P1619 Narrow-Block Encryption

Contributed by Administrator
Thursday, 19 July 2007
Last Updated Tuesday, 27 December 2011

Standard Architecture for Encrypted Shared Storage Media

(Original Project 1619
August 14, 2002 -- updated May 9, 2007)

Scope: This standard specifies
elements of an architecture for cryptographic protection of data on
block-oriented storage devices, describing the methods, algorithms, and
modes of data protection to be used.

P1619 Task Group Chair: Landon Noll, Cisco Systems

P1619 Task Group Editor: Subhash Sankuratripati, NetApp

Current StatusThere is currently an effort to revise P1619 to also include a one-key XTS-AES mode. See the current PAR at IEEE's P1619 project page by clicking on the Approved PAR link. The current PAR expires December 2013.

HistoryIn 2007, the IEEE P1619 task group completed work on a standard for the XTS
encryption algorithm, which is suitable for encryption of stored data
in a fixed-block device, and a standard for an XML-based key-export
format. XTS stands for 'XEX TCB with ciphertext stealing' and is a
narrow-block cryptographic mode. (XEX stands for 'XOR-Encrypt-XOR', and
TCB is Tweakable CodeBook mode encryption). This standard was approved in December 2007 by the IEEE Standards board.

On Sept 4, 2008, NIST completed a public review for XTS-AES and has posted public comments here. Based on these comments, NIST will make a decision whether to adopt XTS-AES as an approved mode of operation under FIPS 140.

How to purchase IEEE Std 1619-2007
To purchase from the IEEE Store, follow these instructions:

- Go to http://www.ieee.org/
- Click on 'Shop' at the very top
- Click on the 'Standards' tab

- Click below 'Search the IEEE shop' on the right side

- Enter "1619" and click 'Search'

- Add the appropriate standards to your Shopping Cart and check-out

For IEEE members, you can use IEEE Xplore (a different way to buy IEEE standards) if you or your company has a subscription to the Digital Library ($35/month).

IEEE 1619: http://ieeexplore.ieee.org/servlet/opac?punumber=4493431

Reference ImplementationsHere is a list of known reference implementations for XTS. If you know of more, please send an e-mail to the chair:

- libtomcrypt

- Brian Gladman's XTS Implementation on his AES code page

FAQ for 1619-2007

This list contains the informal errata on IEEE Std 1619-2007 based on discussion from the e-mail reflector. For official errata, see the IEEE errata sheets .

(Please see Minematsu's comments for other errata)

Q1: "The relation between
"data-unit" and "key-scope": The mapping between data
unit and a key is one to one, or some data-units can be encrypted with
the same key?"

A1: A key scope would normally
apply to more than one data unit. In fact, section 3.1.1 implies that it's
more than one. Although you *could* define key scope == one data unit if
you really wanted.

Q2: "What is the typical
size of such a data unit? I understand it is outside the scope of this
work, but you probably have a size in mind. Is that a disk sector?"

A2: It would make a lot
of sense to use a sector's worth of data as your data unit, or maybe a
file system cluster's worth if you're encrypting a file. For arbitrary
data streams on other types of media there may be other sizes that are
useful in a particular situation.

Q3: "-In line 30 and
31 page 3 there is statement says that the size of a data-unit is 2^128-2
128 bit block. On the next sentence it says that the size of the data-unit
is 2^20 128 bit block. This seems to be a contradiction."

A3: I think the key is in
the words preceding the numbers - namely "shall not" (meaning
may not without grave consequences) and "should not" which isn't
quite as restrictive. No contradiction there.

Q4: " j, the index
of the 128 bit block within a data-unit starts from 0 or 1 (i.e. the first
128 bit block of P is xored with AES(i) or with AES(i)*ALFA?)"

A4: The code in Annex 3
suggests that j starts from 0, as the first multiplication with alpha happens
after XORing the tweak to the first plaintext / ciphertext blocks.

Q5: "Since I (tweak
value), Key1, Key2, are constant for a given data-unit, and j is sequentially
incremented, it seems that within a certain data-unit it is enough to multiply
each T by ALFA in order to get the next T. Is that correct? (i.e. T(n+1)

= T(n)*ALFA) "

A5: Right. That's exactly
what the example code in Annex 3 does.

Q6: " In the formula:
Cq
← 1
XTS-AES-blockEnc(Key, Pj,
i, q) Pj can
be replaced by Pq, correct?"

A6: I think you're right.
Any C(q) would be derived from a P(q) and the tweak. I think j is a typo and should be replaced with q as j is a loop invariant. Same goes for the decryption formula on page 7.

Q7. In section
5.1, should/must "Data Units" be of
common, equal size? is this strictly
enforced...

A7:
Within a particular key scope (i.e., the range of data encrypted by a particular key) all data units shall be the same size. If the data units are not the same size, then the implementation is not compliant with XTS-AES. Enforcement is outside the scope of this standard.

Q8. If data units are
not a multiple of 16 bytes
in length (they need not be), then each data
unit
should do cipher-text stealing for encryption of the last two blocks
(see Figure
2). Is this correct?

A8:
Although the standard does not directly state it, if the size of the data unit within a particular key scope is defined as a non-multiple of 128-bits, then each data unit within the key scope shall use ciphertext stealing.

Q9: If the amount of
data to be encrypted is not a multiple of the data unit size, what is
typically done?

A9: XTS-AES was designed for situations in which the fundamental unit of encryption (i.e., the data unit) is fixed and it is not possible to change (example: hard disks with 512-byte logical blocks). In a hard disk, if a file does not fit exactly within an integer number of logical blocks, then the remainder of the file is typically padded out to the end of the logical block, and the file size is recorded within metadata elsewhere. If you particular application allows flexibility in the data unit size, then it is more appropriate to use an authenticated encryption mode, such as those defined in IEEE Std 1619.1-2007. For example, tape drives allow variable-sized logical blocks, and will append a MAC (Message Authentication Code) to the end of each encrypted record.

Q10:
Is the end of
the data padded (best practice for padding) until it fills a
complete data unit?

A10:
Padding is outside the scope of IEEE Std 1619-2007.