advantage in forging an authentication is*

$$\mathbf{Adv}^{\text{int-ctxt}}_{Pepper}(q,\mu) < \frac{\delta+2}{2^n} \quad \textit{where } \delta=\mu/2n.$$

Proof: Let A_{ctxt} be any algorithm as defined in the Integrity Awareness model and let σ denote the transcript generated by A_{ctxt}'s plaintext query/ciphertext replies. We will use the word ciphertext loosely to define the ciphertext/authentication tag pair. We observe that the final integrity check query made by A_{ctxt} must satisfy one of the following four cases:

Case 1: A_{ctxt} queries a ciphertext using an IV that it has not seen before.

Case 2: A_{ctxt} queries a ciphertext/IV pair using a previously seen IV, but the ciphertext is longer than the one contained in σ.

Case 3: A_{ctxt} queries a ciphertext/IV pair that is either shorter or the same length as that contained in σ, but at least one of the ciphertext blocks is changed by A_{ctxt}.

Case 4: A_{ctxt} queries a truncated version of a ciphertext/IV pair contained in σ.

Recall the advantage A_{ctxt} achieves in forging an authentication is defined as

$$\mathbf{Adv}^{\text{int-ctxt}}_{Pepper,A_{\text{ctxt}}} := \Pr[V_K(C) = 1].$$

If we let Ψ denote the set of all ciphertext such that $V_K(C) = 1$, the above equation can be rewritten as

$$\mathbf{Adv}^{\text{int-ctxt}}_{Pepper,A_{\text{ctxt}}} = \sum_{\sigma} \Pr[\,\Psi \ni C \leftarrow A_{\text{ctxt}} \mid T_{Pepper} = \sigma] \cdot \Pr[T_{Pepper} = \sigma].$$

If we can show that in each of the four cases A_{ctxt}'s advantage is bounded above by some negligible factor over all but a small number of transcripts, then our claim will follow. We now analyze each case separately.

Case 1 is straightforward since the authenticator has never seen an input with this type of IV. Hence $\Pr[\,\Psi \ni C \leftarrow A_{\text{ctxt}} \mid T_{Pepper} = \sigma] = 1/2^n$ for arbitrary transcripts σ. In Case 2, the output of both $R_0^{IV,i_{\text{new}}}$ and $R_1^{IV,i_{\text{new}}}$ for each new counter value i_{new} is random, since they are both independent of the transcript σ. Algorithm A_{ctxt} may choose to keep the same authentication tag $Auth_{IV}$ in its final query or replace it with another. It is not difficult to show the best choice for A_{ctxt} is not to change $Auth_{IV}$. Therefore the probability of success is no larger than

$$\begin{aligned}
&\Pr[\,\Psi \ni C \leftarrow A_{\text{ctxt}} \mid T_{Pepper} = \sigma] \qquad (2)\\
&\le \Pr\big[CS^{IV}_{new} = CS^{IV}_{old} \text{ or } CS^{IV}_{new} \neq CS^{IV}_{old} \text{ and}\\
&\qquad\qquad f\big(IV.0.0.CS^{IV}_{new}\big) = Auth_{IV} \mid T_{Pepper} = \sigma\big]\\
&= \frac{1}{2^n} + \frac{1}{2^n}\Big(1 - \frac{1}{2^n}\Big).
\end{aligned}$$

For Case 3, there are two different types of attacks that A_{ctxt} may choose from. Without loss of generality, we will assume that only one ciphertext block is changed and all the rest remain untouched. Changing several ciphertext blocks does not give A_{ctxt} an advantage. Given this assumption, the two cases are:

(3.a) For some i, the left hand output $L_1^{IV,i}$ is changed, but the right hand $R_1^{IV,i}$ remains the same.

(3.b) For some i, the right hand output $R_1^{IV,i}$ is changed.

For case 3.a, observe the value $R_0^{IV,i}$ must be different from the original, since $L_1^{IV,i}$ has been modified. The output $f(IV.0.i.R_0^{IV,i})$ must be random and hence following Equation 2, A_{ctxt}'s success can be no larger than $1/2^n + 1/2^n(1 - 1/2^n)$. Case 3.b follows the same reasoning, except now the probability that $CS^{IV}_{new} = CS^{IV}_{old}$ is no longer $1/2^n$. In particular, we have

$$\begin{aligned}
&\Pr[CS^{IV}_{new} = CS^{IV}_{old} \mid T_{Pepper} = \sigma]\\
&= \Pr[R^{IV,i}_{0,new} = R^{IV,i}_{0,old} \text{ or } R^{IV,i}_{0,new} \neq R^{IV,i}_{0,old} \text{ and}\\
&\qquad\qquad L^{IV,i}_{0,new} \oplus R^{IV,i}_{1,new} = L^{IV,i}_{0,old} \oplus R^{IV,i}_{1,old}]\\
&= \frac{1}{2^n} + \frac{1}{2^n}\Big(1 - \frac{1}{2^n}\Big).
\end{aligned}$$

Therefore,

$$\Pr[\,\Psi \ni C \leftarrow A_{\text{ctxt}} \mid T_{Pepper} = \sigma] = \frac{3}{2^n}\Big(1 - \frac{1}{2^n}\Big) + \frac{1}{2^{3n}}.$$

For the fourth and final case, suppose A_{ctxt} returns a query to a truncated version of an IV with m_{IV} blocks. Let Λ be the set of all transcripts σ such that,

$$\sum_{i=m'_{IV}+1}^{m_{IV}} \big(L_0^{IV,i} \oplus R_1^{IV,i}\big) = 0 \quad \text{for some } IV \text{ and } m'_{IV} < m_{IV} \tag{3}$$

Since $CS_{new}^{IV} = CS_{old}^{IV}$ if and only if Equation 3 holds, it follows that

$$\Pr[\,\Psi \ni C \leftarrow A_{\text{ctxt}} \mid T_{Pepper} = \sigma] = \begin{cases} 1 & : \text{ if } \sigma \in \Lambda \\ \frac{1}{2^n} & : \text{ else} \end{cases}$$

In each of the four cases, the probability that A_{ctxt} successfully returns an authenticated ciphertext after seeing a transcript $\sigma \notin \Lambda$ is bounded above by $3/2^n$. Therefore if we let $\delta = \mu/2n$, it follows that

$$\begin{aligned}
&\mathbf{Adv}^{\text{int-ctxt}}_{Pepper, A_{\text{ctxt}}}(q, \mu) \\
&= \sum_{\sigma} \Pr[\,\Psi \ni C \leftarrow A_{\text{ctxt}} \mid T_{Pepper} = \sigma] \cdot \Pr[T_{Pepper} = \sigma] \\
&< \sum_{\sigma \in \Lambda} \Pr[T_{Pepper} = \sigma] + \sum_{\sigma \notin \Lambda} \frac{3}{2^n} \cdot \Pr[T_{Pepper} = \sigma] \\
&< \Pr[\,\Lambda \ni \sigma \leftarrow T_{Pepper}] + \frac{3}{2^n} \\
&\leq \frac{\delta - 1}{2^n} + \frac{3}{2^n}.
\end{aligned}$$

Hence,

$$\mathbf{Adv}^{\text{int-ctxt}}_{Pepper}(q, \mu) < \frac{\delta + 2}{2^n}.$$

□

5 Cipher-State Mode of Encryption (CS)

MTC4 is a specific implementation of a Luby-Rackoff cipher using internal state for authentication. Here we examine a more general case and propose a simple method of adding authentication to any round-based block cipher as a mode of encryption. This provides a computationally low cost alternative to CBC mode, with stronger authentication properties. It is parallelizable, allowing faster execution. As with MTC4, the new idea is to tap into the middle of the encryption for authentication information. Of course, the security of the construction depends on the security

of the underlying cipher. The algorithm uses a $2n$-round, d-bit block cipher, E. Half-way through each block encryption, the state (middletext) is tapped and non-commutatively mixed into a running pre-authenticator, CS. The final value of the pre-authenticator is passed through a one-way function and appended to the message. The one-way function may be either created from the cipher, E, or a cryptographically strong hash function, H.

We use a simple linear feedback shift register (LFSR) as a pseudo-random number generator (PRNG) to pre-whiten the plaintext. The ciphertext is post-whitened with the same parameter, R. Multiple steps of the PRNG and the authentication combining operation are easy to compute, facilitating parallelism. The polynomial selected for the authentication combiner and the PRNG is the lexicographically least primitive polynomial, $p(x)$, of degree d. (A polynomial is primitive when x has maximum order). For the CS algorithm with 128-bit AES, we use $p(x) = x^{128} + x^7 + x^2 + x + 1$. Table 1 shows the least primitive polynomials for various degrees.

The algorithm given below and depicted in Figure 3 illustrates the CS construction for a j-block message, $M = m_1, \ldots, m_j$, initialization vector, IV, and encryption key, K.

CS
INPUT $(IV, M), K$
OUTPUT $(IV, C, AUTH)$
Set $CS \leftarrow 0$
Set $R \leftarrow E(K, IV \oplus K) \oplus K$
If $R = 0$, Set $R = K$
For i from 1 to j do
    Set $t \leftarrow E_{1-n}(K, m_i \oplus R)$
    Set $CS \leftarrow CS * x \pmod{p(x)} \oplus t$
    Set $c_i \leftarrow E_{(n+1)-2n}(K, t) \oplus R$
    Set $R \leftarrow R * x \pmod{p(x)}$
IF using E only, Set $AUTH = E(K, CS \oplus R) \oplus CS$
ELSE, Set $AUTH = H(K, CS, R)$
RETURN $(IV, C, AUTH)$

The block cipher is split into two roughly equal pieces, E_{1-n} and $E_{(n+1)-2n}$. E_{1-n} returns the middletext after completing half of the rounds. In the case of AES, this includes the initial *XOR* of the zeroth-round key, through five rounds of AES, finishing after the *XOR* of the fifth-round round key. The middletext is tapped to compute the running pre-authenticator. The second half of AES resumes with the middletext, starting with the S-box mapping of round 6, and continuing through round 10. Since the middletext is not altered, but merely tapped for authentication, the combined result of the two cipher halves is the same as an ordinary AES encryption of the

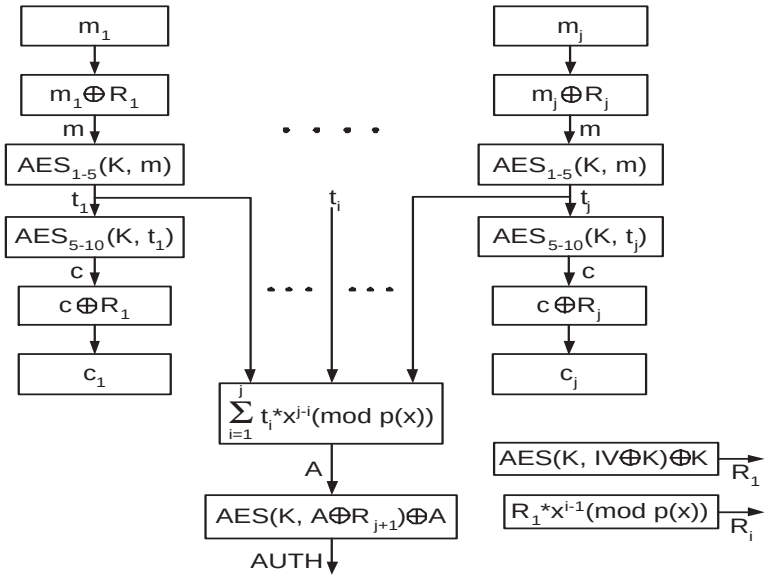

Figure 3. Block diagram for the CS Mode.

plaintext $m_i \oplus R$. Appendix A provides test vectors for CS mode using AES-128. For the additional-round variants of AES, the extra rounds are divided evenly between the two halves. For definiteness, any odd round goes with the first half.

We propose that new ciphers should define this tap point. The location is somewhat arbitrary, but should be far enough away from the start and end of the encryption so that the middletext has no simple relationship to either plaintext or ciphertext. Placing the tap point near the middle of the cipher provides the maximum protection against the differential attack sketched below.

The non-commutative combining operation in the pre-authenticator, CS, is cheap to compute, simple to advance multiple steps, and the results from separate computations are easy to combine. For both encryption and decryption, the authentication combiner and whitening PRNG can be easily adjusted for several kinds of parallelism: low-level parallelism where successive cipher blocks are parceled out; higher-level parallelism where larger chunks of the message are handled by different processors; and even pipelined chip architectures that process consecutive cipher blocks in consecutive clocks. The adjustments are straightforward for the more complex cases of pipelined hardware that intermixes processing for multiple messages, or when messages are broken into variable-sized pieces, or even when several kinds of parallelism are used together.

The IV is used to initialize the LFSR-PRNG for whitening the plaintext and concealing the raw ciphertext, and as an ingredient in the final message authenticator. Ideally, the IVs are unpredictable and cannot be influenced by an opponent. As with the MTC algorithms, nonrepeated IVs are preferred. However, the fact that

Table 1. Least primitive polynomials for selected degrees.

Degree	Primitive Polynomial	Low-order Portion (Hex)
64	$x^{64} + x^4 + x^3 + x + 1$	1B
96	$x^{96} + x^7 + x^6 + x^4 + x^3 + x^2 + 1$	DD
128	$x^{128} + x^7 + x^2 + x + 1$	87
160	$x^{160} + x^5 + x^3 + x^2 + 1$	2D
192	$x^{192} + x^8 + x^6 + x^4 + x^3 + x^2 + 1$	15D
224	$x^{224} + x^8 + x^7 + x^5 + x^4 + x^2 + 1$	1B5
256	$x^{256} + x^{10} + x^5 + x^2 + 1$	425
320	$x^{320} + x^4 + x^3 + x + 1$	1B
384	$x^{384} + x^{10} + x^6 + x^4 + x^3 + x^2 + 1$	45D
512	$x^{512} + x^8 + x^5 + x^2 + 1$	125
768	$x^{768} + x^{13} + x^8 + x^7 + x^5 + x^3 + 1$	21A9
1024	$x^{1024} + x^9 + x^8 + x^7 + x^5 + x + 1$	3A3

the authentication mechanism is hidden from the adversary's view means that the method has a certain amount of resistance to IV reuse.

As a final note, the use of an involutional block cipher is not recommended with this scheme. We don't know of such ciphers in widespread use.

5.1 Security Considerations

One security concern with CS-AES is that it could somehow leak information from the middle of an encryption. We consider this below.

The authenticator value $AUTH$ is computed in a finalization step from the pre-authenticator value CS. This step is either a strong hash or a strong cipher, so we expect no detectable relationship between the pre-authenticator values and authenticator values.

In the strongest attack we know, we assume a long period of IV reuse for the attacker to make headway. (If the IV is changed even occasionally, the attacker has no prospect of collecting a statistically useful amount of message data.) Any attack based on finding weak correlations between middletext values of related messages is doomed, since weak correlations will be destroyed by the finalization step.The only useful datum for an attacker is that two messages have the same authenticator. From this, he guesses that the pre-authenticator values are also the same, and he tries to deduce a relationship between the messages. Two different single-block messages (with the same IV and same key) will have different middletexts and therefore differing pre-authenticator values. So, nontrivial collisions of single-block messages are impos-

Table 2. Software performance comparisons.

Algorithm	MByte/Sec	Algorithm	MByte/Sec
SHA1	77	AES	69
MTC4-SHA1	14	CS-AES-AES	61
RMTC4-SHA1	18	CS-AES-SHA1	61
Pepper-SHA1	26	CS-AES-MD5	61
MD5	239	AES-CBC-HMAC-SHA-1	32
MTC4-MD5	28	AES-CBC-HMAC-MD5	44
RMTC4-MD5	29		
Pepper-MD5	56		

sible. (We can take this a step further: Take a multi-block message and vary one particular block within it, running through all possible values. Then the ciphertext and middletext will run through all values, and so will the pre-authenticator. So, two messages which match in all but one block will have differing pre-authenticators.)

For two-block messages, the attacker can try to engineer a pre-authenticator collision using differentials. (XOR-based differentials propagate transparently through the PRNG whitening step.) He uses a two-block differential (δ_1, δ_2), and hopes that the encryption of the two-block messages (P_1, P_2) and ($P_1 \oplus \delta_1$, $P_2 \oplus \delta_2$) will produce compatible middletext differentials. The middletexts are (M_1, M_2) and (N_1, N_2). For a pre-authenticator collision, the equation $M_1 * x \pmod{p(x)} \oplus M_2 = N_1 * x \pmod{p(x)} \oplus N_2$ must hold. This will happen if the second-block differential $M_2 \oplus N_2$ is a one-bit left shift of the first-block differential $M_1 \oplus N_1$. Also, the high-order-bit of the first-block differential is 0, so no carry occurs in the multiplication by x. The chance of a match is the square of the individual probabilities for the half-cipher differentials, which is comparable to the chance of a differential propagating through the full cipher. For a strong cipher, like AES, this is negligible.

6 Software Implementation Results

To test the performance of our algorithms, we chose Wei Dai's Crypto++ 5.1 C++ cryptographic library [7] as a common framework. The Crypto++ library uses Barreto's implementation of AES [3]. Test programs were compiled using Microsoft Visual C++ 7.1 and executed on a Dell Precision 340 computer with 2.53 GHz Pentium IV processor. 1024-byte messages were used. Table 2 provides comparative figures of the Manticore algorithms and the CS-AES mode against core cryptographic primitives and the typical usage of AES in CBC mode for encryption with HMAC authentication [14].

Our implementation of the MTC4 algorithm is limited by the speed of the primitives. Improvements on the hash primitives are worth investigating. One simple enhancement is to use only the compression functions of the hash algorithms (i.e., no byte-swapping or padding). The number of bytes used for the round and block counters in MTC4 were 1 and 4 respectively. The timing estimates of MTC4 discussed in Section 2 suggest that MTC4-SHA-1 and MTC4-MD5 should be approximately 6 and 8 times slower, respectively, than simply hashing the same message. Taking the authentication steps and extra *XOR* operations into account, these estimates are borne out. MD5 [23] has recently been shown to have collisions [25], but these attacks don't work against a keyed hash, as we use it in our Manticore algorithms.

The RMTC4 algorithms are implemented using a pairwise independent permutation for Rounds 1 and 4. Specifically, we compute $F_{k_i,k_{i+1}}(m) = (k_i m \pmod p) + k_{i+1} \pmod{2^h}$, where k_i and k_{i+1} are derived from K using the available hash function (i.e., $k_i = H(K, i)$), and p is chosen to be $(2^{160} - 47)$ and $(2^{128} - 159)$ for SHA-1 and MD5 respectively and $h = 160$ and 128 respectively. This function results in a 3 times speedup of Rounds 1 and 4 for RMTC4-SHA-1 and a 2 times speedup for RMTC4-MD5.

The CS mode of encryption can run about as fast as the underlying cipher plus a small overhead for authentication in each round and at the end. In software, a significant fraction of the overhead is due to the mixing operation in the whitening parameter, R, and for the running authentication, A. The speed of this operation would be negligible in hardware.

The most common computers today use the register-challenged Pentium microprocessor. Without care, an AES encryption program can wreak havoc with a compiler's register allocation on the Pentium. To counter this problem, we mention a few programming tricks for improving performance.

Typically, the plaintext message will be supplied as a large array of bytes, with multiple calls to the encryption routine to process the entire array. For the CS-AES implementation, the pre-whitening of the plaintext can be done in a separate pass over the array before any calls to the encryption routine. If assembly language is available, the entire R variable can be kept in registers. The carry bit and Pentium rotate instruction are used in the multiplication $R * x$. Similarly, the post-whitening of the ciphertext can be carried out as a separate pass over the ciphertext array, after all the encryptions are done.

The running pre-authenticator A can be pre-multiplied by x before entering the encryption routine. Then, the middle round of the encryption need only *XOR* the middletext cipher state into A. This makes a minimal disturbance to the AES encryption routine. Even better is to simply record all of the middletext cipher states in a separate array, and then, after all the encryptions are completed, compute A with a pass over the array of saved middletext.

Another aspect of highly pipelined machines is the large penalty for a mispredicted-predicted branch. The typical way of coding the modular reduction by the LFSR polynomial $p(x)$ is to check the high order bit with an IF statement, and if needed, *XOR* a small fixup constant, F, into the reduced value. The no-branch way to do this is to always *XOR*, with a value of either 0 or F, computed from the high-order bit. This is most easily done with a conditional-move instruction, when available. The Pentium SETC instruction, which moves the carry bit into a register can also be used to one's advantage. Alternatively, if H is the high order bit, the expression $((H << 8) - 1)\ AND\ F$ will be 0 when H is 0, and F when H is 1. Although somewhat ugly, this is better than the IF statement.

Finally, we note that we have utilized Brian Gladman's highly optimized AES implementations [10] and identified a slight complication in the decryption/authentication phase of the CS-AES mode. Gladman's decryption routine uses a different byte arrangement of the cipher state and never explicitly calculates the end-of-round-5 middletext that we use for the authentication. We chose to write a modified round 5 to compute the required middletext.

The MTC4 and RMTC4 algorithms and the CS mode of encryption are block-parallelizable, which will make implementations with a parallelization capability faster with no loss of security. Future work will include optimizing our algorithms as well as fully utilizing Gladman's efficient AES implementations.

7 Hardware Implementation Results

In this section, we present hardware performance estimates of our encryption modes compared to their underlying cryptographic primitives. Table 3 summarizes the simulation results of VHDL and Verilog code synthesized to a target CMOS $0.5\mu m$, $5V$ library, a radiation-hardened technology. A 30 MHz clock rate was used for the timing figures.

For the MTC4, RMTC4, and Pepper algorithms, we implemented SHA-1 and a 160-bit modular multiplier in a VHDL Register-Transfer-Level design. The SHA-1 algorithm and modular multiply were optimized to use a shift register for the main data storage, which reduced the area used, with a corresponding increase in speed. In RMTC4, we use GF(2) modular arithmetic operations to compute $F_{k_i,k_{i+1}}(m)$ in Rounds 1 and 4.

8 Conclusion

We take advantage of the internal state of a secure block cipher to provide secure authentication. The ciphers we present possess the beneficial attributes of CBC mode

Table 3. Hardware performance comparisons.

Algorithm	Gate Count	Timing
SHA-1	13K	193 clocks/512-bit block
MTC4-SHA-1	25K	193 + 776 clocks/320-bit block
RMTC4-SHA-1	43K	193 + 466 clocks/320-bit block
Pepper-SHA-1	20K	193 + 391 clocks/320-bit block
AES	26K	14 clocks/128-bit block
CS-AES-AES	45K	36 clocks + 16 clocks/128-bit block

without the performance limitations. The arbitrary input size and inherent strength of cryptographic hash functions allow for an extremely flexible round function for a Feistel cipher and a proof of security for privacy and integrity. It allows various length blocks, IV's, counters, and keys, and is able to do so without having to design a new cipher to account for each case. The specification of the MTC4 and Pepper ciphers also allows for an extremely fast authentication step. Not only can the encryption and decryption process be parallelized and/or pipelined, the speed of typical cryptographic hash functions comes to bear and provides for a reasonably fast cipher design. Research opportunities exist for exploring faster hash functions appropriate for the MTC4 and Pepper round functions. We also present a Cipher-State mode of encryption that provides authentication from the internal state of a multi-round block cipher, such as AES. All of our algorithms provide an encryption mode with little overhead for authentication, resistance against IV reuse (except Pepper), positive CBC mode qualities, and opportunities for parallelization.

References

[1] E. Anderson, C. Beaver, T. Draelos, R. Schroeppel, M. Torgerson, "ManTiCore: Encryption with Joint Cipher-State Authentication," *Australasian Conference on Information Security and Privacy*, LNCS 3108, Springer-Verlag, 440-453, 2004.

[2] R. Anderson, E. Biham, "Two Practical and Provably Secure Block Ciphers: BEAR and LION," *Fast Software Encryption*, LNCS 1039, Springer-Verlag, 113-120, 1996.

[3] P. Barreto, "The Block Cipher Rijndael," http://www.esat.kuleuven.ac.be/ ˜rijmen/rijndael/.

[4] M. Bellare, A. Desai, E. Jokipii, P. Rogaway, "A Concrete Security Treatment of Symmetric Encryption," *In FOCS '97*, IEEE, 394-403.

[5] M. Bellare, R. Guerin, P. Rogaway, "XOR MACS: New Methods for Message Authentication using Finite Pseudorandom Functions," *CRYPTO 1995*, LNCS 963, Springer-Verlag, 1995.

[6] M. Bellare, C. Namprempre, "Authenticated Encryption: Relations Among Notions and Analysis of the Generic Composition Paradigm," *ASIACRYPT 2000*, LNCS 1976, Springer-Verlag, 531-545, 2000.

[7] W. Dai, "Crypto++ Library," http://www.eskimo.com/weidai/ cryptlib.html.

[8] Department of Commerce/NIST, "Secure Hash Standard," FIPSPUB 180-1, 2001.

[9] Department of Commerce/NIST, "Advanced Encryption Standard," FIPSPUB 197, 2001.

[10] B. Gladman, "Implementations of AES (Rijndael) in C/C++ and Assembler," http://fp.gladman.plus.com/cryptography_technology/rijndael/.

[11] V. D. Gligor, P. Donescu, "Fast Encryption and Authentication: XCBC Encryption and XECB Authentication Modes," *Fast Software Encryption (FSE)*, LNCS 2355, Springer-Verlag, 92-108, 2001.

[12] C. Jutla, "Encryption Modes with Almost Free Message Integrity," *Advances in Cryptology - EUROCRYPT 2001*, LNCS 2045, Springer-Verlag, 2001.

[13] L. Knudsen, "The Security of Feistel Ciphers with Six Rounds or Less," *J. of Cryptology*, Volume 15 # 3, 207-222, 2002.

[14] H. Krawczyk, M. Bellare, R. Canetti, "HMAC: Keyed hashing for message authentication," Internet RFC 2104, February 1997.

[15] C. H. Lim, "Message Encryption and Authentication Using One-Way Hash Functions," *Proc. of 3rd Annual Workshop on Selected Areas in Cryptology (SAC '96)*, Queens University, Kingston, Ontario, Canada, 117-131, 1996.

[16] M. Luby, C. Rackoff, "How to Construct Pseudorandom Permutations from Pseudorandom Functions," *SIAM Journal of Computing*, 17: # 2, 373-386, 1988.

[17] S. Lucks, "Faster Luby-Rackoff Ciphers," *FSE*, LNCS 1039, 189-203, 1996.

[18] U. M. Maurer, "A Simplified and Generalized treatment of Luby-Rackoff Pseudorandom Permutation Generators," *EUROCRYPT 1992*, LNCS 658, 239-255, 1992.

[19] M. Naor, O. Reingold, "On the Construction of Pseudo-Random Permutations: Luby-Rackoff revisited," *J. of Cryptology*, Volume 12 # 1, 29-66, 1999.

[20] NIST Modes of Operation for Symmetric Key Block Ciphers, http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/.

[21] S. Patel, Z. Ramzan, G. S. Sundaram, "Towards Making Luby-Rackoff Ciphers Optimal and Practical," *Fast Software Encryption*, LNCS 1636, 171-185, 1999.

[22] S. Patel, Z. Ramzan, G. S. Sundaram, "Sha-zam: A Block Cipher. Fast as DES, Secure as SHA," *Contribution for the Third-Generation Partnership Project (3GPP)*, December 6, 1999.

[23] R. Rivest, "The MD5 message digest algorithm," IETF Network Working Group, RFC 1321, April 1992.

[24] P. Rogaway, M. Bellare, J. Black, T. Krovetz, "OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption," *8th ACM Conf. on Computer and Communications Security*, ACM Press, 2001.

[25] X. Wang, D. Feng, X. Lai, H. Yu, "Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD," Cryptology ePrint Archive: Report 2004/199.

[26] D. Whiting, R. Housley, N. Ferguson, "Counter with CBC-MAC (CCM)," June 2002. http://csrc.nist.gov/encryption/modes/proposedmodes/

A Test Vectors for CS-AES-128

The following test parameters and outputs are for the CS algorithm using AES-128 as the cipher and both AES-128 and SHA-1 as the one-way function for the final authenticator.

$\boldsymbol{K}$:	000102030405060708090A0B0C0D0E0F
$\boldsymbol{IV}$:	0123456789ABCDEF0123456789ABCDEF
$\boldsymbol{m_1}$	00112233445566778899AABBCCDDEEFF
$\boldsymbol{R_1}$:	FDED29920913A3DA8C9ECA2F0FD434AA
$\boldsymbol{m_1} \oplus \boldsymbol{R_1}$:	FDFC0BA14D46C5AD04076094C309DA55
$\boldsymbol{t_1}$:	C31FDB743AA199CB78AA156AED162EB9
$\boldsymbol{A_1}$:	C31FDB743AA199CB78AA156AED162EB9
$\boldsymbol{AES}(\boldsymbol{K}, \boldsymbol{m_1} \oplus \boldsymbol{R_1})$:	FEE2017432993F8D81E1251E9BD6125E
$\boldsymbol{c_1}$:	030F28E63B8A9C570D7FEF31940226F4
$\boldsymbol{R_2}$:	FBDA5324122747B5193D945E1FA869D3
$\boldsymbol{A_1} \oplus \boldsymbol{R_2}$:	38C588502886DE7E61978134F2BE476A
$\boldsymbol{AES}(\boldsymbol{K}, \boldsymbol{A_1} \oplus \boldsymbol{R_2})$:	08A2C2E93DFEEBEBEDD5CD4AB7351526
$\boldsymbol{AUTH\text{-}AES}$:	CBBD199D075F7220957FD8205A233B9F
$\boldsymbol{AUTH\text{-}SHA\text{-}1}$:	ECFA375F615DB07834F50C7B9C3B08A9C9D3F12F

The second test is an iterative multi-block test with a total of 1,000,000 blocks. The input parameters are the same as the first test, but the ciphertext of the current block is used as the plaintext in the next block. For this test, intermediate results are given for only the first two blocks and the final block.

$\boldsymbol{K}$:	000102030405060708090A0B0C0D0E0F
$\boldsymbol{IV}$:	0123456789ABCDEF0123456789ABCDEF
$\boldsymbol{m_1}$:	00112233445566778899AABBCCDDEEFF
$\boldsymbol{R_1}$:	FDED29920913A3DA8C9ECA2F0FD434AA
$\boldsymbol{m_1 \oplus R_1}$:	FDFC0BA14D46C5AD04076094C309DA55
$\boldsymbol{t_1}$:	C31FDB743AA199CB78AA156AED162EB9
$\boldsymbol{A_1}$:	C31FDB743AA199CB78AA156AED162EB9
$\boldsymbol{AES(K, m_1 \oplus R_1)}$:	FEE2017432993F8D81E1251E9BD6125E
$\boldsymbol{c_1}$:	030F28E63B8A9C570D7FEF31940226F4
$\boldsymbol{m_2}$:	030F28E63B8A9C570D7FEF31940226F4
$\boldsymbol{R_2}$:	FBDA5324122747B5193D945E1FA869D3
$\boldsymbol{m_2 \oplus R_2}$:	F8D57BC229ADDBE214427B6F8BAA4F27
$\boldsymbol{t_2}$:	E005EF3D83A7F60BD8486A7B15CC93DD
$\boldsymbol{A_2}$:	663A59D5F6E4C59D291C40AECFE0CE28
$\boldsymbol{AES(K, m_2 \oplus R_2)}$:	778A4DF11D9CAB517F68DD65E6053BFA
$\boldsymbol{c_2}$:	8C501ED50FBBECE46655493BF9AD5229
	⋮
$\boldsymbol{m_{1,000,000}}$:	8C9A9C08367E40D4A0BDF5405E0A8358
$\boldsymbol{R_{1,000,000}}$:	8F3461728ECD3A7B3D3CC89808967071
$\boldsymbol{m_{1,000,000} \oplus R_{1,000,000}}$:	03AEFD7AB8B37AAF9D813DD8569CF329
$\boldsymbol{t_{1,000,000}}$:	0DF19348FE1FD9EFEC91D9843B13566A
$\boldsymbol{A_{1,000,000}}$:	D72D708155DB739339471E4D1EAB6D85
$\boldsymbol{AES(K, m_{1,000,000} \oplus R_{1,000,000})}$:	7C73C0F8EA2923A80A65651995CAA8C5
$\boldsymbol{c_{1,000,000}}$:	F347A18A64E419D33759AD819D5CD8B4
$\boldsymbol{R_{1,000,001}}$:	1E68C2E51D9A74F67A799130112CE065
$\boldsymbol{A_{1,000,000} \oplus R_{1,000,001}}$:	C945B26448410765433E8F7D0F878DE0
$\boldsymbol{AES(K, A_{1,000,000} \oplus R_{1,000,001})}$:	4A49085400CF9BA45A84774B6020EF55
$\boldsymbol{AUTH\text{-}AES}$:	9D6478D55514E83763C369067E8B82D0
$\boldsymbol{AUTH\text{-}SHA\text{-}1}$:	29520E37A0D635C41694F30AA9C09FE5AF525D2B

DISTRIBUTION:

2 MS 0785
W. E. Anderson, 5514

5 MS 0785
T. J. Draelos, 5514

2 MS 0785
C. L. Beaver, 5514

2 MS 0785
R. C. Schroeppel, 5514

2 MS 0785
M. D. Torgerson, 5514

1 MS 0785
T. S. McDonald, 5514

1 MS 0785
R. E. Trellue, 5501

1 MS 0451
S. G. Varnado, 5500

1 MS 1202
W. R. Cordwell, 5943

1 MS 1202
A. N. Campbell, 5940

1 MS 1072
R. A. Gonzales, 1735

2 MS 1072
R. D. Miller, 1735

1 MS 1072
K. K. Ma, 1735

1 MS 9018
Central Technical Files, 8945-1

2 MS 0899
Technical Library, 9616

2 MS 0612
Review & Approval Desk, 9612 For DOE/OSTI

1 MS 0123
LDRD Program Office (Attn: Donna Chavez), 1011