Submission to NIST:
Cipher-State (CS)
Mode of Operation for AES

Sandia National Laboratories**
P.O. Box 5800 MS 0785
Albuquerque, NM 87185-0785
FAX: 505-845-7065

Submitter:

Richard C. Schroeppel
Phone: 505-844-9079
Email: rschroe@sandia.gov

Authors:

W. Erik Anderson
Phone: 505-284-9621
E-mail: weander@sandia.gov

Cheryl L. Beaver
Phone: 505-844-9547
E-mail: cbeaver@sandia.gov

Timothy J. Draelos
Phone: 505-844-8698
Email:tjdrael@sandia.gov

Richard C. Schroeppel
Phone: 505-844-9079
E-mail: rschroe@sandia.gov

Mark D. Torgerson
Phone: 505-284-5677
Email: mdtorge@sandia.gov

** 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.

1 Mode Specification

Cipher-State (CS) is a new mode of encryption, which uses information from the internal state of the cipher to provide the authentication very efficiently. This methodology has a number of benefits. The encryption has some of the valuable properties of 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 algorithm is very nearly that of the encryption process alone. Also, the authentication process remains resistant against some initialization vector (IV) reuse. We provide a general construction for using the internal state of any round-based block cipher as an authenticator and we give a concrete example of this general construction that uses the Advanced Encryption Standard (AES) [4] as the encryption primitive.

1.1 Encryption with Cipher-State Mode

CS mode is 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. 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 pre-authenticator, A. 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 also 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). Table 1 shows the least primitive polynomials for various degrees.

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

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

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 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-128, 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 plaintext. 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 other ciphers with a definite round structure, we propose using the midpoint as the tapping point.

The non-commutative combining operation used for the running pre-authenticator A is easy and inexpensive 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 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.

1.2 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 ciphertexts that agree in the same positions.

CS mode has the following properties.

1. Repeated plaintext blocks within the same message encrypt to equal values with negligible probability.
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 the identical position 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 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 [2] and OCB [6] are trivially broken with a few messages processed with the same IV.

Since security under nonce reuse is difficult to achieve, the typical solution is to simply insist that implementations never reuse nonces and thus pass the responsibility to implementors. However, nonce reuse is a significant practical concern. It is 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.

Offering a solution that addresses a pragmatic set of system-wide security issues, including security under nonce reuse, is the motivation of our designs. The hope is the construction of an efficient encryption mode with authentication 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.

In CS mode, 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. Although nonrepeated IVs are preferred, 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.

1.3 Security Considerations

One security concern with CS-AES is that since the computation of the authenticator reaches into the middle of the encryption process, 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 A. 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 of, 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 the 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 effectively impossible. (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. This is negligible for AES, which was specifically designed to withstand this sort of differential attack.

2 Summary of Properties

Security Function	Parallelizable encryption mode with inexpensive authentication
Error Propagation	none
Synchronization	Same IV used by sender and receiver
Parallelizability	Block parallelizable encryption/decryption and authentication generation/verification
Keying Material Requirements	One key
IV Requirements	Tolerant of some IV reuse
Memory Requirements	A few cipher blocks
Pre-processing Capability	R sequence can be precomputed
Message Length Requirements	Arbitrary length, must be padded to a multiple of the cipher block size
Ciphertext Expansion	Fixed size authenticator appended to message

3 Performance Estimates

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 number of block cipher invocations required by the CS encryption mode is $j+2$, where j is the number of blocks in the message. When a cryptographic hash algorithm is used for

the final authentication, $j + 1$ block cipher invocations are required. CS mode is block-parallelizable, which will make implementations with a parallelization capability faster with no loss of security.

To test the performance of our algorithms and measure the overhead relative to the underlying cryptographic primitives, 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 [1]. Test programs were compiled using Microsoft Visual C++ 7.1 and executed on a Dell Precision 340 computer with a 2.53 GHz Pentium IV processor. 1024-byte messages were used during testing. Table 2 provides comparative figures of the the CS-AES mode against the AES cryptographic primitive and the typical usage of AES in CBC mode for encryption with HMAC authentication [5] using SHA-1 [3]. A 128-bit key was used for AES.

Table 2. Performance comparison of the CS-AES algorithm against AES alone and AES-CBC with HMAC-SHA-1 authentication on 1024-byte messages using a 2.53 GHZ Pentium IV PC.

Algorithm	**Mbytes/Second**
AES	69
CS-AES-AES	61
CS-AES-SHA-1	61
AES-CBC-HMAC-SHA-1	32

4 Test Vectors

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`
$\boldsymbol{m_1 \oplus R_1}$ **:**	`FDFC0BA14D46C5AD04076094C309DA55`
$\boldsymbol{t_1}$ **:**	`C31FDB743AA199CB78AA156AED162EB9`
$\boldsymbol{A_1}$ **:**	`C31FDB743AA199CB78AA156AED162EB9`
$\boldsymbol{AES(K, m_1 \oplus R_1)}$ **:**	`FEE2017432993F8D81E1251E9BD6125E`
$\boldsymbol{c_1}$ **:**	`030F28E63B8A9C570D7FEF31940226F4`
$\boldsymbol{R_2}$ **:**	`FBDA5324122747B5193D945E1FA869D3`
$\boldsymbol{A_1 \oplus R_2}$ **:**	`38C588502886DE7E61978134F2BE476A`
$\boldsymbol{AES(K, A_1 \oplus R_2)}$ **:**	`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
	$\vdots$
$\boldsymbol{m_{1,000,000}}$:	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

5 Intellectual Property Statements/Agreements/Disclosures

The authors dedicate this specification and release any intellectual property rights to CS mode to the public domain. The authors are not aware of any patent or patent application that covers CS mode. CS mode is a simple combination of well known pre-whitening and post-whitening techniques with the use of the internal cipher state for authentication and the necessary support to make this safe.

References

1. P. Barreto, "The Block Cipher Rijndael," http://www.esat.kuleuven.ac.be/~rijmen/rijndael/.
2. 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.
3. Department of Commerce/NIST, "Secure Hash Standard," FIPSPUB 180-1, April 17, 2001.
4. Department of Commerce/NIST, "Advanced Encryption Standard," FIPSPUB 197, November 26, 2001.
5. H. Krawczyk, M. Bellare, R. Canetti, "HMAC: Keyed hashing for message authentication," Internet RFC 2104, February 1997.

6. 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.
7. W. Dai, "Crypto++ Library," http://www.eskimo.com/weidai/ cryptlib.html.