ManTiCore: Encryption with Joint Cipher-State Authentication

Cheryl Beaver, Timothy Draelos, Richard Schroeppel, Mark Torgerson

Sandia National Laboratories** Albuquerque, NM 87185-0785
{cbeaver, tjdrael, rschroe, mdtorge}@sandia.gov

Abstract. We describe a new method for authenticated encryption, which uses information from the internal state of the cipher to provide the authentication. This methodology has a number of benefits. The encryption has properties similar to CBC mode, yet the encipherment and authentication mechanisms can be parallelized and/or pipelined. The authentication overhead is minimal, so the computational cost of the authenticated encryption is very nearly that of the encryption process. Also, the authentication process remains resistant against some IV reuse. We present a class of encryption algorithms that are based on cryptographic hash functions. Because of the hash function construction, the MTC4 class of methods supports variable encryption block sizes up to twice the hash output block length and trivially supports variable key lengths. We also provide a more general construction for using the internal state of any round-based block cipher as an authenticator. We give a concrete example of the general construction that uses AES as the encryption primitive. We provide performance measurements for all of our constructions.

Keywords: Authenticated Encryption, Luby-Rackoff, Feistel, Middletext, Hash, Cipher

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. For instance, it is accepted that cipher block chaining (CBC) mode gives certain practical security benefits when encrypting large amounts of data with the same key. However, CBC cannot be parallelized or efficiently pipelined and so is generally undesirable for high-speed data transmissions. Furthermore, the application

** Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

of a typical message authentication code (MAC) for data integrity is a process that must be handled separately and in addition to the encryption.

It would be desirable if one could speed up the entire process by using information gained from the encryption step to accomplish authentication. Recent research has been conducted in establishing authenticated encryption schemes that are more efficient than the current standards and practice [9, 10, 21]. In particular, OCB [21] is parallelizable and offers CBC-like encryption and authentication with only two extra block cipher invocations over that needed for encipherment alone.

The goal of our work is to take a new and different approach - to examine the possibility of using a cipher's internal state as inputs for an authentication mechanism. They are also parallelizable, yet exhibit many of the practical benefits of CBC mode. The authentication mechanism adds minimal cost to the encryption process. The methods also offer security in the face of IV reuse, which, to our knowledge, existing authenticated encryption mechanisms do not provide. 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.

From the highest point of view, we examine block ciphers comprised of $2n$ rounds. The authentication tag for an encrypted message is a function of the encryption state after n rounds. Our first construction is based on a four round Feistel network with cryptographic hash functions as round functions. Many of the construction components necessary for secure encryption and authentication can be added into the round functions because of the hash algorithm's ability to accept arbitrary inputs.

Extensive research has been conducted on the security and construction of low round Feistel network ciphers. In [14], Luby and Rackoff show how to construct $2n$-bit pseudorandom permutations using a Feistel network. Their constructions are secure against any adversary who has combined adaptive chosen plaintext and ciphertext attacks. Their original construction relied on a four-round Feistel network, with each round using a pseudorandom function. In [11] Knudsen provides a nice survey and analysis of the security bounds for low round Feistel constructions.

Much research has also been conducted in finding practical instantiations of a low round Feistel construction using cryptographic hash algorithms as round functions. In [1, 15], the authors examine three round ciphers, while Lim [13] uses cryptographic hash functions in a four round construction. Naor and Reingold [17] show that one may achieve the same level of security if the first and last rounds are replaced with two $2n$-bit permutations that act as strongly universal hash functions, rather than a full cryptographic hash. The hope of their construction was to replace two expensive calls to a pseudorandom function with two non-cryptographic, strongly universal hash functions. In a similar fashion, Patel, Ramzan, and Sundaram [18] show how one can replace the two inner rounds of a four-round Feistel network with a single, truly random primitive, and the outer two rounds with *Bi-symmetric* ϵ-Δ universal hash functions.

For our second construction, we show how to take an arbitrary round-based cipher and extend it to an authenticated encryption scheme with minimal additional overhead for the authentication process. 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 is secure against IV reuse. We use AES as a concrete example. We also provide performance figures for the various constructions.

2 ManTiCore4

As its namesake implies, our construction, ManTiCore (MTC4), is comprised of a number of common elements. The basic cipher elements use cryptographic hash functions in a four-round Feistel network and can be viewed as a variant of [13]. The attractive feature of cryptographic hash functions is their ability to accept arbitrary sized inputs. This allows us to insert an IV and block counters into the round inputs in a simple fashion.

This construction can be used to create a block cipher of any even bit length up to twice the hash size. Of course, the hash may be truncated to produce shorter block sizes. The size of the key is adjustable to any size desired by the user of the cipher and impacts performance only when the input block size of the hash function is exceeded.

Let H be a cryptographically strong hash function that maps 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. Assume here that the message M is padded with some suitable padding scheme, if necessary, so that it is a multiple of $2h$ bits in length.

The following is the MTC4 authenticated cipher.

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

Even though the cryptographic hash functions accept arbitrary-length inputs, typically they do so by processing a block of b bits at a time. For instance, SHA-1 [7] and MD5 [20] operate on 512-bit input blocks and output 160 and 128 bits respectively. From an efficiency standpoint, it makes sense to limit the size of the parameters so that the arguments fit within one input block, that is $k + v + 2 + \log_2(j) + h \leq b$. 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 and MTC4-MD5 should be on the order of $128/512 * 1/2 = 1/8$ times as fast as MD5. 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 authenticators to complete the process. Section 4 gives a detailed timing comparison of these with other ciphers including authentication. We fully expect others to explore the notion of authenticating internal cipher-states, resulting in better and better performance in the future. Figure 1 gives a representation of MTC4.

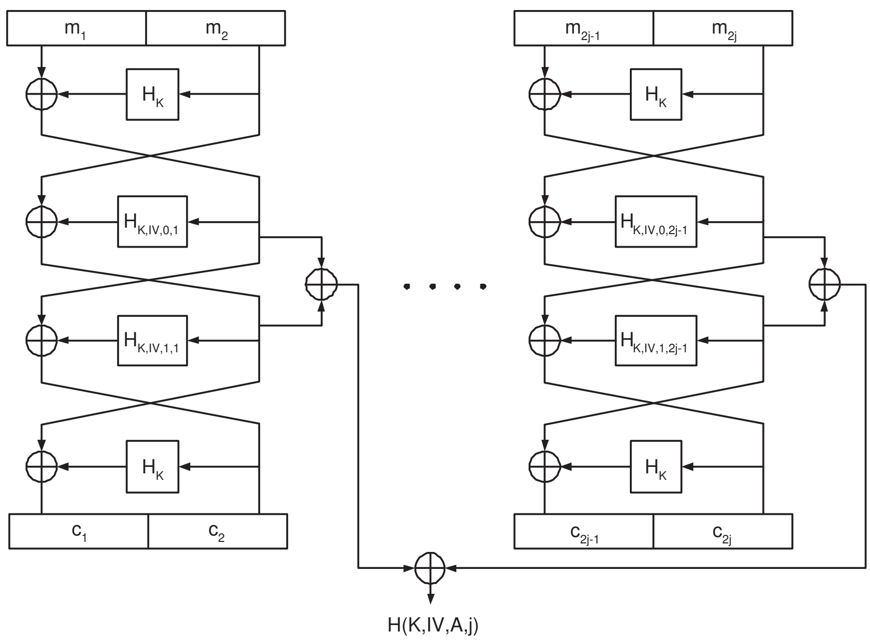

Fig. 1. MTC4

2.1 Security Considerations

When MTC4 is viewed strictly as a block cipher, the non-invertibility property of the cryptographic hash plays an important role in the cipher's strength. Provided $k \leq h$, a cryptographic hash is an ideal function in the sense of [11]. That is, it requires on the order of 2^k hash evaluations to recover the key. This ideal property ensures that an exhaustive key search is the most efficient method to recover the key.

For instance, in [11] it has been shown that with an ideal round function, a $2h$-bit four round Feistel cipher can be distinguished from a random $2h$-bit permutation with on the order of $2^{\frac{h}{2}}$ chosen plaintexts and work on the order of 2^k round function evaluations. Because of the counters in our construction, the only way to get the necessary $2^{\frac{h}{2}}$ chosen plaintexts is to have $2^{\frac{h}{2}}$ distinct messages encrypted with the same IV. In any event, the key recovery work requirements using the distinguisher is the same as exhaustion, but with larger data requirements.

MTC4 is not purely a cipher. There is some opportunity for the authentication mechanism to leak information about the cipher. Information leakage may provide an enhanced distinguisher. A better distinguisher can reduce the amount of data needed from $2^{\frac{h}{2}}$ chosen plaintexts to some lesser amount, but it cannot reduce the key recovery work requirements to below 2^k hash evaluations.

On the other hand, if $k > h$, then a key recovery attack is bounded from below by the minimum of 2^k hash evaluations and the work to find a hash preimage of a particular form – one that reveals the key. However, any attack that exploits the hash inversion option must have the hash outputs to invert. The known distinguishers all involve guessing key information and deriving the round outputs. The distinguishers do not first produce the hash outputs, which can then be inverted. Furthermore, the IV in each message requires that multiple messages be encrypted with the same IV before enough data can be acquired to distinguish.

Aside from key recovery attacks, there are two other notions of security that one may consider when designing a secure authenticated encryption scheme. The first is message privacy, which looks at the security of the encryption and decryption process. The second is ciphertext integrity, which assures authenticated ciphertext is computationally difficult to generate. Since MTC4's message security and authentication are integrated, one must be cautious that the combination does not leak enough information to allow an adversary to mount an attack on these systems.

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

Because of the counters, MTC4 has the following CBC-like properties.

1. Repeated plaintext blocks within the same message encrypt to different values.

2. 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 imply that the cipher can be distinguished from random and thus fails common security criteria. 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 [4] and OCB [21] are trivially broken with a few messages processed with the same IV.

In general, the philosophy seems to be that if the highest level of security cannot be attained under nonce reuse, then it doesn't matter what weaknesses there are in the method. The typical solution is to simply insist that the implementation never reuse nonces and thus pass the responsibility to the implementors. However, nonce reuse is a very practical concern. It is very difficult to guarantee that an implementation of some security mechanism will never produce a repeated nonce, either by natural or malicious means. This issue must be addressed by everything from the management system down through the hardware. For instance, if the particular hardware supporting an algorithm is rebooted, what happens to sequence numbers and the like? Often they simply start over.

Moving away from an all-or-nothing security mentality is the motivation of our designs. The hope is the construction of an efficient authenticated encryption mechanism that has a measured degradation in security when various security 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.

2.2 Reduced ManTiCore4 (RMTC4)

The MTC4 algorithm incorporates very attractive qualities of integral authentication and CBC-like 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, but not collision resistance, is a necessary attribute of a hash function for our cipher. This observation offers fertile research ground for modifying existing or devising new hash functions suitable for MTC4. Another option is to follow the approach of [18, 19] and require less of the first and last round functions.

The following algorithm is a straightforward application of the ideas of Patel, et al. 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.

RMTC4
INPUT $(IV, M), K, K_1, K_2$
OUTPUT $(IV, C, AUTH)$
Set $A \leftarrow 0$
For i from 1 to $2j-1$ by 2 do
  Set $x \leftarrow m_i \oplus F(K_1, m_{i+1})$
  Set $y \leftarrow m_{i+1} \oplus H(K, IV, 0, i, x)$
  Set $A \leftarrow A \oplus x \oplus y$
  Set $c_{i+1} \leftarrow x \oplus H(K, IV, 1, i, y)$
  Set $c_i \leftarrow y \oplus F(K_2, c_{i+1})$
Set $AUTH = H(K, IV, A, j)$
RETURN $(IV, C, AUTH)$

K_1 and K_2 are derived from K using the available hash function, H, as $K_1 = H(K, 1)$ and $K_2 = H(K, 2)$. Depending on the relative speed of F verses H, RMTC4 encryption is up to twice as fast as MTC4. Naor and Reingold [17] 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 Section 4 for results of our software implementation of this linear function defined over prime fields defined by primes 160-bits and 128-bits in size.

3 General Construction for Encryption with Authentication

MTC4 is a specific implementation of a Luby-Rackoff cipher using the internal state for authentication. In this section we examine the more general case and propose a simple method of adding authentication to any round-based block cipher. This method provides a computationally low cost alternative to CBC mode, with stronger authentication properties. It also has the virtue of being 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 of the cipher (the middletext) is tapped and non-commutatively mixed into a running authenticator, A. The final value of the 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 also post-whitened. 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). Table 1 shows the least primitive polynomials for various degrees.

The algorithm given below, GCSA for General Cipher-State Authentication, illustrates this construction.

GCSA
INPUT $(IV, M), K$
OUTPUT $(IV, C, AUTH)$
Set $A \leftarrow 0$
Set $R \leftarrow E(K, IV \oplus K) \oplus K$
If $R = 0$, Set $R = 1$
For i from 1 to j do
    Set $R \leftarrow R * x \pmod{p(x)}$
    Set $t \leftarrow E_{1-n}(K, m_i \oplus R)$
    Set $A \leftarrow A * x \pmod{p(x)} \oplus t$
    Set $c_i \leftarrow E_{(n+1)-2n}(K, t) \oplus R$
IF using E only, Set $AUTH = E(K, A \oplus IV) \oplus A$
ELSE, Set $AUTH = H(K, A, IV)$
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 of the block cipher. 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 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 plaintext $m_i \oplus R$. The first half of AES uses the first six round keys, and the second half uses the last five round keys (AES has a total of eleven round keys). 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.

For other ciphers with a definite round structure, the midpoint division is obvious. We propose that new ciphers should define the tap point as part of their specification. The location of the tap point is somewhat arbitrary, but it should be far enough away from the beginning and end of the encryption so that the selected middletext has no simple relationship to either the plaintext or the ciphertext.

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 non-commutative combining operation used for the running authenticator A 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 the whitening PRNG can be easily adjusted for several kinds of parallelism: low-level parallelism where successive cipher blocks are parceled out to different pieces of hardware; higher-level parallelism where larger chunks of the message are handled by different processors; and even pipelined chip architectures that process consecutive blocks of cipher 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.

An IV is supplied with each message to be encrypted or decrypted. 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 controlled or influenced by an opponent. As with the MTC algorithms, unrepeated IVs are preferred. However, the fact that 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.

4 Empirical Timing Results

Our implementation of the ManTiCore ciphers utilizes existing and accepted cryptographic hash functions in a four round block cipher and is limited by the speed of these primitives. Improvements on the hash primitives used in MTC4 and RMTC4 are possible and worth investigating. The general construction of encryption with cipher-state authentication can run approximately as fast as the underlying cipher plus a small overhead for authentication in each round and at the end of the encipherment process. The ManTiCore algorithms and the General Construction for cipher-state authenticated encryption are block-parallelizable, which will make implementations with a parallelization capability faster with no loss of security.

The ManTiCore ciphers were implemented in C++ using Wei Dai's Crypto++ cryptographic library [6]. The General Construction algorithm using AES, GCSA-AES, was implemented using C code developed by Barreto [2]. Test programs were compiled using Microsoft Visual C++ 6.0 and executed on a Dell Precision 340 computer with a 2.53 GHz Pentium IV processor. Only the compression functions of the hash algorithms are used in the implementation of the round functions in the ManTiCore ciphers (i.e., no byte-swapping or padding). Table 2 presents timing figures for the hash algorithms used as primitives for the ManTiCore ciphers and for several commonly used encryption algorithms.

The timing estimates presented in Section 2 suggest that MTC4-SHA-1 and MTC4-MD5 encryption should be approximately 6 and 8 times slower, respectively, than simply hashing the same message. Taking the authentication steps and extra *XOR* operations of the ManTiCore ciphers into account, these estimates are borne out by our results. Table 3 provides comparative figures of the ManTiCore algorithms and the general cipher-state authentication construction using AES against the typical usage of AES in CBC mode for encryption with HMAC authentication [12]. The RMTC4 algorithms are implemented using a pairwise independent permutation for Rounds 1 and 4. Specifically, we compute $F(K, m) = (k_1 m \pmod{p}) + k_2 \pmod{2^h}$, where k_1 and k_2 are derived from K 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.

Sha-zam [19] is a block cipher that is very similar in construction to our RMTC4 algorithm and, according to the authors, is capable of speeds equivalent to DES. With an optimized, assembly-language implementation, we would expect RMTC4-SHA-1 to have a similar speed. However, assuming CBC mode for encryption, Sha-zam and all of the algorithms in Table 2 must be executed in a block-serial manner, and they will not benefit from parallel processing. Future work will include optimizing the performance of all of our authenticated encryption algorithms.

Table 2. Performance comparison of hash and encryption algorithms on 10,000 byte messages using a 2.53 GHZ Pentium IV PC.

Algorithm	**Mbytes/Second**
SHA1	57
MD5	200
AES	88
DES	33
3DES(EDE)	13

Table 3. Performance comparison of the ManTiCore algorithms and the AES with cipher state authentication algorithm against other combined encryption/authentication algorithms on 10,000 byte messages using a 2.53 GHZ Pentium IV PC.

Algorithm	**Mbytes/Second**
MTC4-SHA1 Encryption and Authentication	10
MTC4-MD5 Encryption and Authentication	22
RMTC4-SHA1 Encryption and Authentication	15
RMTC4-MD5 Encryption and Authentication	28
GCSA-AES-AES	66
GCSA-AES-SHA1	65
GCSA-AES-MD5	66
AES-CBC-HMAC-SHA-1	33
AES-CBC-HMAC-MD5	58

5 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 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. 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 MTC 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 MTC ciphers. We also present a general construction for authentication from the internal state of a multiround block cipher. All of our algorithms provide authenticated encryption with little overhead for authentication, resistance against IV reuse, CBC mode qualities, and opportunities for parallelization.

Acknowledgment: The authors would like to thank Erik Anderson for many useful discussions and comments.

References

1. R. Anderson, E. Biham, "Two Practical and Provably Secure Block Ciphers: BEAR and LION," *Fast Software Encryption*, Lecture Notes in Computer Science, Springer-Verlag, 1996.
2. P. Barreto, "The Block Cipher Rijndael," http://www.esat.kuleuven.ac.be/~rijmen/rijndael/.
3. M. Bellare, A. Desai, E. Jokipii, P. Rogaway, "A Concrete Security Treatment of Symmetric Encryption," *In 38th Annual Symposium on Foundation of Computer Science (FOCS '97)*, IEEE, 394-403.
4. M. Bellare, R. Guerin, P. Rogaway, "XOR MACS: New Methods for Message Authentication using Finite Pseudorandom Functions," *Advances in Cryptology - CRYPTO 1995*, Lecture notes in Computer Science, vol. 963, D. Coppersmith, ed., Springer-Verlag, 1995.
5. M. Bellare, C. Namprempre, "Authenticated Encryption: Relations Among Notions and Analysis of the Generic Composition Paradigm," *ASIACRYPT 2000*, 531-545.
6. W. Dai, "Crypto++ Library," http://www.eskimo.com/weidai/cryptlib.html.
7. Department of Commerce/NIST, "Secure Hash Standard," FIPSPUB 180-1, April 17, 2001.
8. Department of Commerce/NIST, "Advanced Encryption Standard," FIPSPUB 197, November 26, 2001.
9. V. D. Gligor, P. Donescu, "Fast Encryption and Authentication: XCBC Encryption and XECB Authentication Modes," *Fast Software Encryption*, Lecture Notes in Computer Science, Springer-Verlag, 2001.
10. C. Jutla, "Encryption Modes with Almost Free Message Integrity," *Advances in Cryptology - EUROCRYPT 2001*, Lecture notes in Computer Science, vol. 2045, B. Pfitzmann, ed., Springer-Verlag, 2001.

11. L. Knudsen, "The Security of Feistel Ciphers with Six Rounds or Less," *Journal of Cryptology*, Volume 15 # 3, 2002.
12. H. Krawczyk, M. Bellare, R. Canetti, "HMAC: Keyed hashing for message authentication," Internet RFC 2104, February 1997.
13. 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.
14. M. Luby, C. Rackoff, "How to Construct Pseudorandom Permutations from Pseudorandom Functions," *SIAM Journal of Computing*, 17: # 2, 373-386, 1988.
15. S. Lucks, "Faster Luby-Rackoff Ciphers," *Proc. Fast Software Encryption*, LNCS, 1039, 189-203, 1996.
16. U. M. Maurer, "A Simplified and Generalized treatment of Luby-Rackoff Pseudorandom Permutation Generators," *Advances in Cryptology - EUROCRYPT 1992*, LNCS 658, 239-255, 1992.
17. M. Naor, O. Reingold, "On the Construction of Pseudo-Random Permutations: Luby-Rackoff revisited," *J. of Cryptology*, vol. 12, 29-66, 1999.
18. S. Patel, Z. Ramzan, G. S. Sundaram, "Towards Making Luby-Rackoff Ciphers Optimal and Practical," *Proc. Fast Software Encryption*, 171-185, 1999.
19. 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.
20. R. Rivest, "The MD5 message digest algorithm," IETF Network Working Group, RFC 1321, April 1992.
21. P. Rogaway, M. Bellare, J. Black, T. Krovetz, "OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption," *8th ACM Conference on Computer and Communications Security*, ACM Press, 2001.