Crypto Corner
Editors: Peter Gutmann, pgut001@cs.auckland.ac.nz
David Naccache, david.naccache@ens.fr
Charles C. Palmer, ccpalmer@us.ibm.com

XTS: A Mode of AES for Encrypting Hard Disks

Luther Martin
Voltage Security

The recent proliferation of data security and privacy regulations has led to increased interest in encrypting data both in motion and at rest. Although there are successful protocols for encrypting data in motion, a way to encrypt data at rest has proven more elusive. This is due partly to the constraints that storage technologies can impose on encryption technologies. Hard disks are a good example of this.

The IEEE Security in Storage Working Group (SISWG) has developed the XTS mode of the Advanced Encryption Standard (AES) that the IEEE 1619-2007 standard defines.[1] (XTS stands for XEX-based tweaked codebook mode with ciphertext stealing.) This mode works within the constraints of hard disks while keeping the security that the AES algorithm provides.

The US National Institute of Standards and Technology (NIST) has approved XTS for US government use.[2] Approval of a new AES mode happens rarely, so this event is interesting and newsworthy. Here, I describe what motivates XTS's construction, how it works, and what level of security it provides.

The Motivation for XTS

Hard disks are partitioned into circular paths called *tracks*—physical divisions of data that are determined by the data's location on the disk. Tracks are in turn partitioned into fixed-sized logical *sectors*, which can be individually read from or written to a disk. Sectors are the smallest accessible subdivision of a track, typically comprising 512 bytes. A sector might be subdivided into logical *blocks*, which are the same size as the block of data encrypted by a block cipher. The number of bytes in a sector might or might not be an integer multiple of the block size.

Because a sector's bytes are all dedicated to storage, no additional space is available for other information. So, inputs to an encryption algorithm for data on hard disks should include only the data itself, the key used in the encryption algorithm, and available metadata such as the sector number and the block number in that sector. XTS does exactly this. It also protects better against ciphertext manipulation and cut-and-paste attacks than other AES modes working with the same set of constraints.

XTS is based on Phil Rogaway's XEX (Xor-Encrypt-Xor) construction[3] and uses ciphertext stealing[4] to handle sectors not containing a number of bytes equal to an integer multiple of the AES block size.

Tweakable Block Ciphers

Suppose we have a block cipher E that operates on a message M and a key K to produce a ciphertext C. We write this cipher's operation as $C = E_K(M)$. A *tweakable* block cipher uses E to operate on M, K, and tweak T to produce C. We write this cipher's operation as $C = E_K(T, M)$. The tweak operates much like an initialization vector but has different security properties: an initialization vector must be random, whereas a tweak doesn't have to be.

A tweak aims to provide variability of the ciphertext, whereas the key provides security against an adversary recovering the plaintext. It's not necessary to keep a tweak secret, and a tweakable block cipher must remain secure even if an adversary can control the tweak inputs into an encryption operation.

Rogaway's XEX

Suppose that N and α_1 through α_k are elements of GF(2^n)* and that i_1 through i_k are integers. (GF stands for Galois field.) Rogaway showed that if E is a secure block cipher, then $E_K(N, i_1, \ldots, i_k, M) = E_K(M \oplus \Delta) \oplus \Delta$, where

$$\Delta = \alpha_1^{i_1} \ldots \alpha_k^{i_k} E_k(N),$$

and

$$\alpha_1^{i_1} \ldots \alpha_k^{i_k} \neq 1$$

is also a secure block cipher that uses the $k + 1$ tweaks N and i_1 through i_k. Figure 1 shows encryption using this construction.

XTS

The SISWG modified XEX by limiting the number of tweaks to

two and using two keys instead of one. The two tweaks correspond to the sector and block number where data is stored. The use of two keys isn't a criticism of XEX. Rather, it results from the influence of the technology's commercial users, who believe that two keys are better than one.

XTS calculates a ciphertext from M and two keys K_1 and K_2 as

$$E_{K_1,K_2}(i,j,M) = E_{K_1}(M \oplus \Delta) \oplus \Delta,$$

where $\Delta = \alpha^i E_{K_2}(j)$. It sets the value of j to the sector number being encrypted and sets the value of i to the block number in this sector. The constant α is either

- the primitive element GF(2^{128}) represented by the value 2 if the elements of GF(2^{128}) are represented as bit strings, or
- the polynomial x if the elements of GF(2^{128}) are represented by polynomials.

Figure 2 shows encryption using this construction.

Ciphertext Stealing

This approach cleverly combines the last two blocks of ciphertext.

Suppose we want to use E and K to encrypt m blocks of messages M_1 through M_m to produce ciphertext blocks C_1 through C_m, where M_1 through M_{m-1} are the same size as the cipher block but M_m is shorter, having only s bits instead of the full n. To encrypt this sequence of messages using ciphertext stealing, we initially encrypt the first $m - 2$ blocks normally to get C_1 through C_{m-2}.

Let $B[1, \ldots, l]$ denote the lowest l bits of message block B. To process the last two blocks, we first create an intermediate value $C = E_K(M_{m-1})$ and parse this value into $C = C_m[1, \ldots, s] \| C'$. We then calculate the value of C_{m-1} as $C_{m-1} = E_K(M_m[1, \ldots, s] \| C')$.

Figure 3 shows encryption using ciphertext stealing.

XTS seems to have been fairly successful so far. As of April 2009, 14 vendors had supported it.[5]

The SISWG will soon start amending IEEE 1619-2007 to include the option of using only one key. This will simplify key management slightly at no cost in cryptographic security. Future versions of the standard might also let you use additional metadata as tweaks; XEX's inherent flexibility makes this easy to do. □

References

1. *IEEE Std. 1619-2007, Cryptographic Protection of Data on Block-Oriented Storage Devices*, IEEE, 2008.
2. M. Dworkin, *Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices*, NIST Special Publication 800-38E, US Nat'l Inst. of Standards and Technology, 2010; http://csrc.nist.gov/publications/nistpubs/800-38E/nist-sp-800-38E.pdf.
3. P. Rogaway, "Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC," *Proc. Asiacrypt 2004*, Springer, 2004, pp. 16–31.
4. C. Meyer and S. Matyas, *Cryptography: A New Dimension in Computer Data Security*, John Wiley & Sons, 1982.
5. M.V. Ball, *Follow-Up on NIST's Consideration of XTS-AES*, IEEE Security in Storage Working Group, 2009; http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/XTS/follow-up_XTS_comments-Ball.pdf.

***Luther Martin** is the chief security architect at Voltage Security. Contact him at martin@voltage.com.*

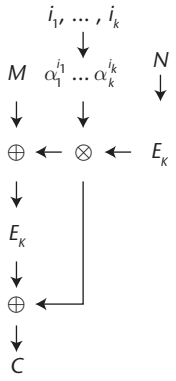

Figure 1. Encryption using Phil Rogaway's XEX (Xor-Encrypt-Xor) construction. XEX forms the basis for the XTS mode of the Advanced Encryption Standard (AES). (XTS stands for XEX-based tweaked codebook mode with ciphertext stealing.)

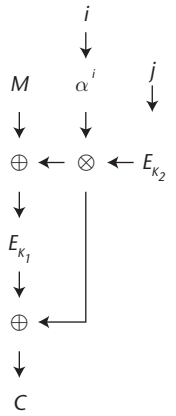

Figure 2. Encryption using the XTS construction. XTS modifies XEX by using only two tweaks and using two keys instead of one.

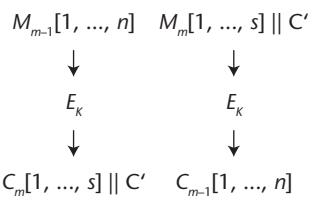

Figure 3. Encryption using ciphertext stealing. This approach handles sectors not containing a number of bytes equal to an integer multiple of the AES block size.