Comments provided to NIST in response to:

Request for Public Comment on XTS/AES

http://csrc.nist.gov/groups/ST/documents/Request-for-Public-Comment-on_XTS.pdf

From:

Seagate Technology
Michael Willett: administrative contact
1251 Waterfront Place
Pittsburgh, PA 15222

Cell: 412 225 1512

Arranged into the five categories suggested by NIST:

➢ The XTS algorithm itself

Summary: the XTS encryption mode is not a good choice for storage encryption. There have been simpler, faster or more secure modes in use.

Description: XTS is ECB mode with tweaks, to prevent attacks involving movements of data blocks and hiding the equality of data blocks at different locations. It was designed such that

1. The ciphertext is freely available for an attacker
2. The data layout is not changed on the storage medium and in transit
3. Data is accessed in fixed sized blocks, independently from each other
4. Encryption is performed in 16 byte blocks, independently from other blocks (except the last two plaintext blocks of a sector, if its size is not a multiple of 16 bytes)
5. There are no other metadata used, except the location of the data blocks within the whole data set
6. The same plaintext is encrypted to different ciphertexts at different locations, but always to the same ciphertext when written to the same location again
7. A standard conformant device can be constructed for decryption of data encrypted by another standard conformant device.

Weaknesses of XTS
- Unchanged *data layout*: unnecessary restriction. Data encryption SW, External encryption modules, Encrypting controllers, and Encrypting devices can all
  - reserve space on the medium
  - relocate blocks
  - dissect/merge blocks, etc.

- Much available *metadata* is left unutilized:
  - in SW: file name, access date, file type, attributes, etc.
  - in SCSI: protection info (checksum, data owner...)
  - in disk drives: age of data, drive ID, track geometry, checksum...
  - in disk arrays: disk ID, array name
- Encryption during sequential (DMA) transfer is not optimized (speed, power consumption)
- Encrypted *data in transit* needs further protection: Address is in the clear, data is susceptible to traffic analysis
- Tweaked ECB mode: the same data in the same place is still recognizable, leaks at repeated peeks
- *Small block* encryption:
  - malleability of documents (especially the ones containing different versions)
  - SW: jump address patch, short data can be changed to desired value with non-negligible probability, with little collateral damage
  - undetected data destruction
  - DRM info, file fingerprint manipulation
  - File system manipulation
  - Hiding data from virus scanners, etc., by just moving it
- *Export control*: the encryption module is a high speed, mass encryption device

XTS has extra *complexity* compared to CBC (2nd key, Galois multiplier, two XOR ops). What does it buy?
- Only some protection against copy-and-paste attacks.

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

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)

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)

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.

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.)

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".]

➢ The appeal of XTS for wider applications

Why standardize XTS at all? No good reason:
- 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.
- 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
- 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

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

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.

➢ Concerns of intellectual property rights

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.