Comments to NIST concerning AES Modes of Operations:

CTR-Mode Encryption

Helger Lipmaa
Helsinki University of Technology (Finland) and
University of Tartu (Estonia)
helger@tml.hut.fi
http://www.tml.hut.fi/~helger

Phillip Rogaway
University of California at Davis (USA) and
Chiang Mai University (Thailand)
rogaway@cs.ucdavis.edu
http://www.cs.ucdavis.edu/~rogaway

David Wagner
University of California Berkeley (USA)
daw@cs.berkeley.edu
http://www.cs.berkeley.edu/~wagner

Abstract

Counter-mode encryption ("CTR mode") was introduced by Diffie and Hellman already in 1979 [5] and is already standardized by, for example, [1, Section 6.4]. It is indeed one of the best known modes that are not standardized in [10]. We suggest that NIST, in standardizing AES modes of operation, should include CTR-mode encryption as one possibility for the next reasons. First, CTR mode has significant efficiency advantages over the standard encryption modes without weakening the security. In particular its tight security has been proven. Second, most of the perceived disadvantages of CTR mode are not valid criticisms, but rather caused by the lack of knowledge.

1 Review of Counter-Mode Encryption

Notation. Let $E_K(X)$ denote the encipherment of an n-bit block X using key K and a block cipher E. For concreteness we assume that $E = \text{AES}$, so $n = 128$. If X is a nonempty string and i is a nonnegative integer, then $X + i$ denotes the $|X|$-bit string that one gets by regarding X as a nonnegative number (written in binary, most significant bit first), adding i to this number, taking the result modulo $2^{|X|}$, and converting this number back into an $|X|$-bit string. This is the customary semantics for computer addition.

Operation. To encrypt using CTR-mode encryption, one starts with a plaintext M (an arbitrary bit string), a key K, and a counter ctr, where ctr is an n-bit string. Let C be the XOR (excusive-or) of M and the first $|M|$ bits of the pad $E_K(\mathit{ctr}) \parallel E_K(\mathit{ctr}+1) \parallel E_K(\mathit{ctr}+2)\cdots$. The ciphertext is (ctr, C), or, more generally, C together with something adequate to recover ctr. To decrypt ciphertext (ctr, C) compute the plaintext M as the XOR of C and the first $|C|$ bits of the pad $E_K(\mathit{ctr}) \parallel E_K(\mathit{ctr}+1) \parallel E_K(\mathit{ctr}+2)\cdots$. Therefore, decryption is the same as encryption with M and C interchanged (see Figure 1). Often we refer to C itself, rather than (ctr, C), as the ciphertext.

Usage scenarios. In the recommended usage scenario, the party encrypting maintains an integer counter, *nonce*, initially 0, and produces the string *ctr* as the 128-bit string which encodes the number $\mathit{nonce} \cdot 2^{64}$. (In other words, *nonce* is regarded as a 64-bit binary number, and *ctr* is constructed by appending to this number 64 zero-bits.) The number *nonce* is incremented following each encryption. Typically, one transmits C along with a string which encodes *nonce*.

A well-designed standard for CTR mode should not be overly prescriptive about how *ctr* is formed or what beyond C is explicitly communicated between sender and receiver. To illustrate some possibilities: (1) the value *ctr* is derived from a nonce *nonce* by the method just described, and the ciphertext specifies both *nonce* and C; (2) the same, except that no *nonce*-value is explicitly transmitted to the receiver because the sender and the receiver maintain state and

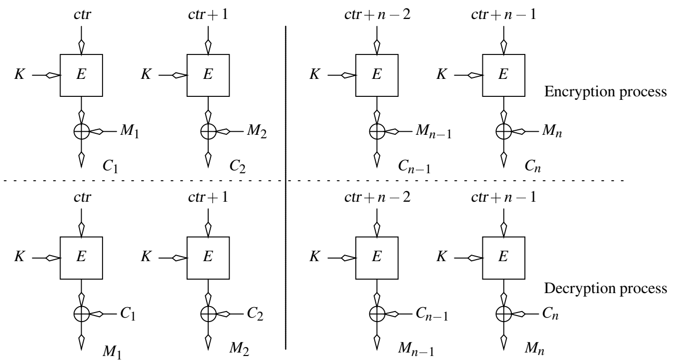

Figure 1: Encryption and decryption process in counter mode.

communicate over a reliable channel; (3) the same, except that *nonce* starts at a random value in $[0..2^{64-1}]$ instead of starting at 0; (4) *ctr* is a random 128-bit string, selected afresh with each message sent; and (5) *ctr* is determined implicitly by other protocol elements, such as an accompanying sequence number (e.g., in the context of IPSec).

The above scenarios make clear that no single method of producing *ctr* is the best in all situations. The standard should clearly describe a small number of recommended ways to form *ctr*. But it ultimately the user's responsibility to ensure that it is impossible, or highly improbable, that a $ctr + i$ value is ever reused with the same key K.

2 Advantages of CTR Mode

Software efficiency. Modern processors support some or all of the following architectural features: aggressive pipelining, multiple instruction dispatch per clock cycle, a large number of registers, and SIMD instructions. By eliminating the computational dependency between C_i and C_j, CTR-mode encryption enables effective utilization of the above features. For many ciphers, a well-optimized implementation of CTR-mode encryption on a processor such as an Pentium III, Itanium, Alpha, or a Motorola AltiVec, may be substantially faster (even more than four times [8]) than a well-optimized implementation of CBC-mode encryption. This is greater than the gain obtained from switching from the slowest to the fastest AES finalist on most platforms [7].

Hardware efficiency. Modes such as CBC encryption are limited in their hardware speed by the maximal rate at which the underlying block cipher can be computed. This is because one must complete the computation of ciphertext C_i before one can begin to compute C_{i+1}. Thus the maximal throughput, in hardware, will be about the reciprocal of the latency for E. In contrast, CTR model is fully parallelizable: one can be computing blocks $C_1, C_2, \ldots$ all at the same time, limited only by the amount of hardware that one throws at the problem. This has been shown to result in $30 \ldots 100$ times speedups for four of the AES finalists [6].

Preprocessing. Because the cryptographic work in enciphering a message M is independent of M, preprocessing can be used, in some environments, to increase speed. That is, one can compute the pad in "spare cycles," even before one knows the plaintext M. When M is known, it is XORed with the already-computed pad. The latter can be done with throughput 10–25 Gbit/s on a contemporary processor.

Random-access. The ith ciphertext block, C_i, can be encrypted in a random-access fashion. This is important in applications like hard-disk encryption, where bulk data needs to be encrypted quickly. When using CBC mode, properly encrypting the ith block requires one to first encrypt the $i-1$ prior blocks.

Provable security. The above efficiency characteristics are not obtained at the expense of security. In fact, the "standard" cryptographic assumption about a block cipher's security—that it is a "pseudorandom permutation" [9, 4]—is enough to prove the security of CTR-mode encryption. See [2], which shows that the concrete security bounds one gets for CTR-mode encryption, using a block cipher, are no worse than what one gets for CBC encryption. (Indeed there are approaches to get *better* security bounds with CTR-mode encryption than with CBC mode, though these do not *directly* use the block cipher E). See [4, 3].) The security of CTR mode is well-analyzed and well-understood.

Simplicity. With CTR mode, both encryption and decryption depend only on E—neither depends on the inverse map, $D = E^{-1}$. So D need not be implemented. This matters most when the inverse direction of the block cipher, D, is substantially different from the forward direction, E, as is the case for all the AES finalists. Moreover, the decryption key scheduling need not be implemented. Together, this makes implementations simpler, and may yield a significant throughput increase in hardware.

Messages of arbitrary bit-length. Unlike other common modes of operation, handling messages of *arbitrary bit-length* is made trivial. No bits are wasted in doing this—the ciphertext C is of the same length as the plaintext M.

See [11] for a slightly different view of the advantages of counter-mode encryption.

3 Perceived Disadvantages of CTR mode

No integrity. CTR-mode encryption provides no message integrity. However, it is not normally considered the purpose of encryption to provide for message integrity, and other common modes, like CBC, likewise fail to provide any meaningful message integrity. If message integrity is desired, the usual approach is to accompany the ciphertext by a MAC (message authentication code). Another concern is that the CTR mode has a "stronger" requirement for an accompanying MAC than does CBC mode. In fact, a CBC-encrypted ciphertext must be accompanied by a MAC whenever integrity is desired; it is wrong to think of the mode as providing any meaningful integrity.

Error propagation. If a bit-flip error occurs in some block of ciphertext, after decryption, the error is localized to the corresponding bit of the corresponding block of plaintext. This is neither good nor bad, but just irrelevant—if detection or correction of errors is desired, this should be accomplished at a different architectural level, using appropriate tools.

Stateful encryption. In the usual way to use CTR mode, the sender is stateful. But in the proper use of a mode like CBC, the sender is also stateful. That is because the IV is either maintained from the last block enciphered, or it is a pseudorandom value, and the usual implementation of a pseudorandom value requires the maintenance of state. In contexts where state can not be maintained and a random value can be obtained, CTR mode, like CBC mode, need not maintain state. In short, the situation with respect to statefulness is essentially the same as with CBC mode.

Sensitivity to usage errors. It is crucial in CTR-mode encryption that a counter-value not be reused (both the *ctr*-value that logically accompanies C, and also all "internal" counters that mask plaintext blocks). What if a user inappropriately reuses a counter? Then all security is lost. But catastrophic errors in usage can be made with any mode of operation. As already mentioned, standard should make clear what is required of the counter-values in order to be secure and recommend a small number of secure usage scenarios.

Interaction with weak ciphers. Successive blocks ctr and $ctr+1$ usually have small Hamming difference. This has lead to the concern that an attacker can obtain many plaintext pairs with a known small plaintext difference, which would facilitate the differential cryptanalysis. However, this concern is only valid if the underlying cipher is differentially weak. It is not the responsibility of a mode of operation to try to compensate (likely without success) for weaknesses in the underlying block cipher; this concern should be addressed when designing the block cipher. Another concern we have heard is that since using a counter for the IV in CBC mode is a bad idea (see, e.g., [12]), maybe CTR mode itself is suspect. This is just sloppy thinking; the problem with using a counter IV is specific to CBC mode, it has nothing to do with CTR mode.

Acknowledgments

We would like to thank Kaisa Nyberg for thoughtful comments.

References

[1] ATM FORUM. ATM Security Specification Version 1.0, af-sec-0100.001. Available from http://www.atmforum.com/atmforum/specs/approved.html.

[2] MIHIR BELLARE, ANAND DESAI, ERON JOKIPII and PHILLIP ROGAWAY. A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES Modes of Operation. Proceedings of 38th Annual Symposium on Foundations of Computer Science, 1997 (FOCS '97).

[3] MIHIR BELLARE and RUSSELL IMPAGLIAZZO. A Tool for Obtaining Tighter Security Analyses of Pseudorandom Function Based Constructions, with Applications to PRP→PRF Conversion. Proceedings of 40th Annual Symposium on Foundations of Computer Science, 1999. (FOCS '99).

[4] MIHIR BELLARE, TED KROVETZ and PHILLIP ROGAWAY. Luby-Rackoff Backwards: Increasing Security by Making Block Ciphers Non-invertible. *Advances in Cryptology—Eurocrypt '98*, Lecture Notes in Computer Science, vol. 1402, Springer-Verlag, Kaisa Nyberg, editor, pp. 266–280, 1998.

[5] WHITFIELD DIFFIE and MARTIN HELLMAN. Privacy and Authentication: An Introduction to Cryptography. Proceedings of the IEEE, 67 (1979), pp. 397–427.

[6] KRIS GAJ and PAWEL CHODOWIEC. Fast Implementation and Fair Comparison of the Final Candidates for Advanced Encryption Standard using Field Programmable Gate Arrays. Technical report, http://ece.gmu.edu/crypto/publications.htm.

[7] HELGER LIPMAA. AES Candidates: A Survey of Implementations. Constantly updated on-line table, available from `http://www.tml.hut.fi/~helger/aes/`.

[8] HELGER LIPMAA. IDEA: A Cipher for Multimedia Architectures? In Stafford Tavares and Henk Meijer, editors, Selected Areas in Cryptography '98, volume 1556 of Lecture Notes in Computer Science, pages 248–263, Kingston, Canada, 17–18 August 1998. Springer-Verlag.

[9] MICHAEL LUBY and CHARLIE RACKOFF. How to Construct Pseudorandom Permutations from Pseudorandom Functions. *SIAM J. Comput.*, vol. 17, No. 2, April 1988.

[10] NATIONAL BUREAU OF STANDARDS (USA). DES Modes of Operation. Federal Information Processing Standard (FIPS) Publication 81, December 1980, http://www.itl.nist.gov/div897/pubs/fip81.htm.

[11] RONALD L. RIVEST, M. J. B. ROBSHAW and Y. L. YIN. The Case for RC6 as the AES. An AES second round comment. `http://csrc.nist.gov/encryption/aes/round2/pubcmnts.htm`.

[12] PHILLIP ROGAWAY. Problems with Proposed IP Cryptography. April 3, 1995. `http://www.cs.ucdavis.edu/~rogaway/papers/draft-rogaway-ipsec-comments-00.txt`.