Public Comments on the XTS-AES Mode

On June 5, 2008, NIST initiated a period of public comment, ending September 3, 2008, on a proposal to approve the XTS-AES mode of operation by reference to IEEE Stnd. 1619-2007. The *Request for Public Comment on XTS* is available at http://csrc.nist.gov/ groups/ST/documents/Request-for-Public-Comment-on_XTS.pdf.

In response to this request, NIST received the public comments below within email messages. NIST also received comment documents from: 1) Moses Liskov and Kazuhiko Minematsu, 2) Seagate Technology, and 3) Matthew Ball, Chair of the IEEE Security in Storage Working Group (P1619) who submitted the XTS-AES mode to NIST. Links to these three documents are available at the comments page, http://csrc.nist.gov /groups/ST/toolkit/BCM/comments.html.

Boaz Shahar

Hello,

My name is Boaz Shahar, engineer in Entropic Communications (Israel).

Recently we developed a security accelerator for one of Entropic Communications product. The NIST CSRC site was a great and very useful resource of information for us: Especially the very clear textbook-level explanations, the easy and comfortable access to NIST standards, the examples and test vectors. I hope that after adopting the IEEE XTS mode NIST will continue with this tradition, and will offer the new XTS mode as an opened standard from NIST resources with the same level as all other NIST standards and recommendations.

Thanks, and regards,

Boaz

This comments is in regards to the XTS algorithm itself:

The XTS algorithm is using a multiplication by ALFA of GF(2^{128}) on each round. This multiplication over GF(2^{128}) is defined in little endian notation (See section 5.2 in the proposed standard). I see two difficulties with this:

1. This impose some extra complexity on implementation, as it is much easier to implement multiplication over GF with Big Endian notation, where the multiplication reduces to simple left shift and xor, in some cases; And:

2. Usually NIST is using Big endian. For instance, SP800-38B about CMAC mode for authentication which is using exactly the same operation is using Big Endian notation.

David Clunie

Hello

I am writing with some responses to the request for public comment on XTS inclusion as an encryption mode of operation of the Advanced Encryption Standard (AES) block cipher.

I am a radiologist and informatics specialist involved in the standardization of digital image transfer for medical imaging devices, formerly the industry co-chair of the DICOM Standards Committee, and currently editor of the DICOM standard and chairman of the DICOM WG 5 on Interchange Media.

These comments are my own and do not represent the position of the DICOM Standards Committee, nor any of its working groups, nor of my employer, RadPharm, Inc.

Comments were requested on the following topics:

- The XTS algorithm itself;

No comment.

- The depth of support in the storage industry for which it was designed;

For XTS to be useful for interchange (removable) media protection, standardization of XTS or a similar mechanism as an encryption mode of operation of AES is desirable, but insufficient; there would also need to be an appropriate means of exchange of keys in a standard and inter-operable manner defined. It is understood that this is outside the scope of the current request, but some indication of whether or not there is such an effort in progress would be helpful.

Or to put this another way, it may be premature to standardize the mode of operation until it is well established that there will be support in the operating system industry for its usage in the context of removable media device drivers (for removable optical media like CD and DVD as well as USB media (flash drives).

- The appeal of XTS for wider applications;

In digital medical image exchange (and digital medical record exchange in general), though online transactions are envisaged, the reality of the state of the art is that most images are exchanged on removable media due to their bulk (hence slow speed of network transfer) and the lack of an authentication infrastructure between loosely coupled providers and users. Current exchange is generally unprotected (no encryption at all), since the threat is presumed to be low and any barrier to reception risks patient safety. However, it is anticipated that encryption may be required in the long term, and it is desirable that a standard, inter-operable, cross-platform storage encryption infrastructure be available in consumer operating systems so that it can be leveraged by medical applications.

- The proposal for the approved specification to be available only by purchase from IEEE;

This proposal is regarded as unacceptable.

FIPS standards should be available to the public for download without charge.

Any charge, no matter how apparently insignificant, limits the accessibility, opportunity for scientific review and likelihood of adoption of the standard.

The IEEE and the participants in P1619 presumably expect significant commercial benefits to flow from the inclusion of XTS as an FIPS, and should be satisfied in that regard rather than expect a flow of revenue from purchase of the P1619 XTS standard itself. Alternatively, there are other means by which IEEE may be remunerated by the federal government that do not involve purchase fees for the standard.

- Concerns of intellectual property rights.

The ambiguity that is stated in the request for comments with respect to nCipher seems unsatisfactory, and should be further investigated and resolved satisfactorily.

The reason being that there may be alternative implementations, or modifications to XTS, that could achieve the same objective as XTS without invoking any likelihood that adoption would be hampered by such existing intellectual property restrictions.

Only an unquestionably unencumbered proposal should be adopted as standard mode of operation.

David
--

Rich Shroeppel

In reference to the proposed encryption mode XTS-AES: It seems inappropriate for the government to be lending its imprimatur to a standard that will cost $105 per copy, and be sold by a private organization. Standards should be available for free for the asking, from a government website. Of course the IEEE is free to do whatever they want, but NIST shouldn't be endorsing it. Especially for an encryption standard, where wide and perpetual public review is needed to assure sufficient security scrutiny. Moreover, if NIST is going to include an external reference in a standard, they should include a suitable hash code to make sure that IEEE doesn't make alterations in the standard.

Rich Schroeppel

Phillip Rogaway

Dear Colleagues,

This email is in response to Morris Dworkin's 8 Jun 08 email to EncryptionModes@nist.gov requesting comments on XTS, SIV, and FFSEM. Here I will briefly comment on XTS.

Comments on XTS

XTS is the two-key version of my XEX construction (Asiacrypt 2004), but operating in ECB mode and with ciphertext stealing for any short final block. While I have no serious objection to NIST approving XTS by reference, I would like to make a couple of points.

First, it is unfortunate that there is nowhere described a cryptographic definition for what security property XTS is supposed to deliver. When the data unit (sector) is a multiple of 128 bits, each 128-bit block within the data unit should be separately enciphered as though by independent, uniformly random permutations. That part is clear. But what security property does one expect for partial final blocks? One might hope, for example, that the final 128+b bits (b<128) would likewise be enciphered as if by a strong PRP. That would seem to be the cleanest natural notion, and it's not too hard to achieve. But XLS does not achieve such an aim (because, for example, C_m does not depend on P_m). One is left to wonder if ciphertext stealing actually works to buy you any strong security property for the final 128+b bits.

Second, I would like to express the opinion that the nominally "correct" solution for (length-preserving) enciphering of disk sectors and the like is to apply a tweakable, strong PRP (aka wide-blocksize encryption) to the (entire) data unit. That notion is strong, well-studied, easy to understand, and readily achievable. There are now some 15+ proposed schemes in the literature for solving this problem. If NIST approves the "lite" enciphering mode that is XTS, this should not be understood to diminish the utility of standardizing a (wide-blocksize) strong PRP. In the end, because of its much weaker security properties, I expect that XTS is an appropriate mechanism choice only in the case that one simply cannot afford the computation or latency associated to computing a strong PRP.

Kind regards,
Phil Rogaway
University of California, Davis

Vijay Bharadwaj and Neils Ferguson

We would like to submit the following response to your request for public comments on XTS-AES. Thank you for the opportunity to provide feedback. In summary, we believe that XTS is not a good choice for standardization, for the reasons given below.

The proposal, and the XTS algorithm itself

Unclear Security Goals

In our opinion, one serious shortcoming of the proposal is that it does not contain a clear statement of what application-level security goals XTS aims to achieve. This makes it difficult to analyze the proposal, or to determine how widely applicable XTS may be. Appendix D contains some hints, and the XEX paper referred to contains some low-level goals. However, it is extremely difficult for a security practitioner or system implementer to understand what assurances can be provided with this mode. This differs from other recent standards such as SP800-38D, which contain clear explanations of this sort.

Temporal effects

The proposal appears to miss the effect of temporal effects on the security of XTS. It is possible for an attacker to observe a disk for a period of time and thereby gain a significant advantage in cryptanalysis. For instance, on a disk with 4 KB blocks where each block is a data unit, an attacker who observes approximately 4000 writes to a given block will have obtained access to 220 cipher blocks with the same tweak and key. Similarly, an attacker who steals a 500 GB encrypted disk may get access to about 1 TB of ciphertext if they were also able to recover the previous contents of each sector. Thus, it seems that the limits for key reuse given in Section 5 and Appendix D can be reached fairly easily in practice. We believe that the proposal lacks the margin of safety that would be expected of a mode which is to be used for 20-30 years.

An attack on large data units

The attack in D.4.2 can be extended. The attacker finds two positions i and j such that P_i \xor C_i = P_j \xor C_j. Let us assume they are in the same data unit. With reasonable probability this is because PP_i = PP_j and CC_i = CC_j. This allows the attacker to compute P_i \xor P_j = PP_i \xor T_i \xor T_j \xor PP_j = T_i \xor T_j = T * \alpha^i \xor T * \alpha^j = T * (\alpha^i \xor \alpha^j) and a simple finite-field division recovers the master tweak of the data unit T. This in turn leads to a knowledge of all tweaks in the data unit. This makes the ciphertext manipulation attacks much easier. In particular, it allows the attacker to choose the value for a subset of the bytes in each plaintext block being manipulated.

This shows that large data units significantly weaken the system. The standard should not allow data units larger than the recommended 220 blocks.

Ciphertext manipulation attacks

AES in XTS mode works with 16-byte blocks, and this allows for very fine-grained ciphertext manipulation attacks. We believe this is a significant problem in practice.

As discussed in section D.2 and D.3 any transparent storage encryption scheme allows ciphertext manipulation attacks. In general, the attacker can perform a number of manipulations to the ciphertext in order to influence the decrypted plaintext, but is limited by the block size S. In particular, the attacker can randomize the plaintext of any block by changing the ciphertext. The attacker's objective is to do this in such a way that some part of the plaintext is changed to a value of his choosing. We look at this in two specific contexts: code modification and data modification.

Code modification

In a code modification attack the attacker randomizes a block of code and tries to corrupt the code in such a way as to introduce a security hole in the system whilst keeping the system functional. Possibilities for the attacker include corrupting one of the access control functions in the code such that it omits certain checks, and corrupting an input validation routine such that it does not clean up untrusted input. Such a function might look like this:

```
C prototype:
Bool IsOperationAllowed( ... );

Assembler code:
IsOperationAllowed:
        <function prolog>  (pushes, setting up the stack frame, etc.)
        ...
        Mov     eax,[...]          // register eax is used during the computations
        ...
        <set eax to return value>
        <function epilog> (pops, revert stack frame, return)
```

The attacker chooses a storage encryption block of bytes in the function implementation with the following properties:

- When the CPU executes the first instruction in the block the eax register is nonzero (with high probability)

- The block does not contain the epilog code or other code that the system uses.

-

The attacker now replaces the first few bytes of the ciphertext block and hopes that the randomized plaintext decrypts to a value such that the first instruction executed in the block is a jump instruction to the epilog code. On the x86 a relative jump instruction is 2 bytes long and can jump up to 128 bytes forward. The probability of getting the right plaintext is one in 2^{16}; high enough for a practical attack.

The small block size of XTS-AES makes this attack rather easy. A larger block size makes it significantly harder. First of all, larger blocks have fewer starting points, so it is harder to find a block inside this function where the eax register is nonzero when the first instruction in the block is exectuted. Second, the larger block forces the attacker to randomize far more code. If the block size is large enough, then the block extends far beyond the jump range of a 2-byte jump instruction so the attack ends up jumping to some randomized plaintext. Thus, with wider blocks an attacker is much more likely to cause a crash rather than a security compromise.

Data modification

A modern operating system has thousands of settings that are important for the security of the system. Each of these settings is an attack target. The attacker tries to find a block of ciphertext that, when randomized, has a reasonable chance of changing one such setting to an insecure value. To change a value in most data formats the 'header' has to be left unchanged (as the software will perform some consistency or sanity checks on the header) but some of the contents has to be changed. This requires that there is a block boundary between the header and the contents. Again, a larger block size significantly increases the security in two ways. There are fewer block boundaries that can be exploited, and randomizing a block damages more of the overall data storage and significantly increases the chance of triggering a software crash/failure, rather than creating a security hole.

Data modification can also be applied to data storage applications like a database. A small block size allows targeted manipulations of just a few values in a single row of the database; a large block size severely restricts the attacker and greatly increases the chances of damaging other (necessary) parts of the data structure.

The proposal for the approved specification to be available only by purchase from IEEE

We believe that it is highly undesirable to standardize an algorithm whose specification, unlike those of other FIPS approved algorithms, is not freely available.

Concerns of intellectual property rights

As stated in the call for comments, the current situation of IP rights with respect to XTS-AES is unclear. We believe that standardizing an algorithm which is so encumbered is undesirable, and that IP issues could inhibit adoption of such a standard.

Niels Ferguson and Vijay Bharadwaj
Microsoft Corporation