XTS mode revisited: high hopes for key scopes?

Milan Brož[1,2]
gmazyland@gmail.com

Vladimír Sedláček[2]
vlada.sedlacek@mail.muni.cz

[1]Cryptsetup project
[2]Faculty of Informatics, Masaryk University, Czechia

October 28, 2025

Abstract

This paper concisely summarizes the XTS block encryption mode for storage sector-based encryption applications and clarifies its limitations. In particular, we aim to provide a unified basis for constructive discussions about the newly introduced key scope change to the IEEE 1619 standard. We also reflect on wide modes that could replace XTS in the future.

1 Introduction

Since the introduction of XTS [10] in 2007, many software and hardware sector-based encryption systems have widely adopted it, e.g., BitLocker [14], VeraCrypt [2], Cryptsetup [1], and TCG Opal [8] self-encrypting drives. Still, it seems to be surrounded by a shroud of misunderstandings, and some of its subtler points are still sometimes contested in the developer communities. It does not help that the existing NIST XTS-AES recommendation [7] only references the paid version of the IEEE standard [10], making it probably the only NIST document that does not include a publicly available definition of the described cryptographic primitive (as already mentioned in Rogaway's analysis [20] in 2011).

The situation is especially problematic in light of the newly introduced key scope change to the IEEE 1619-2025 standard [12] (again behind a paywall), which would render the vast majority of implementations non-compliant. This would force vendors to request (not only) open-source projects to make significant changes to adapt to the new version. In contrast, the reasons and security implications of the change are not communicated at all.

The goals of this paper are threefold:

- Unify the terminology in the XTS context in a way easily understandable to both theoreticians and practicioners (Section 2 and 3).
- Describe XTS security limitations (namely key scopes, maximal sector size, and distinct keys) and clarify some contested points (Section 4).
- Encourage a constructive public discussion to influence future requirements and recommendations (Section 5).

2 Terminology and definitions

2.1 Units and sectors

The storage industry is known to mix the power-of-10 units (SI) and the power-of-2 units for a storage size. To avoid any confusion, we strictly use powers of two with IEC prefixes: a byte is 8 bits, a kibibyte (KiB) is 2^{10} bytes, a mebibyte (MiB) is 2^{20} bytes (1024 KiB), and a tebibyte (TiB) is 2^{40} bytes (1024 MiB).

A block device is an abstraction of either a physical device (like a hard disk or a flash-based device) or a virtualized device that can be accessed (read, written to, or erased) only in units called sectors. A typical sector size is 4096 bytes (4 KiB) or 512 bytes (for older devices). In this text, we will only consider encryption of block devices (even though XTS can be used in other scenarios). Furthermore, we assume that the sector is always read or written to atomically, and individual sectors are accessed independently.[1]

The typical blocksize for modern blockciphers is much smaller than the sector size (16 bytes for AES). Thus, to encrypt the whole sector, we need to use an encryption mode (such as XTS) that builds upon the original cipher. All modern real-world storage devices always use sector sizes that are multiples of 16 bytes, eliminating the need for padding during both encryption and decryption.[2]

2.2 Notation

In this text, E will denote a blockcipher (e.g., AES-128) of blocksize n bits, so that for each key K, E_K is a permutation on $\{0,1\}^n$ – the set of binary n-bit strings. For $s, s' \in \{0,1\}^n$, we will write $s\|s'$ and $s \oplus s'$, for the concatenation and XOR operations, respectively. A binary string with bits $a_{n-1}, \ldots, a_1, a_0$ corresponds to the element $[a_{n-1}x^{n-1} + \cdots + a_1 x + a_0]$ of the finite field $\mathbb{F}_{2^n}$. Thus, we can multiply binary strings by elements of $\mathbb{F}_{2^n}$ using shift and XOR operations (though it depends on the polynomial chosen to represent $\mathbb{F}_{2^n}$).

S will denote the total number of sectors on the device and J the number of n-bit blocks within a sector. Correspondingly, N will always denote a sector number and j the number of the n-bit block within the sector.

We will denote the unencrypted data (plaintext) at block j within sector N by $P_{N,j}$ and naturally extend this by concatenation:

$$P_N := P_{N,0}\|P_{N,1}\|\ldots\|P_{N,J-1}$$

will denote the unencrypted data at sector number N, while

$$P := P_0\|P_1\|\ldots\|P_{S-1}$$

will denote the unencrypted data of the whole device. Analogically, we denote the encrypted data by $C_{N,j} := E_K(P_{N,j}), C_n, C$, respectively.

[1] Modern storage devices are complex systems that can internally use different storage blocks for optimal performance, and can have specific requirements for optimal operations.

[2] The only legacy exception is the 520-byte sector (a 512-byte data sector with an additional 8-byte integrity field) on enterprise devices.

3 XTS encryption mode

The XTS mode – or rather XTS-AES – was first defined in Annex D.4.3 in [10] as an "instantiation" of Rogaway's XEX [19] with the underlying blockcipher being AES-128 or AES-256. Though not quite, as there are several differences:

- While XEX uses a single symmetric key to encrypt both the data and the sector number, XTS uses two different symmetric keys: K to encrypt the data, and K_T to encrypt the sector number.
- While XEX starts at $j = 1$, XTS starts at $j = 0$.
- XTS allows for ciphertext stealing to deal with encrypting data whose size is not a multiple of n. As explained in Section 2.1, we focus only on full sector encryption, so we can ignore this.[3]

3.1 XTS definition

We propose a definition that seeks a compromise between Rogaway's original XEX formality and the practical use case at hand.

Definition 1 (XTS encryption mode)**.** *Let K, K_T be symmetric keys for a blockcipher E of blocksize[4] 128 bits. Then the XTS mode of E encrypts the storage data in the following way:*

$$C_{N,j} := E_K(P_{N,j} \oplus T_{N,j}) \oplus T_{N,j},$$

$$C_N := C_{N,0} \| C_{N,1} \| \dots \| C_{N,J-1},$$

$$C := C_0 \| C_1 \| \dots \| C_{S-1},$$

where

$$T_{N,j} := E_{K_T}(N) \cdot \alpha^j$$

and α is the element $[x]$ in $\mathbb{F}_{2^{128}} := \mathbb{F}_2[x]/(x^{128} + x^7 + x^2 + x + 1)$.

Conceptually, the multiplication by α corresponds to a left shift with overflow. More explicitly, for a binary string $s = a_{127}a_{126} \dots a_1 a_0$ (this is how we denote concatenation of bits), we have

$$s \cdot \alpha = \begin{cases} a_{126}a_{125} \dots a_0 0 & \text{if } a_{127} = 0, \\ a_{126}a_{125} \dots a_0 0 \oplus \underbrace{0 \dots 0}_{120} 10000111 & \text{if } a_{127} = 1, \end{cases}$$

as x^{128} and $x^7 + x^2 + x + 1$ represent the same element in $\mathbb{F}_{2^{128}}$.

[3] XTS has been used in the Linux kernel since 2007. In contrast, ciphertext stealing was only added in 2019, and probably there is still no active in-kernel user.

[4] This is not essential, but multiplication by α differs for different block sizes.

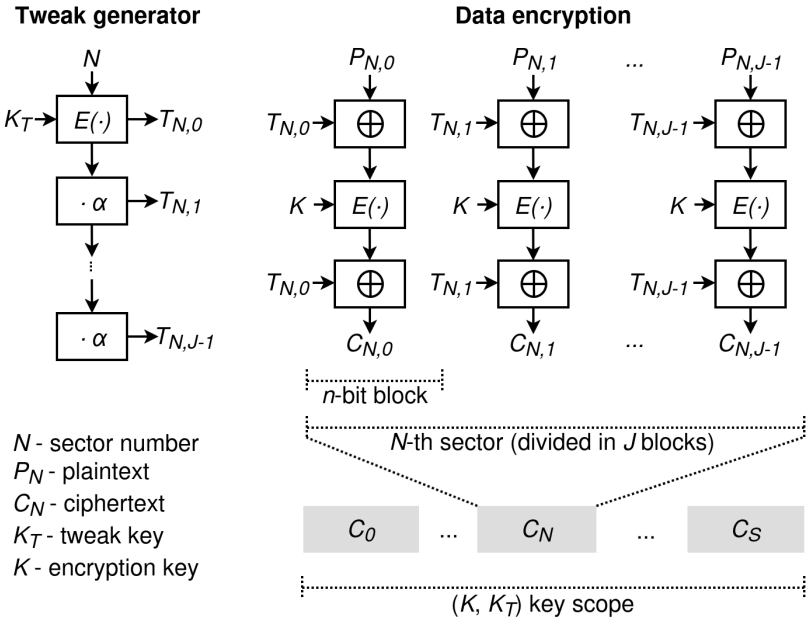

Figure 1: The principle of XTS sector encryption (without ciphertext stealing).

Example 1 (AES-XTS-128 encryption of one sector with two blocks)**.**
For

$$\begin{aligned} N &= 0x00000000000000000000000000000001, \\ K &= 0x11111111111111111111111111111111, \\ K_T &= 0x22222222222222222222222222222222, \\ P_{N,0} &= 0x44444444444444444444444444444444, \\ P_{N,1} &= 0x88888888888888888888888888888888, \end{aligned}$$

the tweak values are generated in the following way:

$$\begin{aligned} T_{N,0} &= E_{K_T}(N) = 0x6752ca5febca0f3fc8dc9dfc2a916295, \\ T_{N,1} &= T_{N,0} \cdot \alpha^1 = 0x49a494bfd6951f7e90b93bf95522c52a. \end{aligned}$$

The sector blocks are then encrypted as follows:

$$\begin{aligned} C_{N,0} &= E_K(P_{N,0} \oplus T_{N,0}) \oplus T_{N,0} \\ &= E_K(0x6752ca5febca0f3fc8dc9dfc2a916295) \oplus T_{N,0} \\ &= 0x13f084e65a7ca361f74957c9b11c7710 \oplus T_{N,0} \\ &= 0x74a24eb9b1b6ac5e3f95ca359b8d1585 \end{aligned}$$

$$\begin{aligned} C_{N,1} &= E_K(P_{N,1} \oplus T_{N,1}) \oplus T_{N,1} \\ &= E_K(0xc12c1c375e1d97f61831b371ddaa4da2) \oplus T_{N,1} \\ &= 0x2cada9d22ad34bf19a226c2c824f0364 \oplus T_{N,1} \\ &= 0x65093d6dfc46548f0a9b57d5d76dc64e \end{aligned}$$

3.2 Threat model

The simplest threat model for length-preserving sector encryption usually considers a "stolen device", allowing the attacker to access only one ciphertext snapshot at time. A typical example of such a threat model is the TCG Opal definition [8]: *"protect the confidentiality of stored user data against unauthorized access once it leaves the owner's control (following a power cycle and subsequent deauthentication)."* This model expects the user to recognize when the protected device "left the owner's control", which is not always feasible.

The real-world threat model also considers situations where the attacker can manipulate the ciphertext after repeatable access to the encrypted device, even performing some form of "traffic analysis". This is important, as encrypted images are often stored in cloud environments. Section D.4.3 in [12] suggests (in addition to introducing a key scope): *"The decision on the maximum amount of data to be encrypted with a single key should take into account the above calculations together with the practical implication of the described attack (e.g., ability of the adversary to modify plaintext of a specific block, where the position of this block may not be under adversary's control)."*.

Relatedly, we should consider adversaries capable of the chosen-ciphertext attack (CCA), i.e., able to query both the encryption and decryption oracle (with K, K_T still being fixed and secret) on inputs of their choice.

Despite having received a lot of criticism [20, 18], the XTS mode has not been replaced by any other mode more than a decade later. Unfortunately, its security goals and threat models are still not well-defined.

4 Security limits of XTS

4.1 Key scopes

The most drastic change in the IEEE 1619-2025 standard [12] is the following requirement: "*Maximum Number of 128-bit blocks in a key scope* $= 2^{36}$ *to* 2^{44}". For a fixed XTS key[5] (K, K_T), this translates to $S \cdot J \leq 2^{36}$ and $S \cdot J \leq 2^{44}$, respectively.

The reason for this comes from Annex D.4.2 in [12], which considers a CCA adversary with access to triples $(P_{N,j}, C_{N,j}, T_{N,j})$ and $(P_{N',j'}, C_{N',j'}, T_{N',j'})$ for some N, N', j, j', such that

$$P_{N,j} \oplus T_{N,j} = P_{N',j'} \oplus T_{N',j'}. \tag{1}$$

If such an adversary can overwrite $P_{N,j}$, they can use the resulting ciphertext to change $C_{N',j'}$ to a value that decrypts to any plaintext of their choice.

There is a non-negligible probability that (1) occurs when the number of plaintexts $(S \cdot J)$ encrypted under the same XTS key (K, K_T) approaches the birthday bound $2^{n/2}$.

[5] By an XTS key, we really mean a concatenated pair of symmetric keys; this is standard terminology.

Consequently, the scope of each key should be limited. The probability of (1) occurring for AES-{128,256} can be estimated by $\frac{(S \cdot J)^2}{2^{128}}$. Thus, encrypting 1 TiB of data (i.e., $S \cdot J = 2^{36}$) would lead to (1) occurring with probability roughly $2^{36 \cdot 2 - 128} = 2^{-56}$, whereas 256 TiB of data would correspond to roughly $2^{44 \cdot 2 - 128} = 2^{-40}$. This agrees with Annex D.4.3 in [10] up to a small constant.[6] However, it is unclear what amount of risk is acceptable in different contexts,[7] even though the change would invalidate compliance of many current encrypted storage devices.

4.2 Maximal sector size

Section 5.1 in [10] states: "*The number of 128-bit blocks within the data unit shall not exceed* 2^{20}*.*" As their data unit translates to our sector, this limits its size to 16 MiB, i.e., $J \leq 2^{20}$. This condition has already been present in [10] as a change from the initial version (2007), where it was only a suggestion ("*should not exceed*"). NIST [7] also strictly mandates this limit, but the rationale has never been explained. As XEX has been proven to be secure against chosen-ciphertext attacks even without this condition [19, 15], its purpose remains unclear. In practice, this is almost never an issue, as the typical sector size is at most 4 KiB, i.e., $J \leq 2^8$ (Section 2.1).

4.3 Distinct K, K_T

Annex C.I in [16] warns about a chosen ciphertext attack against XTS-AES, referring to Section 6 in Rogaway's original paper [19]: "*by obtaining the decryption of only one chosen ciphertext block in a given data sector, an adversary who does not know the key may be able to manipulate the ciphertext in that sector so that one or more plaintext blocks change to any desired value*". The attack crucially relies on two simultaneous conditions: $K = K_T$, and starting the indexation at $j = 0$ (where the tweak is unchanged, as it is multiplied by α^0). There are two obvious ways to prevent the attack:

- Starting from $j = 1$ (as XEX does); then the original security proof applies (Section 6 in [19], Section 4.2 in [15]).
- Requiring that $K \neq K_T$. FIPS 140-3 [16] made this option mandatory for compliance, even though as Liskov and Minematsu argue in [13], this comes from a misapplication of a security design practice (though does not lower the security in any way).

5 Discussion

The XTS encryption mode, as defined in IEEE 1619 [10], has been used for many years in many different systems. Thus, any compliance-breaking changes

[6] Annex D.4.3 in [10] mistakenly uses a general security guarantee $9.5\frac{q^2}{2^n}$ from [19] (valid for XEX, not XTS), but in fact, the factor 9.5 can be dropped if AES is secure [13, 3].

[7] We discussed this issue both in an IEEE SISWG meeting on April 25, 2025 and associated private communication, and got a confirmation that the analysis 4.1 is the one motivating the key scopes. The goal was to satisfy security targets set by NIST (i.e., 2^{-53} and 2^{-37}, respectively, should be good enough), but NIST never explained the choice of the specific constants, just like with the maximal sector size in Section 4.2.

– such as the mandatory use of key scopes in the IEEE 1619 standard [12] – could have a dramatic impact on the ecosystem. This would be amplified even more by the concept of FIPS certifications (which should impact the document [16]).

We are not necessarily arguing against the change; the issue described in Section 4.1 could definitely be a substantial one in some scenarios. However, to justify the increased complexity and potential for new implementation mistakes, it would be helpful to first have good answers to the following questions:

- What are the contexts with a weaker threat model? For example, in the "stolen" device model, where the attacker cannot actively modify the ciphertexts, adding key scopes would not improve security.
- What are the contexts with a stronger threat model? For example, if the attacker can store or extract snapshots of encrypted blocks outside the device, adding key scopes would not necessarily protect against the attack.
- Is there another way to strengthen XTS without the key scopes? One option would be using a blockcipher with larger n. As $n = 256$ would already be acceptable, and NIST already plans to propose a standard for "wide AES" ([17]), this option should be included in future considerations. The problem is that the XTS standard [10] has hardcoded 128-bit blocks everywhere and would need to be updated.
- How should the XTS keys used for key scopes be generated? It is unclear if all the keys used in key scopes must be generated using an approved random number generator, or if they can be derived from a master key with an approved key derivation algorithm. The second option would simplify key scope implementation.
- How to determine which XTS key to use for which sector? And how to handle device resizing? Standardizing (several variants of) answers to these might help with complexity reduction.

Currently, XTS-AES uses a single XTS key for the whole device (for all addressable sectors). Introducing key scopes can lead to incompatible variants of XTS-AES implementations until we specify which XTS key corresponds to a given sector. We propose two partial answers to this problem:

- Implement key scopes in a linear fashion: use each XTS key to encrypt the next (at most) 2^{44} plaintexts, then switch to the next key. A downside is that keys would not be utilized uniformly: if data is stored only on a small part of the device, the keys beyond the first one are never used. Also, for device resizing, keys would need to be added or removed dynamically.
- Implement key scopes with rotating XTS keys: use (N mod m)-th XTS key to encrypt the N-the sector, where m is the total number of XTS keys. This solution must define the maximal device size in advance (to not exceed the required key scope), but easily allows a dynamic resize (up to the maximal size). It also utilizes all keys more uniformly.

Beyond XTS mode

We would like to stress that XTS comes with serious downsides, just like other classical modes such as ECB, CBC, CFB, OFB, and CTR. Namely, they fail to provide full diffusion (i.e., each bit of the ciphertext depending on each bit of the plaintext) if the data to be encrypted exceeds a few blocks [6].

For a long-term use of a length-preserving mode (i.e., disqualifying authenticated encryption), it seems much more fruitful to switch to a wide encryption mode suited for disk encryption (as suggested in [20]). The feasible candidates might include:

- EME2 is based on EME, designed in 2003 and later standardized by IEEE 1619.2 [11]. The mode was never widely adopted, partially because of a questionable status of the related patent (now abandoned[8]).
- Adiantum [4] was developed by Google for low-end systems that do not provide AES hardware acceleration. It is designed to be fast and has low power requirements (suitable for mobile devices).
- HCTR2 [5] (also by Google) builds upon the CTR mode and seems like a performant and conservative option.
- *bbb-ddd-AES* [6] is a new mode built over the docked-double-decker construction [9]. If there is an upper bound on the number of times each tweak can be reused (sometimes satisfied by SSD hardware due to limited lifetime and wear-leveling), it can provide beyond-birthday bound security. The double-decker construction is a flexible framework that allows building wide encryption modes with better characteristics than the older modes mentioned above. As the proposed modes are very recent, there is no independent security analysis, and – unlike for the above candidates – no public reference implementation.

The pragmatic solution is perhaps to keep XTS limits in a sustainable state, as many legacy systems will be using it in the next years. For long-term improvement of disk encryption, we believe it would be worthwile to focus on a newer construction, like the mentioned double-decker framework.

We would be grateful for any feedback or suggestions that could contribute to the improvement of this work. The source code of this document is available on GitHub[9].

Acknowledgements

The authors were supported by the European Union under Grant Agreement No. 101087529 (CHESS – Cyber-security Excellence Hub in Estonia and South Moravia).

[8] https://lore.kernel.org/dm-crypt/kf5lug$8p0$1@ger.gmane.org/
[9] https://github.com/mbroz/xts-paper

References

[1] Cryptsetup authors. Cryptsetup project, 2025. https://gitlab.com/cryptsetup/cryptsetup.

[2] VeraCrypt authors. VeraCrypt disk encryption, 2025. https://veracrypt.io.

[3] Matthew V Ball, Cyril Guyot, James P Hughes, Luther Martin, and Landon Curt Noll. The XTS-AES disk encryption algorithm and the security of ciphertext stealing. *Cryptologia*, 36(1):70–79, 2012.

[4] Paul Crowley and Eric Biggers. Adiantum: length-preserving encryption for entry-level processors. *IACR Transactions on Symmetric Cryptology*, pages 39–61, 2018.

[5] Paul Crowley, Nathan Huckleberry, and Eric Biggers. Length-preserving encryption with HCTR2. Cryptology ePrint Archive, Paper 2021/1441, 2021. https://eprint.iacr.org/2021/1441.

[6] Christoph Dobraunig, Krystian Matusiewicz, Bart Mennink, and Alexander Tereschenko. Efficient instances of docked double decker with AES, and application to authenticated encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 62–92. Springer, 2025.

[7] Morris J. Dworkin. SP 800-38E. Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices. Technical report, National Institute of Standards & Technology, Gaithersburg, MD, United States, 2010. Available at https://csrc.nist.gov/pubs/sp/800/38/e/final.

[8] Trusted Computing Group. Storage Security Subsystem Class: Opal, 2025. Available at https://trustedcomputinggroup.org/resource/storage-work-group-storage-security-subsystem-class-opal/.

[9] Aldo Gunsing, Joan Daemen, and Bart Mennink. Deck-based wide block cipher modes and an exposition of the blinded keyed hashing model. *IACR Transactions on Symmetric Cryptology*, pages 1–22, 2019.

[10] IEEE. IEEE 1619-2018 – Standard for Cryptographic Protection of Data on Block-Oriented Storage Devices, 2018.

[11] IEEE. IEEE 1619.2-2021 Standard for Wide-Block Encryption for Shared Storage Media, 2021.

[12] IEEE. IEEE 1619-2025 - Standard for Cryptographic Protection of Data on Block-Oriented Storage Devices, 2025.

[13] Moses Liskov and Kazuhiko Minematsu. Comments on XTS-AES. *Comments to NIST, available from their web page*, 2008. Available at https://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/XTS/XTS_comments-Liskov_Minematsu.pdf.

[14] Microsoft. BitLocker overview, 2025. https://learn.microsoft.com/en-us/windows/security/operating-system-security/data-protection/bitlocker/.

[15] Kazuhiko Minematsu. Improved security analysis of XEX and LRW modes. In *International Workshop on Selected Areas in Cryptography*, pages 96–113. Springer, 2006.

[16] NIST. Implementation Guidance for FIPS 140-3 and the Cryptographic Module Validation Program, 2024. Available at https://csrc.nist.gov/csrc/media/Projects/cryptographic-module-validation-program/documents/fips%20140-3/FIPS%20140-3%20IG.pdf.

[17] NIST. PRE-DRAFT Call for Comments: NIST Proposes to Standardize a Wider Variant of AES, 2024. Available at https://csrc.nist.gov/pubs/sp/800/197/iprd.

[18] Thomas Ptacek and Erin Ptacek. You Don't Want XTS, 2014. Available at https://sockpuppet.org/blog/2014/04/30/you-dont-want-xts/.

[19] Phillip Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 16–31. Springer, 2004.

[20] Phillip Rogaway. Evaluation of some blockcipher modes of operation, 2011. Available at https://www.cs.ucdavis.edu/~rogaway/papers/modes-cryptrec.pdf.