NOTE: This is a draft for review by the IEEE Security in Storage Working Group

To: EncryptionModes@nist.gov
From: Matthew V. Ball, Sun Microsystems, Chair of IEEE Security in Storage Working Group (P1619)
Subject: Follow-up to NIST's Consideration of XTS-AES as standardized by IEEE Std 1619-2007
Date: ~~Feb 8~~March 28, 2009 (Draft 2)

Table of Contents

Introduction

This is a follow-up to comments previously submitted to NIST concerning the adoption of the XTS-AES mode of operation under FIPS 140 (currently at 140-2). This response is organized in the categories recommended by NIST in the *Request for Public Comments on XTS* (see [Request]):

- The XTS algorithm itself;
- The depth of support in the storage industry for which it was designed;
- The appeal of XTS for wider applications;
- The proposal for the approved specification to be available only by purchase from IEEE;
- Concerns of intellectual property rights.

The following sections are presented in the form of comment/response pairs. Some comments were omitted if they did not appear to need a response.

The XTS algorithm itself

Comments From Moses Liskov and Kazuhiko Minematsu

See [Liskov] for the full comments from Liskov and Minematsu. For brevity, I've only included the introduction here.

> *Overall, we believe that the XTS-AES algorithm, closely based on Rogaway's XEX mode [5] plus ciphertext stealing, is a good choice for the purpose of block-oriented data storage encryption, and the use of an algorithm of this type is well supported by research publications. We have two main criticisms of the publication. First, while XEX uses one key, the proposed XTS algorithm uses two keys; Key1 is used to encipher the whitened plaintext, while Key2 is used to compute the pre- and post-whitening values. We feel that only one key should be used, to serve both purposes. Second, the draft incompletely analyzes the security of XTS-AES; it needs correction and expansion in a couple of areas.*

Liskov and Minematsu continue by showing the security of one-key XTS-AES and suggest a sketch for a proof that shows the security of ciphertext stealing within XTS-AES. Their recommendation is to allow ~~the~~ one-key XTS-AES.

Comments From Vijay Bharadwaj and Neils Ferguson of Microsoft

(see [MiscRsp])

> ***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.*

[Response ~~TBD~~]

While I believe that IEEE std 1619-2007 does provide a good discussion on the security of XTS, it is true that we could have added more details about all the applications that are suitable for XTS. This is a somewhat subjective area, and it is ultimately up to the designers of the cryptographic system to determine whether this mode is appropriate, given their specific design requirements.

> ***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 2^{20} 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.

[Response ~~TBD~~]

Traffic analysis (i.e., monitoring temporal effects) is a weakness of many encryption modes, and for XTS is addressed in IEEE std 1619-2007, section D.2. However, this attack does not seem practical. To monitor snapshots over time, the attacker would need a large local disk for storing the snapshots, or a high-bandwidth connection for sending this data to the attacker's own disk array. An attacker with this level of access probably has direct access to the plaintext and need not bother with traffic analysis.

In general, IEEE 1619-2007 assumes that the attacker cannot perform traffic analysis, but only has access to the ciphertext once, generally after stealing a portable computer and examining the hard drive.

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 $P_i \oplus C_i = P_j \oplus C_j$ *. 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* $P_i \oplus P_j = PP_i \oplus T_i \oplus T_j \oplus PP_j = T_i \oplus T_j = (T \otimes \alpha^i) \oplus (T \otimes \alpha^j) = T \otimes (\alpha^i \oplus \alpha^j)$ *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 2^{20} blocks.

[Response ~~TBD~~]

Clearly, allowing large data units is dangerous. The working group discussed this limit at length and decided to leave it as a 'should' instead of a 'shall' because it is up to the designers to determine the amount of risk they are willing to accept for a collision and leak of the master tweak in the data unit.

That said, it is quite reasonable to limit FIPS-approved applications to 2^{20} blocks within a data unit.

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.

[Response ~~TBD~~]

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.

~~[Response TBD]~~

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.

[Response ~~TBD~~]

IEEE Std 1619-2007 acknowledges in D.3 that the attacker has greater malleability with 16-byte narrow block encryption modes like XTS, versus wide-block encryption modes like EME-2 or XCB (see IEEE P1619.2). However, by making this trade-off, the encryption algorithm need only make one pass over the data unit instead of the two passes required by wide-block encryption modes. It is important for the designers to understand this trade-off and pick the appropriate mode. XTS is appropriate if the attacker has limited access in making malicious attacks and if the programs have a high probability of detecting a randomized 16-byte block.

Comment From Phillip Rogaway of Univ. of CA, Davis

(see [MiscRsp])

> *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*[XTS] *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.*

Response:

Concerning the security of ciphertext stealing, please see the security sketch offered by Liskov and Minematsu in [Liskov]. While this is not an absolute proof, it does suggest that ciphertext stealing is sufficiently secure for applications that use XTS. Also note that NIST has proposed incorporating a similar ciphertext stealing mode for CBC (see [CBC-CS]). Despite lacking a formal security proof, ciphertext stealing still has general approval in the cryptographic community.

> *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.*

Response:

~~**[TBD]**~~I agree with this sentiment, and I have heard interest in later submitting modes such as EME-2 or XCB from IEEE P1619.2. However, there are many applications, like internal hard disk encryption, that cannot afford the extra hardware or latency required by a two-pass wide-block encryption mode. These applications are still secure with XTS because the attacker has limited access to the ciphertext.

Comment From Boaz Shahar of Entropic Communications

(see [MiscRsp])

> *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.*

Response:

The P1619 task group chose little-Endian order because this is optimal for software written on little endian systems, which at this time is the most popular byte-ordering scheme, being on consumer grade x86 systems.

The performance boost is significant on these systems because there is no need to spend cycles to convert the byte-ordering to the native format

In hardware, the performance is the same, although the Verilog or VHDL may be less intuitive.

See threads on P1619 e-mail reflector for more details on the specifics behind this design choice.

Comments From Michael Willett of Seagate Technology

(see [Seagate1])

Weaknesses of XTS

1. *Unchanged data layout: unnecessary restriction. Data encryption SW, External encryption modules, Encrypting controllers, and Encrypting devices can all:*
    1. *reserve space on the medium*
    2. *relocate blocks*
    3. *dissect/merge blocks, etc.*
2. *Much available metadata is left unutilized:*
    1. *in SW: file name, access date, file type, attributes, etc.*
    2. *in SCSI: protection info (checksum, data owner…)*
    3. *in disk drives: age of data, drive ID, track geometry, checksum…*
    4. *in disk arrays: disk ID, array name*
3. *Encryption during sequential (DMA) transfer is not optimized (speed, power consumption)*
4. *Encrypted data in transit needs further protection: Address is in the clear, data is susceptible to traffic analysis*
5. *Tweaked ECB mode: the same data in the same place is still recognizable, leaks at repeated peeks*
6. *Small block encryption:*
    1. *malleability of documents (especially the ones containing different versions)*
    2. *SW: jump address patch, short data can be changed to desired value with non-negligible probability, with little collateral damage*
    3. *undetected data destruction*
    4. *DRM info, file fingerprint manipulation*
    5. *File system manipulation*
    6. *Hiding data from virus scanners, etc., by just moving it*
7. *Export control: the encryption module is a high speed, mass encryption device*
8.

9. *XTS has extra complexity compared to CBC (2nd key, Galois multiplier, two XOR ops). What does it buy?*
10.
    - *Only some protection against copy-and-paste attacks.*
11.
12. *The XTS standard Annex D.4.3. says that there is a "strong security guarantee as long as the same key is not used to encrypt much more than a terabyte of data". This is not the capacity of the storage device, but the observed amount of cipher-text by an adversary. If he can record the encrypted traffic or make repeated snapshots of the stored data, this limit can be reached easily at smaller storage devices, too. Therefore, the user must periodically re-encrypt large amounts of stored data, or partition the storage space into many bands, each encrypted with a different key. It is very expensive in processing time, in power consumption, or in the complexity of securely storing and managing many keys, etc.*

Response:

~~**[TBD]**~~Please see responses to correspondingly number sections below:

1. I do not see a restriction on data layout imposed by XTS. In IEEE Std 1619-2007, the definition of a data unit states that "a data unit does not necessarily correspond to a physical or logical block on the storage device. It is possible to provide whatever necessary logical mapping is needed to assign unique

tweak values to each data unit.

2. I do not see how unused metadata is relevant to XTS.
3. I do not see how DMA transfer is not optimized under XTS. XTS allows parallel processing, so there is nothing intrinsic to XTS that would prevent optimal DMA transfer. I could understand this argument for a wide-block mode that requires two passes across the data, but not for narrow-block encryption.
4. XTS is not designed to protect encrypted data in transit. XTS is for storage encryption. There are many other good protocols, like ESP, TLS, or FC-SP, that protect data in transit.
5. This is addressed in the previous discussion on traffic analysis
6. See previous response to a similar comment by Bharadwaj and Ferguson. Narrow-block encryption does have greater malleability than wide-block encryption, but there is a trade-off between processing time and performance.
7. U.S. export control for mass encryption devices is under the Bureau of Industry and Security (BIS), not by NIST, and as such I do not see the relevance in this context.
8. The extra complexity of XTS compared to CBC is not as large as is stated here. An implementation does not need a Galois Multiplier, but can instead use a low-cost linear feedback shift register (LFSR) CBC already has an XOR operation of its own, so XTS adds one XOR operation, not two. Overall, compared to the cost of an AES cipher, these additions are minimal, and afford many advantages, like parallel operation and decreased malleability by the attacker.
9. When Annex D.4.3 states that the same key should not encrypt more than 2^{40} blocks (about 16 Terabytes), this is with the extremely conservative probability of a distinguishing attack succeeding as 1 in 2^{-53}, which is extraordinarily conservative. Also, the consequence of a success in this attack is being able to distinguish XTS from an ideal tweakable block cipher, not to recover the encryption key or plaintext. It would be quite reasonable to encrypt more than 16 Terabytes under the same key, especially if there is reason to believe that traffic analysis is not a threat to a particular system.

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

Current Industry Support

The industry support for XTS-AES is substantial, although this wasn't reflected in the public comments received by NIST. After talking to representatives of these companies, many thought that sending a letter of support to NIST was unnecessary because the support was already overwhelming.

As of ~~January~~March 2009, here is a list of companies and products that support or plan to support XTS-AES encryption:

Hard Disk Vendors

I have been contacted by at least two major hard disk vendors who stated that a future hard disk drive will support XTS-AES encryption. Other hard disk vendors are planning to support XTS, but are waiting on a decision from NIST before making announcements.

AMCC (from Paul Von Stamwitz)

AMCC has a product, the PPC 460SX, that supports XTS. See:

https://www.amcc.com/MyAMCC/jsp/public/productDetail/product_detail.jsp?productID=PPC460SX

Bloombase Technologies (from Harry Xiang)

Bloombase Spitfire storage encryption server supports XTS-AES for SAN, DAS, disk and tape encryption as usual we do for other regional standard bodies in support of Camellia, SCB2, SSF33, etc.

Brocade (from Scott Kipp)

Brocade uses XTS-AES in our new line of encryption products including the Brocade Encryption Switch - a 32 port switch, and the FS8-18 blade for the Brocade DCX Backbone. We use XTS-AES to encrypt disk-drive based products These are being sold through several companies such as EMC and NetApp with more to come. To find out more about our products, you can go here:

http://www.brocade.com/products-solutions/solutions/security/products.page

CipherMax (from Steven Tan)

CipherMax newest family of CM200D, CM250 and CM500 FC security switch products uses XTS-AES mode for disk encryption.

CM200D is a 1U 16port FC switch with support for disk encryption. The CM250 is a mid-range enterprise class FC switch supporting up to 4 x 16 port ELC1624 line cards for disk encryption (XTS-AES) or tape encryption (XTS-CCM) that can scale from 16 to 64ports. Similarly, the CM500 is an enterprise class FC switch that uses the same ELC1624 line cards for disk or tape encryption supporting up to 16 x ELC1624 cards that can scale from 16 to 256ports.

EMC (from David Black)

EMC PowerPath Encryption with RSA supports AES XTS encryption:

http://www.emc.com/products/detail/software/powerpath-encryption-rsa.htm

http://www.emc.com/collateral/software/white-papers/rkmpp-sb-0808-lowres.pdf

See p.8 of the latter whitepaper.

Freescale Semiconductor (from Michael Torla)

XTS can be found in the following shipping Freescale products:

* *PowerQuicc devices:*
  - *MPC8314E*
  - *MPC8315E*
  - *MPC8536E*

and XTS is planned for the following announced but not-yet-released Freescale products:

* *QorIQ devices:*
  - *P1010*
  - *P1020*
  - *P2020*
  - *P4080*
* *DSP devices:*
  - *MSC8156*

All future QorIQ devices will include XTS.

GED-i (from Rony Shapiro)

GED-i uses XTS-AES in its GSA-2000 storage disk encryption product line. GSA-2000 devices provide storage disk data encryption and security for iSCSI and FC SAN servers. Our products are certified as interoperable with Sun and IBM storage solutions.

For more information, see http://www.ged-i.com

Helion (from Colin Sinclair)

Helion have FPGA and ASIC IP Cores available for AES-XTS mode, also providing CBC mode for legacy compatibility. We support both key sizes, variants for encrypt-only or full encrypt-decrypt, with optional ciphertext-stealing support. Our product pages can be found here:

http://www.heliontech.com/storage.htm

http://www.heliontech.com/aes_xex.htm

Hifn (from Doug Whiting)

Hifn has a chip that is taped out and coming back very soon that includes AES-XTS mode. It's a generic crypto/compression lookaside (PCIe) processor.

Oxford Semiconductor (from Gary Calder)

Oxford Semiconductor supports XTS-AES for inclusion in future products.

PMC-Sierra (from Gary Nichols and Peichen Chang)

PMC-Sierra has both a FibreChannel Protocol Controller with XTS and a SAS Protocol Controller with XTS.

PMC is currently shipping the Tachyon QE8e+ Quad Port Fibre Channel protocol controller with XTS-AES encryption. The primary use model for the encryption function is to protect data written to disks. Tachyon products power approximately 70% of the world's disk array systems. Moving forward, PMC plans to continue offering XTS-AES functionality in our Fibre Channel and SAS/SATA protocol controllers

SafeNet (from Marcel van Loon)

SafeNet provides a family of Silicon IP cores supporting XTS-AES in a range of performances. This IP gets included in our customer's products that require support for XTS-AES. We've seen considerable interest in these cores from partners.

A description of the SafeXcel EIP-38 family of cores can be found here:

http://www.safenet-inc.com/products/ip/safeXcel_IP_AESGCMXTS_Acc.asp

Thales (from Bob Lockhart)

Thales plans to implement XTS if it is accepted by NIST as a certified mode of operation.

TrueCrypt

TrueCrypt, the leading open source disk encryption software, supports XTS as the default mode.

WinMagic (from Gary McCracken)

WinMagic will implement XTS-AES when it becomes a FIPS 140 approved mode of operation.

WinMagic produces full disk encryption software for desktops and laptops. FIPS 140 certification is required by many of our customers.

Comment from David Clunie

(see [MiscRsp])

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).

Response:
NIST has standardized many previous modes of operation (e.g., ECB, CTR, CBC, GCM, CCM) without standardizing the corresponding key management of the keys used in these modes. SP 800-56A describes a framework for establishing secrets using Diffie-Hellman and ECC, and SP 800-56B describes using RSA for establishing a shared secret. SP 800-57 provides recommendations for key management, but does not go into specific protocols or recommendations for interoperability. In general, NIST has kept its requirements general and left it to other organizations, like IETF or IEEE, for defining the specifics.

For current key management efforts that are standardizing key interchange, see IETF KEYPROV, OASIS EKMI, IEEE P1619.3, or other related efforts.

Comments From Michael Willett of Seagate Technology

(see [Seagate1])

In sealed secure storage the following are possible and desirable, but not considered with XTS:

- *access control*
- *user authentication*
- *data authentication (integrity check)*
- *tamper detection.*

For example, with the (mandatory) access control, copy-and-paste attacks are automatically thwarted, but their prevention is the only distinguishing security feature of XTS.

For secure removable media storage application (like tape) the following are desirable, but missing from XTS:

- *data authentication*
- *large block encryption*
- *independent encryption from the (possibly changing or unknown) data location*
- *unpredictable tweak (user supplied, stored IV)*

Response:

Access control and authentication are essential components in any cryptographic system, but do not generally relate to cryptographic modes. Integrity checks and IVs are out-of-scope for XTS because the constraint of the mode requires that no extra data be added. If you have room for an integrity check, then CCM, GCM, CBC-HMAC, XTS-HMAC (e.g., modes standardized in IEEE std 1619.1-2007) are more appropriate.

Is XTS a good tradeoff between security and complexity (costs)? NO:

- *There are asymmetric large block encryption modes (e.g. BitLocker of Windows Vista) with similar complexity but better security for storage systems*
- *There are faster, simpler encryption modes used together with access control (e.g. CBC)*

Response:

Large block encryption modes need two passes over the data, so it is hard to characterize this as 'similar complexity'.

CBC is not universally faster and simpler than XTS. XTS can operate in parallel, which makes it faster when multiple AES execution units are available.

What security problem XTS addresses? (Threat model, attack scenarios)

- *Lost laptop scenario: CBC with encrypted address as IV is adequate, faster, simpler, cheaper*
- *Janitor access of locked PC: If the key is properly destroyed in memory, it is similar to the lost laptop case*
- *Disk theft from datacenter: ~ lost laptop*
- *When repeated access to the ciphertext is assumed (removable media, external encryption module): Only copy-and-paste attacks are mitigated by XTS, all other small block problems remain.*

Response:

XTS provides a solution for storage encryption systems where the attacker has limited access to ciphertext, and where using wide-block encryption or authenticated encryption is not feasible due to limited processing power or space limitations. XTS is less malleable than CBC because an attacker can flip particular bits within a block, but with XTS can only randomize the whole block. Also, XTS is a parallel design and CBC is serial, making XTS faster in environments that have multiple AES execution units.

Ciphertext stealing of XTS is defined with a block swap. There is no reason to destroy the straight block order, causing unnecessary implementation difficulties, confusion, incompatibilities with existing modules.

- *Without the swap of the last two blocks, the data buffer could be accessed sequentially backward or forward.*
- *More recent definitions of ciphertext stealing do not swap blocks (see Henk, Tillborg: Encyclopedia of Cryptography and Security, Springer 2005, p387.)*

Response:

There is no reason that you could not add a special 'formatter' that swaps the last two blocks before writing it to disk. I will recommend that NIST clarify this. The ordering of the blocks in ciphertext stealing is not a security issue, and there are situations where either order is preferable.

Danger of standardizing a “disk” encryption mode, which is not optimal for storage encryption:

- *Customers demand it in place of more secure solutions (as “the” secure storage standard)*
- *Computer manufacturers relay the demand of their customers to disk manufacturers, SW houses, controller designers; forcing wide spread implementation of a less secure, more*

expensive encryption mode

- *In the most common applications there are unnecessary costs in key generation/storage, algorithm complexity, speed, power consumption*
- *Poor security systems can be sold as standards conformant, thereby reducing the overall security in storage*

[Therefore, if the XTS mode gets approved by NIST, it must not be labeled as "storage/disk encryption", but something like as an "encryption mode, resistant to moving blocks".]

[Response

~~TBD]~~This last comment seems subjective and is only an issue if XTS truly is inferior. However, as has been explain multiple times, XTS has particular strengths that make it suitable for applications that have fixed-size sectors and lack the processing power to do two passes over the data for a wide-block encryption. NIST offers a toolbox of encryption modes that all have their strengths and weaknesses in particular contexts, and XTS is no different in this regard.

The appeal of XTS for wider applications

Comment From David Clunie

(see [MiscRsp])

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.

Comments From Michael Willett of Seagate Technology

(see [Seagate1])

Why standardize XTS at all? No good reason:

1. *Interoperability: dumb disks with encrypted data can be connected to different controllers. BUT most new disk drives have internal encryption, and their platters cannot be mixed and matched.*
2. *Archived data recovery in the distant future: Small audience. Only specialized archival devices benefit from a standard, if their encryption module is separate from the medium (tape, CD-ROM…). However, these devices are better served by higher security encryption*
3. *By employing a standard there is no need to perform security analysis for the encryption mode (as for a proprietary solution). BUT the encryption mode is the least of the worries. Some other harder issues to be considered:*
    - *the applicability of XTS needs to be shown*
    - *the implementation has to be validated*
    - *implementation errors have to be avoided: audits, reviews, tests (buffer overrun, ill formed messages, weak parameters…)*
    - *robust protocols (message replay, man-in-the-middle attacks, non-random nonces, weak session keys…)*

- ~~-~~*firmware code, crypto keys, sensitive memory content need protection*
- ~~-~~*authenticated secure firmware (prevent ROM swap/patch, safe update)*
- ~~-~~*protection of wires, pins, RAM chips (freezing attacks)*
- ~~-~~*protection of debug ports*

[Response:

~~**TBD]**~~Standardizing XTS provides a parallel encryption mode that has roughly the same computation complexity as CBC, but provides less malleable ciphertext. Clearly there are other issues beside the encryption mode needed to make a system secure. But this is true for all the SP 800-38 series encryption modes.

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

Comment From Vijay Bharadwaj and Neils Ferguson of Microsoft

(see [MiscRsp])

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

Comment from Rich Schroeppel

(see [MiscRsp])

> *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.*

Comment From David Clunie

(see [MiscRsp])

> *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.*

Response:

The IEEE-SA (IEEE Standards Association) and the members of the IEEE P1619 task group are financially independent entities. The IEEE-SA oversees the standards creation process and pays for editors and other publication costs, and recuperates its costs by selling the standard. The IEEE P1619 task group members include representatives of companies who create products with XTS-AES.

IEEE-SA does offer a way for the standards developers to pay for IEEE's publication costs, and have the standard offered to the public for free (see, for example, the IEEE 802 series). However, the group members were unable (and in some cases unwilling) to pay this fee.

Comments From Michael Willett of Seagate Technology

(see [Seagate1])

> *As stated in the NIST e-mail, the IEEE has provided a "free" copy of the XTS algorithm description on a temporary basis for the purpose of supporting this Call for Comment. But, after 90 days, the IEEE will charge for a copy of the algorithm ($85/$105). As far as we know, this is without precedent for NIST: to certify an algorithm for which the NIST public has to pay. Seagate is against being charged. But, Seagate is against NIST certification of this algorithm in the first place.*

General Response

Most of the comments were highly critical of requiring that implementors purchase the IEEE 1619 standard. This is a natural response. We have come to expect free standards from NIST and this goes against that expectation. NIST itself prefers free standards, as do the members of the IEEE P1619 working group.

However, all standards cost money. It's just a matter of who pays. In this case, the members of IEEE P1619 did not pay anything fees to participate, nor did NIST pay to develop this standard using U.S. taxpayer money. This structure allowed the participation of top cryptographers to develop the details of the XTS-AES mode, free of cost to them.

Unfortunately, if NIST doesn't pay, and the IEEE P1619 members don't pay, then the publication costs get passed on to the user. For corporations, this is generally not an issue because there is an expectation to pay for standards as part of product development. Additionally, many companies already have company-wide access to the IEEE library, allowing employees to download IEEE 1619 at no additional cost.

Also note that NIST has an existing precedent to reference standards that cost money. Here is a partial list of standards that NIST references and that cost money now, or used to cost money:

1. ANS X9.31-1998, Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA). (as referenced in FIPS 186-3 and FIPS 140-2, Annex C)
2. ANS X9.62-2005, Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA). (as referenced in FIPS 186-3 and FIPS 140-2 Annex C)
3. ANS X9.80, Prime Number Generation, Primality Testing and Primality Certificates. (as referenced in FIPS 186-3)
4. IEEE 802.11i (as reference in FIPS 140-2 Implementation Guidance)

Overall, referencing standards that cost money is a common industry practice, and in this case is a good way for NIST to save money for tax payers, although it unfortunately does cause some inconvenience to some implementors.

Concerns of intellectual property rights

Comments From Vijay Bharadwaj and Neils Ferguson of Microsoft

(see [MiscRsp])

> *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.*

Comment From David Clunie

(see [MiscRsp])

> *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.

Comments From Michael Willett of Seagate Technology

(see [Seagate1])

As stated in the NIST e-mail and web site:

"The chair of the (Security in Storage) SISWG informed NIST that he is unaware of any patent claims on XTS, but that NeoScale Systems, subsequently acquired by nCipher, submitted a Letter of Assurance of Essential Patents to the IEEE, without elaborating on what aspect of IEEE 1619 was patented."

This information suggests that the XTS is encumbered with possible patent claims of unknown quantity and ownership, aided by the voluntary nature of the IEEE IP declaration process. NIST should not certify an algorithm under such circumstances, since the NIST certification would encourage mandatory adoption of XTS in the storage industry, with unknown royalty consequences on the implementers. AES candidates were solicited with royalty-free pre-conditions and the granting of any patent rights to NIST. We expect no less for a certified mode for AES.

General Response

Many commenters were uneasy with the ambiguity of the intellectual property (IP) rights situation, in regards to the Letter of Assurance (LoA) submitted by NeoScale systems. In truth, this ambiguity exists for all technology. There are many potentially unenforceable patents that are continually submitted to the patent office, and which are thrown out or substantially changed after years of prosecution. This is just a case where were are made aware of a claim that *might* be relevant.

Investigation NeoScale Systems' IP Claims

Subsequent to submitting the LoA to IEEE, NeoScale Systems declared bankruptcy, and the bulk of the assets were purchased by nCipher (now owned by Thales). However, after nCipher investigated the matter further, they discovered that they did not purchase the IP in question (i.e., that covers IEEE Std 1619). This IP is still the property of the bank that acquired NeoScale's assets as a result of the bankruptcy.

From searching for patents owned by NeoScale Systems, the following patents applications are current as of Jan 2009. Please note that that I am not a lawyer and that these comments are only my opinion (commentary in *italics*):

- **20090013016** (Noll, LeBlanc), SYSTEM AND METHOD FOR PROCESSING DATA FOR DATA SECURITY, Filed July 2007: *This application does not mention XTS or IEEE 1619, although it does patent some general storage encryption methods.*
- **20080273706** (Noll), System and Method for Controlled Access Key Management, Filed May 2007: *Covers key management and does not mention XTS or disk encryption.*
- **20060277387** (Rhoades) System and method for hardware allocation of memory resources, filed April 2006: *Does not apply to cryptography.*
- **20050080761** (Sundararajan, et al) Data path media security system and method in a storage area network , filed Oct 2003: *Does not discuss XTS or related encryption techniques.*
- **20050041812** (Sundararajan, et al) Method and system for stateful storage processing in storage area networks, filed Oct 2003: *Does not discuss XTS or related encryption techniques.*

- **20050033988** (Chandrashekhar, et al) Method and system for transparent encryption and authentication of file data protocols over internet protocol, Filed Oct 2003: *Mentions AES encryption, but does not mention XTS or XEX. The part to watch for is a claim on a NAS server where the "[encryption or decryption] is provided by a NIST approved process".*

I could not find any granted U.S. patents that are ~~issued~~assigned to NeoScale Systems.

NeoScale Systems submitted a Letter of Assurance to IEEE in March 2007 (see [NLoA]). It is unclear which specific patent application this was intended to cover, but none of the published applications specifically cover the XTS or XEX mode of encryption.

IP Claims by Rogaway

The creator of XEX, Philip Rogaway, does not have any IP claims on XEX (and by extension, XTS) (see [RogIP]). Here is the statement from Rogaway to Shai Halevi:

> *I know of no IP claims that would in* [fact] *touch upon XEX itself; I myself never claimed anything in the direction of a tweakable conventional blockcipher.*

Conclusions and Recommendations

~~[TBD]~~Overall, I recommend that NIST accept XTS-AES (as defined in IEEE Std 1619-2007) as an Approved Mode of Operation with the following changes:

- Allow 'one-key' variants of XTS-AES, where the same 128-bit or 256-bit key is used within both AES ciphers.
- Prohibit data unit sizes larger than 2^{20} blocks.
- Clarify that the ordering of the blocks in ciphertext stealing is a logical ordering, and can be mapped to a different physical ordering (i.e., it is possible to swap the order of the ciphertext stealing blocks)

References

[Ball] Ball, Matthew, "NIST's Consideration of XTS-AES as standardized by IEEE Std 1619-2007." Sept 2008. See http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/XTS/XTS_comments-Ball.pdf

[CBC-CS] National Institute of Standards and Technology (NIST), *Proposal To Extend CBC Mode By "Ciphertext Stealing"*, May 2007. See http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/ciphertext%20stealing%20proposal.pdf

[Liskov] Liskov, Moses and Kazuhiko Minematsu, "Comments on XTS-AES." Sept 2008. See http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/XTS/XTS_comments-Liskov_Minematsu.pdf

[MiscRsp] Bharadwaj, Clunie, Ferguson, Rogaway, Shahar, and Shroeppel, "Public Comments on the XTS-AES Mode." (Collected XTS Comments) Sept 2008. See http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/XTS/collected_XTS_comments.pdf

[NLoA] NeoScale's Letter of Assurance to IEEE on IEEE Std 1619, March 2007, see http://standards.ieee.org/db/patents/loa-1619-neoscale-08Mar2007.pdf

[Request] NIST, Request for Public Comment on XTS, 2008, see http://csrc.nist.gov/groups/ST/documents/Request-for-Public-Comment-on_XTS.pdf

[RogIP] Rogaway, Phillip, "Quick check on IP situation of XEX", Sept 2006, See http://grouper.ieee.org/groups/1619/email/msg01309.html

[Seagate1] Willet, Michael, Seagate Technology, “Comments provided to NIST in response to: Request for Public Comment on XTS/AES.” Sept 2008. See http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/XTS/revised_XTS_comments-Seagate.pdf

[Seagate2] Hars, Laszlo, Seagate Technology, “Response from Laszlo Hars.” Sept 2008. See http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/XTS/XTS_comments-Hars.pdf