NIST Internal Report
NIST IR 8459

Report on the Block Cipher Modes of Operation in the NIST SP 800-38 Series

Nicky Mouha
Morris Dworkin

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8459

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
U.S. DEPARTMENT OF COMMERCE

NIST Internal Report
NIST IR 8459

Report on the Block Cipher Modes of Operation in the NIST SP 800-38 Series

Nicky Mouha
Strativia

Morris Dworkin
Computer Security Division
Information Technology Laboratory

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8459

September 2024

U.S. Department of Commerce
Gina M. Raimondo, Secretary

National Institute of Standards and Technology
Laurie E. Locascio, NIST Director and Under Secretary of Commerce for Standards and Technology

Certain commercial equipment, instruments, software, or materials, commercial or non-commercial, are identified in this paper in order to specify the experimental procedure adequately. Such identification does not imply recommendation or endorsement of any product or service by NIST, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

There may be references in this publication to other publications currently under development by NIST in accordance with its assigned statutory responsibilities. The information in this publication, including concepts and methodologies, may be used by federal agencies even before the completion of such companion publications. Thus, until each publication is completed, current requirements, guidelines, and procedures, where they exist, remain operative. For planning and transition purposes, federal agencies may wish to closely follow the development of these new publications by NIST.

Organizations are encouraged to review all draft publications during public comment periods and provide feedback to NIST. Many NIST cybersecurity publications, other than the ones noted above, are available at https://csrc.nist.gov/publications.

Publication History
Approved by the NIST Editorial Review Board on 2024-01-03

How to Cite this NIST Technical Series Publication:
Mouha N, Dworkin M (2024) Report on the Block Cipher Modes of Operation in the NIST SP 800-38 Series. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Interagency or Internal Report (IR) NIST IR 8459. https://doi.org/10.6028/NIST.IR.8459

Author ORCID iDs
Nicky Mouha: 0000-0001-8861-782X
Morris Dworkin: 0000-0003-2745-9906

Contact Information
cryptopubreviewboard@nist.gov

National Institute of Standards and Technology
Attn: Computer Security Division, Information Technology Laboratory
100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930

Additional Information
Additional information about this publication is available at https://csrc.nist.gov/pubs/ir/8459/final, including related content, potential updates, and document history.

All comments are subject to release under the Freedom of Information Act (FOIA).

Abstract

This report focuses on the NIST-recommended block cipher modes of operation specified in NIST Special Publications (SP) 800-38A through 800-38F. The goal is to provide a concise survey of relevant research results about the algorithms and their implementations. Based on these findings, the report concludes with a set of recommendations to improve the corresponding standards.

Keywords

AES; block cipher; cryptography; mode of operation; SP 800-38; standardization; Triple-DES.

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analyses to advance the development and productive use of information technology. ITL's responsibilities include the development of management, administrative, technical, and physical standards and guidelines for the cost-effective security and privacy of other than national security-related information in federal information systems.

Table of Contents

Acknowledgments

The authors would like to thank Elaine Barker, Michael Davidson, Jim Foti, Jinkeon Kang, Stefan Lucks, Yu Sasaki, Meltem Sönmez Turan, Isabel Van Wyk, and their NIST colleagues for providing valuable feedback during the development of this publication.

1. Introduction

NIST Interagency or Internal Report (NIST IR) 8319 [55] focuses on the Advanced Encryption Standard (AES) [22] that was standardized in FIPS 197 [58]. AES is one of only two block ciphers that are currently standardized by NIST. The other is the Triple Data Encryption Algorithm (TDEA) [10], also known as Triple-DES (Data Encryption Standard). Triple-DES is deprecated and will be disallowed after 2023 [11].

A block cipher can only process inputs of a specific length, known as the block size. AES has a block size of 128 bits, and Triple-DES has a block size of 64 bits. To process inputs of other lengths, it is necessary to use a mode of operation. A mode of operation can process shorter or larger inputs by performing one or more calls to an underlying block cipher.

In fact, it is possible to claim that a block cipher is always used in combination with a mode of operation: using a block cipher "directly" is equivalent to using it in the Electronic Codebook (ECB) mode with the restriction that the input must be exactly one block in length (e.g., 128 bits in the case of AES). Therefore, the real-world applications of the NIST-recommended modes of operation overlap with the applications of the underlying block cipher and include virtually all web browsers, Wi-Fi and cellular devices, and contact and contactless chip cards, as described in NIST IR 8319 [55].

Therefore, a logical next step after NIST IR 8319 is to analyze the recommended modes of operation. This report analyzes the block cipher modes of operation that are standardized in NIST Special Publication (SP) 800-38A through 800-38F. More specifically:

- NIST SP 800-38A [25] defines the Electronic Codebook (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB), and Counter (CTR) modes, which will be referred to collectively as the "five confidentiality modes."
- The Addendum to NIST SP 800-38A [26] defines three variants of the CBC mode: the CBC-CS1, CBC-CS2, and CBC-CS3 modes, where "CS" indicates "ciphertext stealing."
- NIST SP 800-38B [27] defines the Cipher-based Message Authentication Code (CMAC) mode.
- NIST SP 800-38C [28] defines the Counter with Cipher Block Chaining-Message Authentication Code (CCM) mode.
- NIST SP 800-38D [29] defines the Galois/Counter Mode (GCM) and its specialization GMAC to generate a Message Authentication Code (MAC).
- NIST SP 800-38E [30] defines the XTS-AES mode, where XTS stands for "XEX Tweakable block cipher with ciphertext Stealing," and XEX stands for "eXclusive-or Encrypt eXclusive-or."
- NIST SP 800-38F [31] defines the AES Key Wrap (KW) mode, the AES Key Wrap with Padding (KWP) mode, and the TDEA Key Wrap (TKW) mode.

Note that NIST SP 800-38G [32], which defines the Format-Preserving Encryption (FPE) modes FF1 and FF3, is currently undergoing a revision that is proposed in Draft NIST SP 800-38G, Rev. 1 [33].

2. Scope

The Crypto Publication Review Board identifies standards and guidelines to be reviewed in order to reaffirm, update, revise, convert, or withdraw the standard or guideline.[1] As explained in NIST IR 8319 [55], the review process presents an opportunity to retrospectively examine the development of the standard or guideline under review with a focus on the technical properties that are likely to be much better understood in the years since the standard or guideline was finalized. This report will mainly focus on the technical merits, although it is important to also provide editorial comments to determine whether the standards or guidelines are correct, complete, consistent, and unambiguous.

NIST IR 8319 discussed the scope of a review, which needs to be broad enough to capture possible issues with the standard or guideline but narrow enough to provide concrete recommendations within a reasonable amount of time. More specifically, reviews focus on the resources available on the NIST website related to the standard or guideline combined with academic publications and other publicly available resources to understand the security properties of a cryptographic algorithm or implementation.

By analyzing how the resources available on the NIST website related to the standard or guideline are organized, the implicit and explicit guarantees and assumptions between different standards and guidelines can become apparent. The instantiation of the block cipher modes of operation requires an approved block cipher, such as AES; therefore, the report of the block cipher modes of operation can be based on technical assumptions that were evaluated in NIST IR 8319. For example, NIST IR 8319 explored whether the AES block cipher behaves as a "pseudo-random permutation" (PRP), so that the block cipher modes of operation can rely on this assumption.

NIST IR 8319 already mentioned that certain properties, such as the impact of the 128-bit block size of AES, will be analyzed for the modes of operation. This report will also analyze whether certain security requirements of AES implementations (e.g., resistance against side-channel attacks) are still true when they are used in a particular mode of operation or give an indication of new problems that may appear when modes of operation are used in an application.

The failure of the security properties of the block cipher modes and their implementations leading to attacks will be investigated, with a particular focus on attacks that lead to practical insecurities that are listed in the NIST National Vulnerability Database (NVD), such as the Sweet32 attack [16] that is listed in the NVD as CVE-2016-2183 [67]. In order to avoid inadvertently overlooking security problems, internal NIST feedback and public comments will be taken into consideration. As such, this report has the intention of complementing rather than supplementing other reports on the NIST-recommended modes of operation, such as the report by Rogaway [74].

For historical reasons, not all NIST-recommended modes were designed according to a predefined list of security requirements. An attempt to retrospectively formalize a set of security requirements would be difficult. Instead, the question raised by Richard Barnes at Real

[1] See https://csrc.nist.gov/Projects/crypto-publication-review-project.

World Crypto 2021 is addressed, which is whether “primitives do what people expect them to do.”[2]

It turns out that this is often not the case and that many users expect that the modes of operation have similar properties to the block cipher, except that they support wider inputs. This is, for example, evidenced by numerous incorrect statements in online forums that a “damaged” ciphertext leads to an unrecoverable (or largely unrecoverable) plaintext. Therefore, a focus of this report is to investigate the gap between the required and actual properties of the NIST-recommended modes of operation in the applications where they are used.

[2] This comment was made on the Zulip chat of the conference, which seems to be no longer in service so that no reference can be provided.

3. Modes of Operation

There are many different modes of operation to support a wide range of applications. The modes are intended for processing inputs of some specific length(s) by means of repeated calls to a block cipher. The block cipher can only process inputs of a certain fixed length, known as the block size. Recall that the block size of AES is 128 bits, whereas Triple-DES has a block size of 64 bits.

The block cipher requires a key to operate. For example, in the case of AES, the length of the key is either 128, 192, or 256 bits. Therefore, the mode of operation will also require a key. The mode will either specify how to generate the keys for the block cipher or just pass the key to the block cipher in an unmodified way. The mode of operation can use the forward block cipher operation or the inverse block cipher operation. If the mode of operation is invertible, it is possible to refer to the forward mode operation as encryption and the inverse mode operation as decryption. There are applications where the mode of operation is only intended to be used in the forward direction, such as when the mode of operation is used to compute an (authentication) tag to verify the integrity (or authenticity) of a message.

The security properties that an implementation of a mode of operation must satisfy depend on the specific mode of operation and on the application where it is used. In what follows, the security properties that are required will be analyzed, and explanations will be provided if a failure of the security properties could lead to attacks.

4. NIST SP 800-38A: Five Confidentiality Modes

NIST SP 800-38A [25] defines five confidentiality modes: ECB, CBC, CFB, OFB, and CTR. Three variants of the CBC mode are defined in the Addendum to NIST SP 800-38A [26]: CBC-CS1, CBC-CS2, and CBC-CS3. Except for the ECB and CTR modes, an Initialization Vector (IV) with a length equal to the block size is required by each of these modes in addition to the variable-length plaintext and the key. In the case of the ECB mode, there is no IV input. CTR is defined with a sequence of unique counter blocks as an additional input.

Although NIST SP 800-38A does not explicitly define the security requirements of a confidentiality mode, it does provide examples of attacks where confidentiality is compromised. In the attack examples, the adversary knows certain plaintext blocks and uses the ciphertext to derive other plaintext blocks that were not known to the adversary.

This notion is consistent with cryptographic literature, where the security requirement for confidentiality assumes that the adversary knows and can even choose (parts of) the plaintext and observe the corresponding ciphertext [14]. For an example of a practical attack where the adversary has this power, see the BEAST (Browser Exploit Against SSL/TLS) attack on the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols, known in the NVD as CVE-2011-3389 [62].

The OFB and CTR modes in NIST SP 800-38A are not intended to provide any notion of integrity, meaning that they are not secure against adversaries who can modify the ciphertexts. For example, the OFB and CTR modes process every bit of the plaintext separately, so an adversary can make arbitrary changes to bits in the plaintext by flipping the corresponding bits in the ciphertext. This property is explained in Appendix D of NIST SP 800-38A. The same “bit flipping attack” also applies to the IV of the CBC mode. Indeed, Appendix D of NIST SP 800-38A clarifies that the integrity of the IV used for the CBC mode needs to be protected for this reason.

Historically, it was assumed that the CBC and CFB modes *do* provide some notion of integrity because bit errors in the ciphertext will lead to random errors in at least one block of the plaintext.[3] However, the statement in Appendix D of NIST SP 800-38A that “the existence of such bit errors may be detected by their randomizing effect on the decryption of the succeeding ciphertext segment” can be misunderstood. More specifically, some integrity protection mechanism against random errors in the plaintext was thought to be sufficient, such as introducing a cyclic redundancy check (CRC) to the plaintext. This is not the case, however, as shown by an attack on the CBC and CFB modes as used in version 1.5 of the Secure Shell (SSH) protocol [37], known as CVE-1999-1085 [59]. Note that a patch was introduced to try to detect an attack that exploited this vulnerability; however, the patch introduced a vulnerability that could lead to arbitrary code execution: CVE-2001-0144 [60].

Unfortunately, a proof of impossibility by Jutla [47] shows that there will always exist an attack on the integrity mechanism when the plaintext is expanded with a constant amount if the mode of operation consists of only block cipher calls and XOR operations, which includes all modes of operation in NIST SP 800-38A. Therefore, it is incorrect to assume that changes in the ciphertext

[3] CFB does not process the data in blocks but rather in segments that may be shorter than the block size. In this case, at least one segment of the plaintext will contain random errors when there are bit errors in the ciphertext.

for the modes in NIST SP 800-38A will result in detectable changes in the plaintext, and they should not be used without a cryptographic mechanism to provide integrity, such as a Message Authentication Code (MAC) algorithm.

Even when a confidentiality mode of NIST SP 800-38A is used in combination with a secure cryptographic MAC algorithm, the application can still be vulnerable to practical attacks. For example, SSL 3.0, TLS 1.0, and other commonly used protocols use a combination of CBC and a MAC algorithm that is vulnerable to padding oracle attacks [19][81], known as CVE-2003-0078 [61]. TLS 1.1 and 1.2 include countermeasures against padding oracle attacks, but the Lucky13 attack [1] shows how a timing attack can exploit a padding oracle in implementations that were thought to have been fixed. It turns out that addressing the Lucky13 attacks is highly non-trivial, as explained by Almeida et al. [2]. A fix for the Lucky13 attack in OpenSSL even introduced a new vulnerability (CVE-2016-2107) [66].

Starting with TLS 1.3, a decision was made to deprecate all modes of operation that only provided confidentiality protection. The only modes of operation that remain offer the combined functionality of confidentiality and integrity and are known as Authenticated Encryption with Associated Data (AEAD) modes. The AEAD modes that are supported in TLS 1.3 include the two AEAD modes that NIST has standardized: CCM and GCM, which are standardized in NIST SP 800-38C and NIST SP 800-38D, respectively.

In what follows, the first paragraph of this section will be revisited. Recall that except for the ECB and CTR modes, all modes of operation in NIST SP 800-38A require an IV. The requirements for the IV will be discussed, and the implications of the lack of an IV for the ECB and CTR modes will be explained. Since the modes of operation support variable-length plaintexts, the impact of the length of the plaintext and the block size will be discussed.

4.1. Initialization Vector (IV)

The ECB mode does not use an IV. It processes every block of the plaintext independently, and therefore, if a plaintext block occurs more than once, the corresponding ciphertext blocks will be the same. NIST 800-38A notes that “if this property is undesirable in a particular application, the ECB mode should not be used.” If no integrity mechanism is used along with the ECB mode, an adversary can also maliciously substitute ciphertext blocks without affecting the decryption of adjacent ciphertext blocks.

In practice, application developers may not have a fully informed opinion on the risks of using the ECB mode. Because there is no IV input, ECB is the easiest (and most efficient) mode to implement. Perhaps not surprisingly, there are many entries in the NVD of applications that are vulnerable due to the use of the ECB mode. Examples of such applications include Bouncy Castle (CVE-2016-1000344) [68], Jenkins (CVE-2017-2598) [69], and Zoom (CVE-2020-11500) [71].

NIST SP 800-57, Part 1 Rev. 4 [7], stated that the ECB mode is not recommended for general use, but this statement was removed in the next revision [8]. Even if ECB is not recommended for general use, it may be suitable for some specific use cases.

Rogaway [74] points out that the ECB mode can provide confidentiality when the plaintext is uniformly random. An example of a uniformly random plaintext could be a cryptographic key that needs to be encrypted with another key. However, a better solution for this "key wrapping" problem would be to ensure both the confidentiality and integrity of the key. This scenario will be discussed again later when analyzing the modes of operation for key wrapping defined in NIST SP 800-38F.

There are, nevertheless, some specific applications where the ECB mode is used securely in another NIST standard:

- Appendix A.1 of NIST SP 800-73-4, Part 2 [20], provides an example of how the ECB mode is used in a challenge-response protocol. The goal is to validate that a Personal Identity Verification (PIV) card is authentic (i.e., not a counterfeit card) by providing the card with a challenge that is encrypted using a block cipher in the ECB mode. The application should ensure that challenges are not used more than once so that the key is required to compute the response.
- In Section 4.2.2 of NIST SP 800-73-4, Part 2 [20], the ECB mode is used on a counter to generate the IV for CBC encryption. The use of ECB to generate an IV for the CBC mode is not secure in general, as will be discussed later. However, this specific implementation of the counter leads to a mode of operation that can be proven to be secure based on the security of the underlying block cipher.[4]

Note that these applications of ECB are degenerate in two ways: the plaintext is not secret (so the goal is not to ensure the confidentiality of the plaintext), and the length of the plaintext is only one block (whereas modes of operation are intended to support variable-length plaintexts).

This is different from insecure applications that use ECB for encrypting secrets (e.g., CVE-2017-2598 [69]), and it may make more sense to interpret these use cases as a Message Authentication Code (MAC) algorithm computed on the input. In fact, the use of ECB on a one-block message is identical to the CBC-MAC defined in FIPS 81 [57]. Note that CBC-MAC was withdrawn for general use, as it is insecure for variable-length messages (as explained in the CMAC recommendation [27]). However, the CBC-MAC construction continues to be a component of approved cryptographic techniques, such as CCM [28], FF1 [32], and CTR_DRBG [9] (under the name "BCC"), as well as a conditioning component in NIST SP 800-90B [79].

The CTR mode is not defined with an IV in NIST SP 800-38A but with a sequence of unique counter blocks. For a well-defined CTR mode, the number of counter blocks should be at least as long as the number of blocks in the plaintext (for the encryption operation) or ciphertext (for the decryption operation). Appendix B.1 of NIST SP 800-38A suggests using an incrementing function to generate the counter blocks so that all counter blocks can be generated

[4] A game-based security proof was worked out but not included in this report.

deterministically from the initial counter block. Appendix B.2 in NIST SP 800-38A provides two suggestions to generate the counter blocks:

1. The first approach corresponds to an encryption mode that keeps track of the number of blocks encrypted so far under the given key. The initial counter block can have any value, and the incrementing function can be applied to a subset of the initial counter block. For this encryption mode, the initial counter block can be fixed to the all-zero block without any loss of security so that the encryption mode does not need an additional input (no IV nor sequence of counter blocks). Also, without any loss of security, the incrementing function can be applied to the entire initial counter block, not just to a subset of the bits in the block.
2. The second approach corresponds to an encryption mode that reserves half of the initial counter block for a unique IV, whereas the incrementing function is applied to the other half for every block of the message. Appendix B.2 in NIST SP 800-38A points out that this restricts the maximum length of the plaintext to 2^{64} blocks in the case of AES due to its block size of 128 bits. Moreover, in this case, if the IVs are randomly generated for about 2^{32} plaintexts, then some IV is expected to repeat due to the birthday paradox.

The CTR mode allows other methods to generate the counter blocks as long as the incrementing function has a sufficiently long period. This definition is arguably broad enough to consider the OFB mode as a special case of the CTR mode, where the incrementing function is the block cipher using the given key. Rogaway [74] is correct to point out that the uniqueness of the IVs is not sufficient for the OFB mode; it should also be ensured that the previous outputs of the incrementing function cannot be used as the IV. However, Rogaway [74] is not correct to suggest that the CTR mode was not included in the original four modes of FIPS 81 [57] due to a "(largely antiquated) belief that a confidentiality mode should provide some sort of non-malleability guarantee"; otherwise OFB would not have been one of the original four modes in FIPS 81.

Lipmaa et al. [51] argued for the inclusion of the CTR mode in NIST SP 800-38A and stated that "a well-designed standard for the CTR mode should not be overly prescriptive about how [the counter blocks are] formed." However, Rogaway does point out in Section 5.6.6 of [74] that the CTR specification may be too open-ended, which makes it more likely that an implementation of the CTR mode may be provided with a non-unique sequence of counter blocks. Rogaway then points out that compliance testing may suffer as well. Indeed, NIST's Advanced Encryption Standard Algorithm Validation Suite (AESAVS) document [12] states that "it is not possible to implement an automated test for [the CTR] mode" and requires a manual review of the design and implementation of the CTR mode in addition to performing the automated tests for the ECB mode.

The reuse of one of the counter blocks for the CTR mode leads to a devastating confidentiality attack, as explained in Appendix B of NIST SP 800-38A. The attack is similar to the reuse of a one-time pad (OTP). An adversary who knows certain plaintext blocks can use the ciphertext to derive all other plaintext blocks at locations where the counter blocks are reused by means of a simple XOR (Exclusive Or) operation.

NIST SP 800-38A states that, whereas the IV for OFB needs to be unique, the IV for the CBC and CFB modes needs to be "unpredictable." The standard clarifies that "it must not be possible to predict the IV that will be associated with the plaintext in advance of the generation of the IV."[5] Two methods to generate unpredictable IVs are suggested: encrypting a nonce (a value that is only used once) or generating the IV using an approved random number generator. This seems to leave open the possibility that the IV for the CBC and CFB modes is randomly generated once, kept secret, but reused for every plaintext. The consequence of IV reuse is a loss of confidentiality: the ciphertext will reveal not only when a plaintext repeats but even whether the first block or blocks of the plaintext are equal. Confidentiality can also fail in other ways, as shown by the following two examples:

- The first block of the CFB mode is generated in the same way as the OFB and CTR modes, so the confidentiality attack in Section 6.4 of NIST SP 800-38A applies to the first block.
- If the CBC mode is used with the key as the IV, then the IV (and therefore the key) can be recovered with a chosen-plaintext attack. This attack is described in Section 7.6 of [80].

Lastly, the suggestion to encrypt a nonce for the CBC and CFB modes is not secure, as explained in Section 4.6 by Rogaway [74].

4.2. Plaintext Length

The modes of operation specified in NIST SP 800-38A support variable-length inputs, and the length of the input impacts the security of the algorithm and its implementation.

For example, if the length of the ciphertext is the same as the length of the corresponding plaintext, then an adversary may distinguish the encryptions of "cat" and "fish" just by observing the ciphertext lengths. It can be argued that this property is somewhat inevitable. Tezcan and Vaudenay [77] state that: "Practically, we can always distinguish an encrypted [text] message from an encrypted [high definition] video stream." However, in a real-world application, a meaningful notion of confidentiality may not be achieved if it is possible for an adversary to guess which text message or video stream was encrypted based on the ciphertext length. Perhaps even more worryingly, when passwords or cookies combined with adversary-chosen plaintexts are compressed before encryption, the length of the ciphertext can allow for the recovery of the password or cookie. This is the attack vector that was used in the CRIME (Compression Ratio Info-leak Made Easy) (CVE-2012-4929) [63] and BREACH (Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext) (CVE-2013-3587) [64] attacks.

In NIST SP 800-38G [32], NIST recognizes that short ciphertexts are especially vulnerable to plaintext guessing given that the domain will be small as well. Moreover, short ciphertexts indicate that the conventional countermeasure to hide the length of the plaintext (i.e., padding

[5] This seems to imply that the previous ciphertext's last block is unsuitable as an IV because it would be known before the current encryption. This was nevertheless how CBC was implemented in the TLS 1.0 protocol and was exploited by the BEAST attack (CVE-2011-3389) [62]. Countermeasures against this attack were introduced in TLS 1.1.

the plaintext before encryption) has not been applied. Therefore, NIST SP 800-38G [32] calls for a minimum domain of one hundred possible values, which was increased to one million in Draft NIST SP 800-38G, Rev. 1 [33]. For binary inputs, this corresponds to a domain of at least 20 bits. It is possible to encrypt a one-bit plaintext for the OFB mode, CTR mode, and CFB mode with a one-bit segment size. To make plaintext guessing more difficult, an increase in the minimum size of the plaintext should be considered.

NIST SP 800-38A does not allow for the encryption of a zero-length plaintext for any of the modes. For the ECB, CBC, and CFB modes, the plaintext must be at least one block (or one segment in the case of CFB). NIST SP 800-38A clarifies that the plaintext must be padded to enforce this minimum. NIST's AESAVS [12] only provides test vectors that are a nonzero multiple of the block or segment size. The addition of manual reviews or tests to ensure that certain plaintexts (e.g., the zero-length plaintext) are rejected or cannot occur should be considered.

NIST provides variants of CBC for plaintexts that are not a multiple of the block size: the "ciphertext stealing" variants CS1, CS2, and CS3 that are standardized in the SP 800-38A Addendum. However, these variants cannot be used for very small plaintexts since they require that the length of the plaintext is at least equal to the block size.

4.3. Block Size

As mentioned in NIST IR 8319 [55], the impact of the 128-bit block size of AES will be discussed here in the context of the NIST-recommended modes of operation. NIST IR 8319 pointed out how the Sweet32 attack [16] shows the practical insecurity of block ciphers, such as Triple-DES, with a 64-bit block size. This section will explain how the block size impacts the security of the mode (however, note that the block size may also have a functional impact):

- As stated in Section 4.1, the ECB and CBC modes of operation require that the length of the plaintext is a multiple of the block size. The three "ciphertext stealing" variants of CBC do not have this requirement; however, they do require that the length of the plaintext is at least equal to the block size.
- For efficiency reasons, an application may want the input to fit within one block (e.g., the two ECB examples of NIST SP 800-73-4 [20] mentioned in Section 4.1).

The impact of the block size may be best understood using a typical security proof for a mode of operation. First, two technical terms need to be briefly introduced:

1. As mentioned in Section 4, the attack examples in NIST SP 800-38A assume that the adversary knows certain plaintext blocks and uses the ciphertext to derive some other plaintext blocks. As explained in Section 5.6 of [14], the notion of "semantic security" captures this idea. Block cipher security proofs use an easier notion called "chosen plaintext security," where the adversary chooses a sequence of pairs of plaintexts and receives a sequence of ciphertexts that correspond to either always the "left" or always the "right" plaintext of each pair. In the end, the adversary needs to "guess" whether the ciphertexts correspond to the "left" or the "right" plaintext. (To further empower

the adversary, the plaintext pairs can be chosen adaptively based on previous answers.) Thus, "semantic security" implies "chosen plaintext security," and both security notions have been shown to be equivalent (see Section 5.6 of [14]).

2. When the adversary makes its "guess," it is assumed a priori that both outcomes are equally likely and that an *advantage* (Adv) is associated with the adversary's guess. An advantage of zero means that the guess is no better than flipping a fair coin, whereas an advantage of one means that the adversary's guess is always correct.

With this, Theorem 5.19 of [14] proves the following security result for CBC with a randomly chosen IV:

$$\mathbf{Adv}_{\mathrm{SE}}^{\mathrm{IND-CPA}}(A) \leq \mathbf{Adv}_{E}^{\mathrm{PRF}}(B) + \frac{\sigma^2}{2^n}$$

Here, *A* is an adversary that attacks the "left-or-right indistinguishability under a chosen-plaintext attack" (IND-CPA) of the CBC symmetric encryption (SE) scheme, querying at most σ *n*-bit blocks in total, and *B* is an adversary that attacks the pseudo-random function (PRF) security of the block cipher *E*, making at most σ block cipher queries.[6] If the block cipher *E* is instantiated by AES, then the block size in bits is $n = 128$. NIST IR 8319 [55] studied the pseudo-random permutation (PRP) security of AES, which is related to the PRF security by means of the PRP/PRF Switching Lemma [13].

Therefore, the PRP security of AES suffices to prove the security of the CBC mode. This means that as long as AES is secure (as a PRP), any attacks on the CBC mode of operation are ruled out. However, the security result only holds up to a certain amount of data that is processed under one key. Indeed, after encrypting about $\sigma = 2^{n/2}$ 128-bit blocks or 256 EiB (exbibytes) of data, $\sigma^2/2^n = 1$. In this case, the above inequality reduces to the trivial statement that the advantage of *A* is at most one, notwithstanding the PRP security of AES. In other words, "all bets are off" regarding the security of CBC after encrypting $2^{n/2}$ blocks of data.

In fact, after about $2^{n/2}$ blocks of data are processed under one key, a collision attack on the CBC mode of operation will succeed with high probability. Collision attacks are also known as birthday attacks, a term that refers to the birthday paradox, which states that in a room of 23 people, the probability that two people have the same birthday is more than 50 %. An example of a collision attack on CBC is the Sweet32 attack [16], which shows the practical insecurity of 64-bit block ciphers such as Triple-DES in commonly used protocols such as TLS.

It would be incorrect to assume that it is safe to process close to $2^{n/2}$ blocks of data under one key. The CBC security result cannot preclude that an attack with a non-negligible advantage exists; moreover, Bhargavan and Leurent [16] caution in their Sweet32 paper that the key must be changed *well before* reaching $2^{n/2}$ blocks of data. Otherwise, the adversary may simply repeat the attack for several TLS sessions to obtain a high probability that the attack is successful for at least one session.

[6] For adversaries with limited running time, Theorem 5.19 of [14] also provides upper bounds on the running times of A and B.

Limitations of the amount of data that can be encrypted per key are considered in several NIST standards. For example, in NIST SP 800-67, Rev. 2 [10], the amount of data that may be encrypted under one key for Triple-DES is limited to 2^{20} 64-bit blocks or 8 MiB (mebibytes). Similarly, NIST SP 800-38B (CMAC) [27] imposes a limit of 2^{48} 128-bit blocks or 4 PiB (pebibytes) for AES and 2^{21} 64-bit blocks (16 MiB) for Triple-DES, corresponding to collision probabilities of less than one in a billion and less than one in a million, respectively. Interestingly, although NIST provides general guidance about key lengths in [11], it does not seem to provide general guidance about the acceptable success probabilities for cryptographic attacks.

In the case of AES, whether it is realistic to encrypt 4 PiB of data under the same key depends on the application. For example, the limit may be reached for large storage devices or for high-speed connections: for example, if data is encrypted at a rate of 400 gigabit per second, the per-key limit will be reached after about 25 hours. Overcoming this per-key limit would require a "beyond birthday bound" mode of operation or a wider block cipher (e.g., a block cipher with a block size of 256 bits).

The explanation so far has only focused on the CBC mode. However, similar security results and attacks can be obtained for the CFB, OFB, and CTR modes; for each of these modes, the key must be changed *well before* encrypting $2^{n/2}$ blocks of data. For example, an attack similar to the Sweet32 attack on CBC can also be applied to the CTR mode [50]. In the case of the ECB mode, a trivial attack against semantic security can be mounted by encrypting only two blocks of data [14].

The analysis so far assumes classical adversaries. For NIST-recommended modes of operation against quantum adversaries conducting queries under superposition, Anand et al. [3] show that CBC, CFB, OFB, and CTR are secure under certain assumptions.

5. NIST SP 800-38B: The CMAC Mode for Authentication

NIST SP 800-38B [27] defines the CMAC mode of operation. CMAC takes a variable-length message of zero or more bits as input and outputs a fixed-length bit string *T*, referred to as the tag or the MAC. The goal of CMAC is to provide an acceptable level of assurance about the integrity of the message. That is, it should be computationally difficult to construct a *forgery*, which corresponds to constructing a valid tag for a message that has not been queried before.

Note that the input is referred to as the "message" rather than the "plaintext." It is possible that the message will be sent in the clear or that the message corresponds to a ciphertext (e.g., in an "Encrypt-then-MAC" construction for authenticated encryption). The goal of CMAC is not to "hide" the contents of the message, so there is no need for an IV input to ensure that the same message leads to different outputs.

CMAC is a variant of CBC-MAC, which refers to using CBC with an all-zero IV and returning the last ciphertext block as the tag. Consequently, the security properties of CMAC are similar to those of CBC, in that CMAC comes with a provable security result and also requires that the key must be changed *well before* processing $2^{n/2}$ blocks of data (see Section 4.3).[7] However, compared to CBC, the CMAC security result contains an additional term of 2^{-t}, where *t* is the length of the tag in bits.

As explained in NIST SP 800-38B [27], the role of the tag length *t* is to prevent guessing attacks: a randomly chosen tag will be valid with a probability of 2^{-t}, and an adversary can perform repeated guesses to increase the probability that one or more of them will be accepted as valid. To protect against guessing attacks, NIST SP 800-38B requires a tag length of at least 64 bits in general and allows shorter tags only for specific applications. Similar to the considerations for collision attacks in Section 4.3, an upper bound for the success probability of a guessing attack (involving multiple trials) can be derived from the speed of the connection. In fact, guessing attacks cannot be performed "offline" but require that the device (containing the key) performs the verification operation to determine whether the guess is correct. This contrasts with exhaustive key search, which can be performed offline after a small number of plaintext-ciphertext pairs have been obtained.

However, the success probability of a guessing attack depends on the tag length rather than the key. Therefore, the recommendation in NIST SP 800-38B to retire the key after a certain number of tag verification failures may be misunderstood. Changing the key does not prevent guessing attacks at all, but it does limit exposure since the adversary can no longer use data that was valid under an old key. Moreover, an adversary may target several devices at the same time.

The inclusion of more detailed guidance regarding applications where short tags may be appropriate should be considered for NIST SP 800-38B. For GCM, more detailed guidance is given in Appendix C of NIST SP 800-38D [29], which suggests that tags of 32 or 64 bits may be appropriate for some voice or video applications.

[7] In this context, any block collision can lead to a message forgery that is straightforward to extend to additional forgeries.

6. NIST SP 800-38C: The CCM Mode for Authentication and Confidentiality

NIST SP 800-38C [28] specifies the CCM mode of operation: a combination of the CTR and CBC-MAC modes of operation using the MAC-then-Encrypt paradigm. As explained in NIST SP 800-38C [28], it is intended to be compatible with IEEE 802.11 [42]. A security proof for CCM is given by Jonsson [44]. The goal of CCM is to provide authenticated encryption with associated data (AEAD). The AEAD functionality is also provided by GCM (see Section 7), so the focus will be on pointing out some similarities and differences between these two NIST-recommended AEAD modes of operation.

In the case of CCM, the tag is part of the ciphertext. This convention is different than the specification of GCM in NIST SP 800-38D [29], where the plaintext and ciphertext are of the same length and the tag is a separate output. To simplify the analysis of CCM in this report, the analysis will follow the CCM convention and assume that the tag is part of the ciphertext.

The input of an AEAD algorithm is the "plaintext" (which is to be both encrypted and authenticated), the "associated data" (which is only to be authenticated), and the nonce (a bit string that is not to be repeated). In contrast to the definition of the CTR mode in NIST SP 800-38A, a zero-length plaintext is allowed for CCM. This can be useful for applications that require an AEAD algorithm that provides both "integrity and confidentiality" and "integrity without confidentiality" (e.g., MACsec [41]). Otherwise, an extra cipher suite would be needed to handle the integrity-only case.

There is, unfortunately, an error in the current specification of CCM: the first step of the decryption-verification process implies that "INVALID" is returned for zero-length plaintexts. Encryption followed by decryption under the same key should always return the original plaintext (and not "INVALID").

Among the topics in Rogaway and Wagner's critique of CCM [76] is its parameterization: they point out that the user is required to choose the maximum message length, which involves a trade-off with the nonce length. For example, if the user choses a nonce length of 104 bits, then the length of the plaintext must be less than 64 KiB (kibibytes). If plaintext of a length up to $2^{64} - 1$ bytes is allowed, then the nonce can only be 56 bits long. For the latter case, repetition is expected after about 2^{28} random nonces due to the birthday paradox, which may be reached in some practical applications. This may occur, for example, if an application processes a variable-length nonce by hashing and truncating the output to the required length.

Unlike CMAC, CCM does not allow tags of less than 32 bits. This may be interpreted as a functional requirement imposed by the encoding of CCM. Tag lengths of less than 64 bits require a "careful analysis of the risks of accepting inauthentic data as authentic" [28]. A suggestion for improvement could be to disallow or restrict such use cases to avoid the risk that short tags would be used in applications where they are not appropriate.

Rogaway and Wagner [76] suggest that even one-bit tags are acceptable for video frames, as a limited number of forgeries does not have a "detrimental effect" for such applications. However, it becomes difficult to ensure technical privacy properties (e.g., anonymity of the sender and unlinkability between the sender and receiver) if there is no mechanism to ensure

that an adversary cannot make modifications to the ciphertext at its source and detect those modifications when the plaintext is made publicly available (e.g., by a journalist).

7. NIST SP 800-38D: Galois/Counter Mode (GCM) and GMAC

GCM, specified in NIST SP 800-38D [29], combines the CTR mode of operation with a polynomial hash function called GHASH using the Encrypt-then-MAC paradigm. The nonce for GCM can be of any length from 1 to $2^{64} - 1$ bits. An advantage of the GCM specification is that variable-length nonces can be used where desired, or a 96-bit nonce can be used "directly," i.e., without any of the overhead that comes with a variable-length nonce.

Iwata et al. [43] showed that the original proof of security for GCM was flawed as well as how to "repair" it. They also showed that GCM has better security bounds for 96-bit nonces than for variable-length nonces. Consequently, some protocols such as TLS 1.3 only allow GCM with a 96-bit nonce.

Another shortcoming of GCM is that the maximum plaintext length is $2^{39} - 256$ bits, which is about 64 GiB (gibibytes). As with CCM, the limitation is due to the underlying CTR mode: going beyond the limit leads to a complete breakdown in security, which is similar to the reuse of a one-time pad. Files of more than 64 GiB are not unusual in video editing and genomics research, but they cannot be encrypted using the GCM mode. For example, the Cryfa [38][72] tool to encrypt genomic data suggests using the command line tool "split" to handle files of more than 64 GiB (and the "cat" command line tool to rejoin the parts after decryption), a cumbersome process that could have been avoided if GCM supported larger plaintexts.

NIST's Galois/Counter Mode (GCM) and GMAC Validation System (GCMVS) [48] also contain tests for testing for GCM-AES-XPN, where "XPN" is an abbreviation of "eXtended Packet Numbering." GCM-AES-XPN is used in IEEE 802.1AE, also known as MACsec [41]. The only difference with GCM is that there is an additional salt input, which is XORed with the IV input to create the nonce for GCM.

The rationale for the salt input of GCM-AES-XPN is to mitigate the concern that "counter mode gives an attacker exactly what they want for integral cryptanalysis: a complete set of block cipher inputs that differ only in some bit positions" [54]. This is an incorrect understanding of the security of block cipher modes of operation and of the AES block cipher. The security model assumes that the adversary can query plaintexts of its choosing. However, care must now be taken that no additional vulnerability is introduced by GCM-AES-XPN: rather than ensuring the uniqueness of the nonce, the uniqueness of the XOR of the IV and the salt must be ensured. Similarly, Cryfa shuffles the plaintext before the GCM encryption is applied in an unnecessary attempt to mitigate known-plaintext attacks.

Lastly, note that in the multi-key security [56] setting where cryptographic algorithms are used under different keys, Luykx et al. proved that GCM, as a mode, does not have multi-key degradation [53].

8. NIST SP 800-38E: The XTS-AES Mode for Confidentiality on Storage Devices

The goal of XTS-AES is full-disk encryption (FDE), which provides a length-preserving encryption of every "sector" or "block" of a storage device, typically consisting of 512, 2048, or 4096 bytes. However, XTS-AES is also used in other contexts where length-preserving encryption is desirable, such as to encrypt swap files, memory content, or file content in file-based encryption (FBE). Hereafter, XTS-AES will be referred to as XTS for simplicity.

For a brief overview of the FDE setting, see Ferguson's paper [34], which introduces the BitLocker Drive Encryption feature of Windows Vista. Ferguson notes that "any time you want to encrypt data, AES-CBC is a leading candidate." However, Ferguson points out that when using CBC, "the attacker can flip arbitrary bits in one block [of plaintext] at the cost of randomizing the previous block [of plaintext]," and explains how this property can be used to attack executable files. This property is also noted in NIST SP 800-38A, as was explained in Section 4. Fujita et al. [36] also show that CBC-encrypted binary files can be modified so that the decryption allows arbitrary code execution (ACE).

Ferguson points out security problems with other solutions, such as LRW (Liskov-Rivest-Wagner), an ECB-like mode that independently encrypts every 128-bit block of data using a different AES-based tweakable block cipher. He nevertheless settles on using AES-CBC but combines it with an "elephant diffuser" to spread differences over an entire sector. Such a diffuser may sound similar to the shuffling done by Cryfa before GCM encryption (see Section 7), but there is an essential difference: modifying a valid GCM ciphertext should lead to a verification failure, but FDE applications will always return "some" plaintext with potentially dangerous consequences. Note, however, that the default encryption method in BitLocker was changed to AES-CBC *without* the diffuser, and since Windows 10, XTS-AES became the default encryption algorithm for BitLocker.

XTS has many similarities with LRW in the sense that it is also a "narrow block" mode of operation that independently encrypts every 128-bit block of data. In fact, LRW was present in the initial drafts of the IEEE 1619 [39] until the algorithm was replaced by XTS following concerns about the insecurity of LRW in the Key-Dependent Message (KDM) setting. KDM security was introduced by Black et al. [18] to consider the scenario in which the adversary has access to the encryption of plaintexts that may depend on the key. It was pointed out during the standardization of IEEE 1619 that key-dependent data may be swapped to disk and, therefore, be part of the plaintext of a full-disk encryption scheme [6].

Two other notable features of XTS are that it requires two different AES keys for two different AES block ciphers (i.e., the key size for XTS is twice the security strength) and that partial final blocks can be handled using "ciphertext stealing." Both properties have been subject to some criticism. Liskov and Minematsu [52] point out that the use of two AES keys may have been due to an incorrect understanding of the underlying XEX construction.[8] Rogaway [74] argues that the mechanism for ciphertext stealing may not have the expected security properties.

[8] The Chair of the IEEE 1619 Working Group suggested that NIST consider allowing the same AES key to be used in both AES block ciphers prior to NIST standardization [6]. However, such a one-key variant of XTS is not a secure instantiation of XEX. As pointed out by Liskov and Minematsu [52], an exponent of zero would need to be avoided as well.

It seems that the decision to standardize a “narrow block” rather than a “wide block” (i.e., sector-wide) mode of operation in IEEE 1619 was mainly due to patent concerns. Nevertheless, IEEE 1619.2 [40] standardizes two “wide block” modes of operation, which additionally provide support for associated data.

Lastly, note that a “wide block” mode of operation called Adiantum [21] can be enabled on devices that run Android 9 and higher. Adiantum is not based on AES and is only recommended for devices that lack AES instructions. By default, Android’s FBE uses AES in two different modes: 1) CBC mode with ciphertext stealing to encrypt filenames and 2) XTS mode to encrypt file content and metadata [5].

9. NIST SP 800-38F: Methods for Key Wrapping

The key wrapping problem addresses the authenticated encryption of a key with another key. This encryption is done deterministically, so there is no IV involved, and the same plaintext key should not be encrypted twice under the same key-wrapping key.

NIST SP 800-38F [31] presents a solution to the key wrapping problem that entails a large number of AES evaluations, as shown in Figure 1 of that publication. At the time of this writing, Triple-DES can be used instead of AES, although Triple-DES will be disallowed after 2023.

An academic treatment of deterministic authenticated encryption by Rogaway and Shrimpton [75] shows that the same problem can also be addressed by much more efficient modes of operation. They formalize the approach in NIST 800-38F as “pad-then-encipher,” where “enciphering” refers to the “wide block” mode of operation mentioned in Section 8. Moreover, Rogaway and Shrimpton introduce the term “misuse-resistant” authenticated encryption in the sense that it provides the maximum level of security when nonces are reused. This “misuse-resistant” setting will be revisited in Section 10.

10. Implementation Considerations

NIST IR 8319 [55] noted that, ideally, implementations would compute cryptographic algorithms correctly and without revealing any additional information that an attacker may use to recover the key; however, in the physical world, information may inadvertently be leaked through side channels or through the injection of faults into the computation. In particular, information may leak not only through the implementation of AES (e.g., due to a cache-timing attack [15] or Rowhammer attack [49]) but also through the implementation of the mode of operation. Another potential concern is that, due to certain implementation failures, IVs may be reused, either when an adversary manages to set the IV to a certain value, or when the implementation fails to ensure the uniqueness of the IVs. Non-invasive attacks on implementations of modes of operation can be mounted through either side-channel information or fault injection [23].

As discussed in Section 4, it is difficult for an implementation to avoid timing attacks when a confidentiality mode is combined with an integrity mode in commonly used protocols, which motivated the deprecation of all confidentiality modes in favor of AEAD modes when TLS 1.3 was introduced. In fact, a very typical example of a timing attack applies to any unprotected implementation of a mode that provides integrity (e.g., CMAC, CCM, GCM, and GMAC). Specifically, the verification operation involves comparing the given tag with the calculated tag, and a straightforward implementation will perform this comparison byte-by-byte, terminating early if an inequality is found.

This timing information allows an adversary to guess the correct value of the tag byte-by-byte, even when the tag is computed remotely over a network. An example of an application that was vulnerable to this attack was Jenkins (CVE-2020-2102 [70]), which performed a non-constant-time comparison function when verifying an HMAC (Keyed-hash Message Authentication Code) tag. Note that a constant-time comparison function should be implemented carefully, as there is a risk that compiler optimizations will turn constant-time code into variable-time code. Unfortunately, it seems that none of the documents in the NIST SP 800-38 series contain a warning that implementations may be vulnerable to this timing attack.[9]

In the security definitions, processing a message is assumed to be an atomic operation, so the adversary cannot observe part of the plaintext or the ciphertext before the computation is complete and cannot observe the decrypted ciphertext before verification is complete. However, this may not be the case for common implementations that process inputs in smaller chunks and may transmit partial outputs when available or store them in an insecure buffer. This leads to “blockwise-adaptive” attacks on CBC [46] and attacks on CCM and GCM under the Release of Unverified Plaintext (RUP) [4].[10]

[9] There is a note on p. 11 of NIST SP 800-38C that an implementation shall ensure that an adversary cannot distinguish between different error messages, “for example, from the timing of the error message.” The timing attack that is mentioned is unrelated to this; the question is whether the error message in the final step of the decryption-verification process is returned in constant time. Another unrelated note appears in NIST SP 800-38D, which mentions side-channel information from internal error logs.

[10] Andreeva et al. [4] point out that achieving the highest level of security in the RUP setting requires the encode-then-encipher approach mentioned in Section 9 as “pad-then-encipher.”

Likewise, it is assumed for the CCM and GCM modes that the nonce is not reused and that the length of the tag is sufficiently long. Otherwise:

- Reuse of the nonce in the CCM or GCM mode results in a reuse of the counter blocks for the underlying CTR mode. This leads to the devastating confidentiality attack described in Section 4.1: with knowledge of certain plaintext blocks, the adversary can use the ciphertext to derive all other plaintext blocks at locations where the counter is reused using a simple XOR operation.
- If the tag is not sufficiently long, a guessing attack can be mounted, as explained in Section 5. The adversary can choose tags randomly and independent of the ciphertext and, when successful, use the fact that the underlying CTR allows an adversary to make arbitrary changes to the plaintext by flipping the corresponding bits in the ciphertext (as explained in Section 4).

In the case of GCM, even more devastating attacks exist in the aforementioned settings. The adversary can recover the authentication key when short tags are used (as shown by Ferguson [35]) or when the nonce is reused (as shown by Joux [45]).

In NIST IR 8319 [55], the single-key setting assumed that the key is drawn uniformly at random and is therefore unknown to the adversary. An adversary may nevertheless know the key if it is generated with an insecure or improperly implemented random bit generator (RBG) or when the protocol uses HMAC with an insecure hash function, such as MD5 or SHA-1, due to the Sloth attack [17] (CVE-2015-7575 [65]).

NIST IR 8319 also explored the case where the adversary knows the key or can even choose the key. This attack scenario can be considered for modes of operation as well. Of interest here is an attack on GCM, where an adversary can construct one ciphertext that decrypts two different plaintexts (under two different keys), leading to a practical attack by Dodis et al. [24] on Facebook's attachment franking scheme.

Another topic to explore is what happens when the same key is used under two different sets of parameters. This occurs, for example, in the variable stretch setting analyzed by Reyhanitabar et al. [73], where one key is used with two different tag lengths. For example, an implementation may provide an additional input that allows the application to choose a "short" tag or a "long" tag, based on the security and efficiency requirements for each message. Given that all of the NIST-recommended modes that provide integrity derive shorter tags by simple truncation, this allows an adversary to trivially construct a "forgery": given a ciphertext, the adversary can produce a "new" ciphertext by simply truncating the tag of the ciphertext to the desired length.

Arguably, KDM security (see Section 8) could be considered to be another type of "misuse." Proper key management should avoid the case in which the plaintext depends on the key, but this may be difficult to ensure in FDE applications.

For applications that may be vulnerable to certain "misuse" scenarios, implementers may not be aware of the potential misuse (or may decide to conveniently ignore the issue). Alternatively, they may resort to certain ad hoc solutions to deal with the problem at hand. A

better approach could be for NIST to alleviate this burden for developers by standardizing a cryptographic mode of operation that is robust against various types of misuse. However, certain types of "misuse," such as nonce repetition or short tags, must not be recommended except for a very restricted set of use cases, as they are insecure for general use and cannot be used to mitigate bad protocol design.

11. Editorial Comments

The goal of editorial comments is to assess whether the standards are correct, complete, consistent, and unambiguous. Overall, the NIST SP 800 Series has a very high editorial quality, and errors in the specifications have already been corrected. At the time of this writing, one exception is the error in the CCM specification, as explained in Section 6.

There are also some inconsistencies in notation and terminology between the different documents, such as whether the tag is part of the ciphertext or not (see Section 6).

In addition to improving the consistency of the notation, terminology, and perhaps even the structure of the NIST SP 800-38 Series, it would be helpful to explicitly point out some differences between the modes.

12. Recommendations

The following is a summary of this report's recommendations:

- Consider disallowing ECB for encrypting secrets.
- Consider not yet deprecating the other NIST SP 800-38A modes, as they are widely used in certain applications where a more secure NIST-recommended alternative is not yet available.
- Consider the standardization of an AEAD mode of operation to address certain "misuse" scenarios, including nonce reuse and short tags (or no tags). This mode of operation is not intended for general use but must be restricted to the specific applications where such types of "misuse" may be unavoidable. The efficiency of this AEAD mode must be similar or better than the current NIST-recommended modes.
- Consider not yet deprecating NIST SP 800-38E and SP 800-38F, as the applications for which they are intended also require a mode with certain misuse resistance properties that is not yet available as a NIST standard.
- Consider reaffirming NIST SP 800-38B, SP 800-38C, and SP 800-38D and possibly making some corrections to aim for consistent levels of security between the documents, such as providing consistent restrictions on tag lengths for certain applications. Additionally, backward compatible extensions of these standards may be considered if there is sufficient demand (e.g., extending the specifications to overcome the plaintext length limits).
- Consider minor fixes and clarifications for all documents.

References

[1] AlFardan NJ, Paterson KG (2013) Lucky Thirteen: Breaking the TLS and DTLS Record Protocols. 2013 IEEE Symposium on Security and Privacy (IEEE S&P 2013) (IEEE, San Francisco, CA), pp. 526–540. https://doi.org/10.1109/SP.2013.42

[2] Almeida JB, Barbosa M, Barthe G, Dupressoir F (2016) Verifiable Side-Channel Security of Cryptographic Implementations: Constant-Time MEE-CBC. Fast Software Encryption (FSE 2016), ed Peyrin T (Springer, Bochum, Germany), pp. 163–184. https://doi.org/10.1007/978-3-662-52993-5_9

[3] Anand MV, Targhi EE, Tabia GN, Unruh D (2016) Post-Quantum Security of the CBC, CFB, OFB, CTR, and XTS Modes of Operation. Post-Quantum Cryptography (PQCrypto 2016), ed Takagi T (Springer, Fukuoka, Japan), Lecture Notes in Computer Science 9606, pp. 44–63. https://doi.org/10.1007/978-3-319-29360-8_4

[4] Andreeva E, Bogdanov A, Luykx A, Mennink B, Mouha N, Yasuda K (2014) How to Securely Release Unverified Plaintext in Authenticated Encryption, Advances in Cryptology – ASIACRYPT 2014, eds Sarkar P, Iwata T (Springer, Kaoshiung, Taiwan), Lecture Notes in Computer Science 8873, pp. 105–125. https://doi.org/10.1007/978-3-662-45611-8_6

[5] Android Open Source Project (2021) File-Based Encryption. Available at https://source.android.com/security/encryption/file-based

[6] Ball MV (2008) NIST's Consideration of XTS-AES as standardized by IEEE Std 1619-2007. Available at https://csrc.nist.gov/csrc/media/projects/block-cipher-techniques/documents/bcm/comments/xts/xts_comments-ball.pdf

[7] Barker EB (2016) Recommendation for Key Management: Part 1 – General. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-57 Part 1, Rev. 4. https://doi.org/10.6028/NIST.SP.800-57pt1r4

[8] Barker EB (2020) Recommendation for Key Management: Part 1 – General. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-57 Part 1, Rev. 5. https://doi.org/10.6028/NIST.SP.800-57pt1r5

[9] Barker EB, Kelsey J (2015) Recommendation for Random Number Generation Using Deterministic Random Bit Generators. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-90A Rev. 1. https://doi.org/10.6028/NIST.SP.800-90Ar1

[10] Barker E, Mouha N (2017) Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-67 Rev. 2. https://doi.org/10.6028/NIST.SP.800-67r2

[11] Barker E, Roginsky A (2019) Transitioning the Use of Cryptographic Algorithms and Key Lengths. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-131A Rev. 2. https://doi.org/10.6028/NIST.SP.800-131Ar2

[12] Bassham LE (2002) The Advanced Encryption Standard Algorithm Validation Suite (AESAVS). Available at https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Algorithm-Validation-Program/documents/aes/AESAVS.pdf

[13] Bellare M, Rogaway P (2008) Code-Based Game-Playing Proofs and the Security of Triple Encryption. Cryptology ePrint Archive preprint, Report 2004/331. https://eprint.iacr.org/2004/331

[14] Bellare M, Rogaway P (2015) Introduction to Modern Cryptography. Available at https://web.cs.ucdavis.edu/~rogaway/classes/227/spring05/book/main.pdf

[15] Bernstein DJ (2005) Cache-timing attacks on AES. Available at https://cr.yp.to/antiforgery/cachetiming-20050414.pdf

[16] Bhargavan K, Leurent G (2016) On the Practical (In-)Security of 64-Bit Block Ciphers: Collision Attacks on HTTP over TLS and OpenVPN. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (ACM CCS 2016) (ACM, Vienna, Austria), pp. 456–467. https://doi.org/10.1145/2976749.2978423

[17] Bhargavan K, Leurent G (2016) Transcript Collision Attacks: Breaking Authentication in TLS, IKE, and SSH. Proceedings of the 2016 Network and Distributed System Security Symposium (NDSS 2016) (Internet Society, San Diego, SA) https://doi.org/10.14722/ndss.2016.23418

[18] Black J, Rogaway P, Shrimpton T (2002) Encryption-Scheme Security in the Presence of Key-Dependent Messages. Selected Areas in Cryptography (SAC 2002), eds Nyberg K, Heys HM (Springer, St. Johns, Canada), Lecture Notes in Computer Science 2595, pp. 62–75. https://doi.org/10.1007/3-540-36492-7_6

[19] Canvel B, Hiltgen A, Vaudenay S, Vuagnoux M (2003) Password Interception in a SSL/TLS Channel. Advances in Cryptology (CRYPTO 2003), ed Boneh D (Springer, Santa Barbara, CA), Lecture Notes in Computer Science 2729, pp. 583–599. https://doi.org/10.1007/978-3-540-45146-4_34

[20] Cooper D, Ferraiolo H, Mehta K, Francomacaro S, Chandramouli R, Mohler J (2015) Interfaces for Personal Identity Verification. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-73-4. https://doi.org/10.6028/NIST.SP.800-73-4

[21] Crowley P, Biggers E (2018) Adiantum: Length-Preserving Encryption for Entry-Level Processors. IACR Transactions on Symmetric Cryptology 2018(4), pp. 39–61. https://doi.org/10.13154/tosc.v2018.i4.39-61

[22] Daemen J, Rijmen V (2020) The Design of Rijndael (Springer, Berlin, Heidelberg, Germany), 2nd Ed. https://doi.org/10.1007/978-3-662-60769-5

[23] Dobraunig C, Eichlseder M, Korak T, Lomné V, Mendel F (2016) Statistical Fault Attacks on Nonce-Based Authenticated Encryption Schemes. Advances in Cryptology (ASIACRYPT 2016), eds Cheon JH, Takagi T (Springer, Hanoi, Vietnam), Lecture Notes in Computer Science 10031, pp. 369–395. https://doi.org/10.1007/978-3-662-53887-6_14

[24] Dodis Y, Grubbs P, Ristenpart T, Woodage J (2018) Fast Message Franking: From Invisible Salamanders to Encryptment. Advances in Cryptology (CRYPTO 2018), eds Shacham H, Boldyreva A (Springer, Santa Barbara, CA), Lecture Notes in Computer Science 10991, pp. 155–186. https://doi.org/10.1007/978-3-319-96884-1_6

[25] Dworkin M (2001) Recommendation for Block Cipher Modes of Operation: Methods and Techniques. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-38A. https://doi.org/10.6028/NIST.SP.800-38A

[26] Dworkin M (2010) Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode. (National Institute of Standards and Technology, Gaithersburg, MD), Addendum to NIST Special Publication (SP) 800-38A. https://doi.org/10.6028/NIST.SP.800-38A-Add

[27] Dworkin M (2005) Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-38B. https://doi.org/10.6028/NIST.SP.800-38B

[28] Dworkin M (2004) Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-38C. https://doi.org/10.6028/NIST.SP.800-38C

[29] Dworkin M (2007) Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-38D. https://doi.org/10.6028/NIST.SP.800-38D

[30] Dworkin M (2010) Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-38E. https://doi.org/10.6028/NIST.SP.800-38E

[31] Dworkin M (2012) Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-38F. https://doi.org/10.6028/NIST.SP.800-38F

[32] Dworkin M (2016) Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-38G. https://doi.org/10.6028/NIST.SP.800-38G

[33] Dworkin M (2019) Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption. (National Institute of Standards and Technology, Gaithersburg, MD), Draft NIST Special Publication (SP) SP 800-38G Rev. 1. https://doi.org/10.6028/NIST.SP.800-38Gr1-draft

[34] Ferguson N (2006) AES-CBC + Elephant diffuser – A Disk Encryption Algorithm for Windows Vista. Available at https://www.microsoft.com/en-us/download/details.aspx?id=13866

[35] Ferguson N (2005) Authentication Weaknesses in GCM. Available at https://csrc.nist.gov/CSRC/media/Projects/Block-Cipher-Techniques/documents/BCM/Comments/CWC-GCM/Ferguson2.pdf

[36] Fujita R, Isobe T, Minematsu K (2020) ACE in Chains: How Risky Is CBC Encryption of Binary Executable Files? Applied Cryptography and Network Security (ACNS 2020), eds Conti M et al. (Springer, Rome, Italy), Lecture Notes in Computer Science 12146, pp. 187–207. https://doi.org/10.1007/978-3-030-57808-4_10

[37] Furtoransky A, Kargieman E, Pacetti A (1998) An Attack on CRC-32 Integrity Checks of Encrypted Channels using CBC and CFB Modes. Available at

https://www.coresecurity.com/sites/default/files/private-files/publications/2016/05/KargiemanPacettiRicharte_1998-CRC32.pdf

[38] Hosseini M, Pratas D, Pinho AJ (2019) Cryfa: A Secure Encryption Tool for Genomic Data. Bioinformatics 35(1), pp. 146–148. https://doi.org/10.1093/bioinformatics/bty645

[39] Institute of Electrical and Electronics Engineers (2001) IEEE Standard for Cryptographic Protection of Data on Block-Oriented Storage Devices. IEEE Std 1619-2018 (Revision of IEEE Std 1619-2007). https://doi.org/10.1109/IEEESTD.2019.8637988

[40] Institute of Electrical and Electronics Engineers (2001) IEEE Standard for Wide-Block Encryption for Shared Storage Media. IEEE Std 1619.2-2010. https://doi.org/10.1109/IEEESTD.2011.5729263

[41] Institute of Electrical and Electronics Engineers (2018) IEEE Standard for Local and Metropolitan Area Networks-Media Access Control (MAC) Security. IEEE Std 802.1AE 2018 (Revision of IEEE Std 802.1AE-2006). https://doi.org/10.1109/IEEESTD.2018.8585421

[42] Institute of Electrical and Electronics Engineers (2021) IEEE Standard for Information Technology — Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks — Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016). https://doi.org/10.1109/IEEESTD.2021.9363693

[43] Iwata T, Ohashi K, Minematsu K (2012) Breaking and Repairing GCM Security Proofs. Advances in Cryptology (CRYPTO 2012), eds Safavi-Naini R, Canetti R (Springer, Santa Barbara, CA), Lecture Notes in Computer Science 7417, pp. 31–49. https://doi.org/10.1007/978-3-642-32009-5_3

[44] Jonsson J (2006) On the Security of CTR + CBC-MAC. Selected Areas in Cryptography (SAC 2003), eds Nyberg K, Heys HM (Springer, Ottawa, Canada), Lecture Notes in Computer Science 2595, pp. 76–93. https://doi.org/10.1007/3-540-36492-7_7

[45] Joux A (2007) Authentication Failures in NIST version of GCM. Available at https://csrc.nist.gov/csrc/media/projects/block-cipher-techniques/documents/bcm/joux_comments.pdf

[46] Joux A, Martinet G, Valette F (2003) Blockwise-Adaptive Attackers Revisiting the (In)Security of Some Provably Secure Encryption Modes: CBC, GEM, IACBC. Advances in Cryptology (CRYPTO 2002), ed Moti Yung (Springer, Santa Barbara, CA), Lecture Notes in Computer Science 2442, pp. 17–30. https://doi.org/10.1007/3-540-45708-9_2

[47] Jutla CS (2004) Lower Bound on Linear Authenticated Encryption. Selected Areas in Cryptography (SAC 2004), eds Matsui M, Zuccherato RJ (Springer, Waterloo, Canada), Lecture Notes in Computer Science 3006, pp. 348–360. https://doi.org/10.1007/978-3-540-24654-1_25

[48] Keller SS, Hall TA (2017) The Galois/Counter Mode (GCM) and GMAC Validation System (GCMVS) with the Addition of XPN Validation Testing. Available at https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Algorithm-Validation-Program/documents/mac/gcmvs.pdf

[49] Kim Y, Daly R, Kim J, Fallin C, Lee JH, Lee D, Wilkerson D, Lai K, Mutlu O (2014) Flipping Bits in Memory without Accessing Them: An Experimental Study of DRAM Disturbance Errors. 2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA), (IEEE, Minneapolis, MN), pp. 361–372. https://doi.org/10.1109/ISCA.2014.6853210

[50] Leurent G, Sibleyras F (2018) The Missing Difference Problem, and Its Applications to Counter Mode Encryption. Advances in Cryptology (EUROCRYPT 2018), eds Nielsen JB, Rijmen V (Springer, Tel-Aviv, Israel), Lecture Notes in Computer Science 10821, pp. 745–770. https://doi.org/10.1007/978-3-319-78375-8_24

[51] Lipmaa H, Rogaway P, Wagner D (2000) Comments to NIST concerning AES Modes of Operations: CTR-Mode Encryption. Available at https://www.cs.ucdavis.edu/~rogaway/papers/ctr.pdf

[52] Liskov M, Minematsu K (2008) Comments on XTS-AES. Available at https://csrc.nist.gov/csrc/media/projects/block-cipher-techniques/documents/bcm/comments/xts/xts_comments-liskov_minematsu.pdf

[53] Luykx A, Mennink B, Paterson KG (2017) Analyzing Multi-Key Security Degradation. Advances in Cryptology (ASIACRYPT 2017), eds Takagi T, Peyrin T (Springer, Hong Kong, China), Lecture Notes in Computer Science 10625, pp. 575–605. https://doi.org/10.1007/978-3-319-70697-9_20

[54] McGrew D (2013) Generation of Deterministic Initialization Vectors (IVs) and Nonces. Available at https://datatracker.ietf.org/doc/html/draft-mcgrew-iv-gen-03

[55] Mouha N (2021) Review of the Advanced Encryption Standard. (National Institute of Standards and Technology, Gaithersburg, NIST Interagency or Internal Report (IR) 8319. https://doi.org/10.6028/NIST.IR.8319

[56] Mouha N, Luykx A (2015) Multi-Key Security: The Even-Mansour Construction Revisited. Advances in Cryptology (CRYPTO 2015), eds Gennaro R, Robshaw M (Springer, Santa Barbara, CA), Lecture Notes in Computer Science 9215, pp. 209–223. https://doi.org/10.1007/978-3-662-47989-6_10

[57] National Bureau of Standards (1980) DES Modes of Operation. (Department of Commerce, Washington, D.C.), Federal Information Processing Standards Publications (FIPS PUBS) FIPS 81. https://doi.org/10.6028/NBS.FIPS.81

[58] National Institute of Standards and Technology (2001; Updated 2023) Advanced Encryption Standard (AES). (Department of Commerce, Washington, D.C.), Federal Information Processing Standards Publications (FIPS PUBS) FIPS 197, November 2001; Updated May 2023, https://doi.org/10.6028/NIST.FIPS.197-upd1

[59] National Vulnerability Database (2016) CVE-1999-1085. Available at https://nvd.nist.gov/vuln/detail/CVE-1999-1085

[60] National Vulnerability Database (2018) CVE-2001-0144. Available at https://nvd.nist.gov/vuln/detail/CVE-2001-0144

[61] National Vulnerability Database (2016) CVE-2003-0078. Available at https://nvd.nist.gov/vuln/detail/CVE-2003-0078

[62] National Vulnerability Database (2018) CVE-2011-3389. Available at https://nvd.nist.gov/vuln/detail/CVE-2011-3389

[63] National Vulnerability Database (2018) CVE-2012-4929. Available at https://nvd.nist.gov/vuln/detail/CVE-2012-4929
[64] National Vulnerability Database (2020) CVE-2013-3587. Available at https://nvd.nist.gov/vuln/detail/CVE-2013-3587
[65] National Vulnerability Database (2018) CVE-2015-7575. Available at https://nvd.nist.gov/vuln/detail/CVE-2015-7575
[66] National Vulnerability Database (2018) CVE-2016-2107. Available at https://nvd.nist.gov/vuln/detail/CVE-2016-2107
[67] National Vulnerability Database (2021) CVE-2016-2183. Available at https://nvd.nist.gov/vuln/detail/CVE-2016-2183
[68] National Vulnerability Database (2020) CVE-2016-1000344. Available at https://nvd.nist.gov/vuln/detail/CVE-2016-1000344
[69] National Vulnerability Database (2019) CVE-2017-2598. Available at https://nvd.nist.gov/vuln/detail/CVE-2017-2598
[70] National Vulnerability Database (2020) CVE-2020-2102. Available at https://nvd.nist.gov/vuln/detail/CVE-2020-2102
[71] National Vulnerability Database (2020) CVE-2020-11500. Available at https://nvd.nist.gov/vuln/detail/CVE-2020-11500
[72] Pratas D, Hosseini M, Pinho AJ (2017) Cryfa: A Tool to Compact and Encrypt FASTA Files. Practical Applications of Computational Biology & Bioinformatics (PACBB 2017), eds Fdez-Riverola F et al. (Springer, Porto, Portugal), Lecture Notes in Computer Science 616, pp. 305–312. https://doi.org/10.1007/978-3-319-60816-7_37
[73] Reyhanitabar R, Vizár D, Vaudenay S (2016) Authenticated Encryption with Variable Stretch. Advances in Cryptology (ASIACRYPT 2016), eds Cheon JH, Takagi T (Springer, Hanoi, Vietnam), Lecture Notes in Computer Science 10031, pp. 396–425. https://doi.org/10.1007/978-3-662-53887-6_15
[74] Rogaway P (2011) Evaluation of Some Blockcipher Modes of Operation. CRYPTREC Report, Available at: https://www.cs.ucdavis.edu/~rogaway/papers/modes.pdf
[75] Rogaway P, Shrimpton T (2006) A Provable-Security Treatment of the Key-Wrap Problem. Advances in Cryptology (EUROCRYPT 2006), ed Vaudenay S (Springer, Saint Petersburg, Russia), Lecture Notes in Computer Science 4004, pp. 373–390. https://doi.org/10.1007/11761679_23, full version available at https://web.cs.ucdavis.edu/~rogaway/papers/keywrap.pdf
[76] Rogaway P, Wagner D (2003) A Critique of CCM. Available at https://csrc.nist.gov/CSRC/media/Projects/Block-Cipher-Techniques/documents/BCM/Comments/800-38-series-drafts/CCM/RW_CCM_comments.pdf
[77] Tezcan T, Vaudenay S (2011) On Hiding a Plaintext Length by Preencryption. Applied Cryptography and Network Security (ACNS 2011), eds Lopez J, Tsudik G (Springer, Nerja, Spain), Lecture Notes in Computer Science 6715, pp. 345–358. https://doi.org/10.1007/978-3-642-21554-4_20
[78] Thompson A, Taylor BN (2008) Guide for the Use of the International System of Units (SI). (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 811. https://doi.org/10.6028/NIST.SP.811e2008

[79] Turan MS, Barker E, Kelsey J, McKay KA, Baish ML, Boyle M (2018) Recommendation for the Entropy Sources Used for Random Bit Generation. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-90B. https://doi.org/10.6028/NIST.SP.800-90B

[80] Van Houtven L (2020) Crypto 101. Available at https://www.crypto101.io/Crypto101.pdf

[81] Vaudenay S (2002) Security Flaws Induced by CBC Padding — Applications to SSL, IPSEC, WTLS... Advances in Cryptology (EUROCRYPT 2002), ed Knudsen LR (Springer, Amsterdam, The Netherlands), Lecture Notes in Computer Science 2332, pp. 534–545, https://doi.org/10.1007/3-540-46035-7_35

[82] "Definitions," Title 44 U.S. Code, Sec. 3542. 2013 ed. https://www.govinfo.gov/app/details/USCODE-2013-title44/USCODE-2013-title44-chap35-subchapIII-sec3542

Appendix A. List of Symbols, Abbreviations, and Acronyms

ACE
Arbitrary Code Execution

Adv
Advantage

AEAD
Authenticated Encryption with Associated Data

AES
Advanced Encryption Standard

AESAVS
Advanced Encryption Standard Algorithm Validation Suite

BEAST
Browser Exploit Against SSL/TLS

BREACH
Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext

CBC
Cipher Block Chaining

CBC-MAC
Cipher Block Chaining-Message Authentication Code

CCM
Counter with Cipher Block Chaining-Message Authentication Code

CFB
Cipher Feedback

CMAC
Cipher-based Message Authentication Code

CRC
Cyclic Redundancy Check

CRIME
Compression Ratio Info-leak Made Easy

CS
Ciphertext Stealing

CTR
Counter

CVE
Common Vulnerabilities and Exposures

DES
Data Encryption Standard

DRBG
Deterministic Random Bit Generator

ECB
Electronic Codebook

FBE
File-Based Encryption

FDE
Full-Disk Encryption

FIPS
Federal Information Processing Standard

FOIA
Freedom of Information Act

FPE
Format-Preserving Encryption

GCM
Galois/Counter Mode

GMAC
Galois Message Authentication Code

HMAC
Hash-based Message Authentication Code

IEEE
Institute of Electrical and Electronics Engineers

IND-CPA
Left-or-right INDistinguishability under a Chosen-Plaintext Attack

IV
Initialization Vector

KDM
Key-Dependent Message

KW
AES Key Wrap

KWP
AES Key Wrap with Padding

LRW
Liskov-Rivest-Wagner

MAC
Message Authentication Code

MACsec
Media Access Control Security

NIST IR
NIST Interagency or Internal Report

NVD
National Vulnerability Database

OFB
Output Feedback

OTP
One-Time Pad

PIV
Personal Identity Verification

PRF
Pseudo-Random Function

PRP
Pseudo-Random Permutation

RBG
Random Bit Generator

RUP
Release of Unverified Plaintext

SE
Symmetric Encryption

SP
Special Publication

SSH
Secure Shell

SSL
Secure Sockets Layer

TDEA
Triple Data Encryption Algorithm

TKW
TDEA Key Wrap

TLS
Transport Layer Security

XEX
XOR Encrypt XOR

XOR
eXclusive-OR

XPN
eXtended Packet Number

XTS
XEX Tweakable Block Cipher with Ciphertext Stealing

Appendix B. Glossary

associated data
Data that is authenticated but not encrypted.

atomic operation
An atomic operation is effectively executed as a single step, no other process can read or modify the internal state while the atomic operation is executed.

authentication tag (tag)
A cryptographic checksum on data that is designed to reveal both accidental errors and the intentional modification of the data.

bit
A binary digit having a value of 0 or 1.

bit error
The substitution of a '0' bit for a '1' bit or vice versa.

bit string
A finite ordered sequence of bits.

block
For a given block cipher, a bit string whose length is the block size of the block cipher.

block cipher
A family of functions and their inverse functions that is parameterized by cryptographic keys; the functions map bit strings of a fixed length to bit strings of the same length.

block size
The number of bits in an input (or output) block of the block cipher.

byte
A sequence of 8 bits.

ciphertext
The encrypted form of the plaintext.

collision
For a given function, a pair of distinct input values that yield the same output value.

confidentiality
Preserving authorized restrictions on information access and disclosure, including a means for protecting personal privacy and proprietary information. [82]

cookie
A piece of state information supplied by a web server to a browser that is temporarily stored and returned to the server on any subsequent visits or requests.

cryptographic key
A parameter used in the block cipher algorithm that determines the forward cipher operation and the inverse cipher operation.

decryption
The process of transforming ciphertext into plaintext.

EiB (exbibyte)
2^{60} bytes. [78, Sec. 4.3]

elephant diffuser
A (now deprecated) component in BitLocker Drive Encryption to increase resistance against ciphertext modification.

encryption
The process of transforming plaintext into ciphertext.

exclusive-or
The bitwise addition, modulo 2, of two bit strings of equal length.

GiB (gibibyte)
2^{30} bytes. [78, Sec. 4.3]

integrity
Guarding against improper information modification or destruction; includes ensuring information non-repudiation and authenticity. [82]

KiB (kibibyte)
2^{10} bytes. [78, Sec. 4.3]

MAC algorithm
An algorithm that computes an authentication tag from a message and a key. The term “MAC” is sometimes also used to refer to the authentication tag itself.

MiB (mebibyte)
2^{20} bytes. [78, Sec. 4.3]

mode of operation (mode)
An algorithm for the cryptographic transformation of data that is based on a block cipher.

nonce
A value that is only used once.

password
A string of characters (letters, numbers, and other symbols) that is used to authenticate an identity, to verify access authorization, or to derive cryptographic keys.

PiB (pebibyte)
2^{50} bytes. [78, Sec. 4.3]

plaintext
Intelligible data that has meaning and can be understood without the application of decryption.

segment
In the CFB mode, a sequence of bits whose length is a parameter that does not exceed the block size.

semantic security
What can be efficiently computed about some plaintexts from their ciphertexts can be computed, just as easily, in the absence of those ciphertexts. [14]

TiB (tebibyte)
2^{40} bytes. [78, Sec. 4.3]