PUBLIC COMMENTS ON SP 800-38A, Recommendation for Block Cipher Modes of Operation: Methods and Techniques and SP 800-38A Addendum, Three Variants of Ciphertext Stealing for CBC Mode

Comment period: May 10, 2021 – June 11, 2021

In May 2021, NIST's Crypto Publication Review Board initiated a review of **SP 800-38A, Recommendation for Block Cipher Modes of Operation: Methods and Techniques** and SP 800-38A Addendum, Three Variants of Ciphertext Stealing for CBC Mode. This document includes the public comments received during the comment period from May 20, 2021 and June 11, 2021.

More details about this review are available from NIST's Crypto Publication Review Project site.

LIST OF COMMENTS

1. Comments from Mike Grimm (Microsoft), June 9, 2021

NIST requested public comments on the existing AES standard FIPS 197 and block cipher mode standard SP800-38A. These are the comments of Microsoft.

Replace, Rather than Change, Existing Standards

Whilst we have significant concerns with the current state of AES and the block cipher modes, we recommend no changes to these standards. At best, changes to the existing standards would address only some of our concerns. The cost of any change to an existing system is very large; experience shows that it takes over a decade of focused work to switch to a new cryptographic algorithm. Doing that work to address only a subset of our concerns is not worthwhile.

Concerns with Current Standards

We have three main concerns with the current standards:

- Limited block size and key size of AES
- Problems with block cipher modes
- Use of nonces

AES Block and Key Size

The 128-bit block size and 256-bit key size of AES are limiting the security that we can provide to our customers. With new designs regularly targeting 192- or 256-bit security levels, AES becomes a bottleneck.

The 256-bit key size is a limitation. We often use AES-GCM for performance reasons. Because unique nonce generation is hard, some designs generate a fresh AES key for each message or data item, and then use a constant nonce value. The limited key size allows for a multi-target attack where the attacker pre-computes results for a large set of AES keys and waits for the victim to pick one of those keys for a message. This limits the security that can be achieved with AES-256 to a 128-bit security level. Reality is slightly better as the amount of data the victim processes is limited, but with exabyte (10^18 = 2^60 Bytes) datasets already in use and zettabyte (10^21 = 2^70) in the near future, the decrease in security from the desired 256-bit level is significant.

Security for large data sets is also limited by the 128-bit block size. All block cipher modes in current use have weaknesses when the amount of data processed gets close to the birthday bound of 2^64 blocks. For example: with AES-CBC encryption, if two ciphertext blocks have the same value then this reveals the difference between two plaintext blocks. A single exabyte dataset has 2^56 blocks, and thus a chance of around 10^-5 of leaking data, a risk that users should not be forced to take. zettabyte datasets are almost guaranteed to leak data. Similar problems occur in all modes.

Assuming quantum computers can be made to work, a 256-bit key provides only 128 bits of security, and quantum attacks that target the 128-bit block size would require only 2^64 work.

Cipher Modes

The traditional block cipher modes (CBC, ECB, CFB, …) were defined a long time ago, and do not adequately address some practical problems. For example, CBC is perfectly fine if the data is in multiples of the block size. Unfortunately, the commonly used padding together with CBC creates an encryption algorithm that is vulnerable to padding oracle attacks. This is not a flaw of CBC, it is a flaw of the padding and the lack of authentication.

These problems, together with the lack of standardized solutions, has led to a very large number of different constructions. The popular AES-CBC-HMAC-SHA256 combination has been implemented dozens of times, often in incompatible (and sometimes insecure) ways, since there is no standard specifying this combination.

The popular AES-GCM mode has a limited data size that has caused security problems and confusion around re-keying, and this will get worse with time. A file of 64 GB is quite normal in video editing, and cannot be encrypted with AES-GCM.

Use of Nonces

AES-GCM is the go-to algorithm for high-performance data encryption. Unfortunately, the requirement of having a unique nonce turns out to be very hard to achieve in many applications. We have seen numerous uses of AES-GCM that were insecure because of nonce re-use. (Note that Microsoft does not allow the use of random 96-bit nonce values for GCM due to the risk of nonce collision.)

Experience shows that asking developers that do not have a deep understanding of cryptography to generate unique nonce values often leads to insecure designs.

Suggested New Standards

The goal of standards should not just be to standardize an algorithm, but to make it easy for a developer to use the standard in a secure manner. Rather than a set of tools from which an expert cryptographer can build a secure application, standards should provide tools that allow ordinary developers to build secure applications. With this in mind, we recommend that NIST develop two new standards.

New Block Cipher Algorithm

We recommend that NIST standardize a new block cipher with the following properties:

- Key size = 512 bits
- Block size = 512 bits
- Target security level = 256 bits
- Software performance similar to AES with AES-NI instructions

The key and block size are large enough that any kind of collision or meet-in-the-middle attack will still yield a 256-bit security level. The larger block size also eliminates the need for developer-visible nonces, allowing random nonces to be used. Note that the key size is larger than the security level. This is somewhat unusual but avoids the problem of key-collision attacks without needlessly increasing the security level.

The need for good performance in software will probably result in a block cipher that uses the AES round function as a component as that allows the use of the AES round function instructions available on Intel, AMD, and ARM CPUs.

The large block size and key size also makes this cipher secure against generic quantum computing attacks.

New Modes

For this new cipher, we recommend that NIST standardize two new block cipher modes for the following functions:

- Authenticated encryption with authenticated data for data sizes up to 2^128 bytes. This mode should not require a caller-generated nonce.
- Storage encryption, similar to XTS-AES.

The Need for a Secure Alternative to AES

Right now, there is no alternative to AES. If an efficient attack on AES were published tomorrow (perhaps assisted by a quantum computer), then we have no practical way of securing the world's data. For this reason alone, NIST should work on a new block cipher standard.

2. Comments from Matt Campagna, Shay Gueron, Panos Kampanakis, Colm MacCarthaigh, Margaret Salter and Daniel Simon (Amazon), June 11, 2021

Dear Review Board,

The NIST process of standardizing AES has been a success. It has set the standard for standardizing new cryptographic primitives. NIST's commitment to an open design submission and analysis process for selecting new standards establishes trust within the community. It also harnesses the global community to arrive at high assurance cryptographic schemes. We view the periodic reviews and associated public comment periods on existing standards and Special Publications as a useful and necessary component of providing up to date guidance to the cryptographic community.

Comments on FIPS 197, Advanced Encryption Standard (AES), 2001

==========

A good supplement to AES would be a 256-bit block cipher.

A 128-bit block cipher is a permutation of {0, 1}^128 regardless of the key length and its output is distinguishable from random (with high probability) after 2^{64} blocks. Security wise, this limits the number of calls allowed with a given key and at cloud scale this quickly becomes a cause for re-keying. Moreover, modes like GCM require the probability of a {key, iv} collision to be less than 2^{-32}. Note that managing state in a distributed cloud environment introduces latency and complexity. Thus, to avoid state management we often use a random nonce construction, which limits us to 2^{32} encryption invocations under a given key. This adds the burden of frequent re-keying. It would be useful to alleviate these limitations, and a standardized 256-bit block cipher (permutation of {0, 1}^256) is a valuable primitive for building appropriate schemes.

One efficient approach to define such wide block ciphers would leverage the investment of the ecosystem and the industry in hardware and software support for some components of the AES, specifically, inversion in GF(2^8) (which is the source of non-linearity in the AES-like block cipher design). This operation is the major element that speeds the instructions and the full AES (and Rijndael 256) implementation. Using this observation can help the adoption of a new block cipher variant.

A straightforward candidate would be Rijndael 256, which was part of the original proposal for AES (although only the 128-bit block size was standardized). Code that executes Rijndael 256 could use the AES-NI (see Figure 30. "Using the AES instructions to compute a 256-bit block size RINJDAEL round" in [1]). In 2016, the throughput of code that uses AES-NI, running pipelined Rijndael 256 encryption, is 1.54 cycles per byte (cpb) (see the report in [2]). By comparison, AES throughput on such processors is 0.65 cpb. Note that recent architectures have doubled and quadrupled the throughput of AES (the above performance ratio remains the same). Recent architectures also offer instructions to execute GF(2^8) directly, which can be used for AES, Rijndael 256 and other constructions (e.g., SM4). Hardware implementation of Rijndael 256 could re-use components that support AES.

There are also some alternatives that rely on the AES round, as a "fast primitive", and may constitute a different cipher, even with flexible block size [2]. For example, Simpira 256-bit permutation has throughput of 0.94 cpb, and it can be used e.g., with an Even-Mansour construction, to define a block cipher.

A second independent encryption function

With the deprecation of DES and the limitations on 3DES, we are again left at a single point of algorithmic failure in our FIPS certified crypto suite. While AES has held up to analysis over the past 20 years, a second encryption function would provide additional agility should a catastrophic failure occur. While crypto-agility adds a degree of complexity, we value algorithm diversity above this complexity cost.

Comments on NIST SP800-38A

==========

We value the diversity of block cipher modes of operation family. We also recognize that some of these comments may be more appropriate during upcoming comment periods for NIST SP800-38 volumes (B-F).

Addition of Offset CodeBook (OCB) mode

OCB is an Authenticated Encryption with Associated Data (AEAD) scheme. OCB performance in software implementations is generally at least as good as GCM, because OCB does not require a (relatively expensive) Galois field multiplication for each block of input.

MACTag validation enforcement

Implementations of NIST SP800-38D Recommendation for BlockCipher Modes of Operation:

Galois/Counter Mode (GCM) and GMAC Galois/Counter Mode (GCM) allows for recovering the plaintext before (or concurrently with) the authentication. This property can be combined with traditional API practices of operating on passed buffers, and motivate aggressive optimizations with security-wise undesired shortcuts. There is an opportunity to add an additional mode that enforces tag validation prior to decryption alleviating the release of computed plaintext prior to authentication. A possible example can be found in [3]

Derived Key Modes

The block size limits the number of blocks that can be protected under the same key. Some modes of operation impose additional limits on the number of invocations under the same key. A derived key from an IV mode can help mitigate these constraints [4]. A NIST-specified derived key and IV mode would provide the community with a high-assurance construction and foster interoperability. Examples include AES-GCM-SIV [5] which benefits usecases where a stateful counter, a state, or proper randomness cannot be guaranteed.

Handling nonce and IV misuse

Errors in IV and nonce construction continue to plague implementations [6 - 12]. The current publication's Section 5.3 Initialization Vector and Appendix C: Generation of Initialization Vectors details methods of constructing IVs, should be strengthened.

> "The IV need not be secret; however, for the CBC and CFB modes, the IV for any particular execution of the encryption process must be unpredictable, and, for the OFB mode, unique IVs must be used for each execution of the encryption process. The generation of IVs is discussed in Appendix C."

In particular, it would be useful to lead off this paragraph with the importance of using an initialization vector appropriate to the mode, as opposed to comments about secrecy. Section 5.3 should recommend they use a method from Appendix C. A clearer warning about the potential loss of the security of the mode if they fail to use an appropriate method would also be useful.

Stricter guidance on the use of ECB mode

Electronic Code Book mode is known to be problematic in use. There have been numerous deployments that have used ECB erroneously [13 - 21]. The existing language in Section 6.1

> "In the ECB mode, under a given key, any given plaintext block always gets encrypted to the same ciphertext block. If this property is undesirable in a particular application, the ECB mode should not be used."

could be strengthened to prevent misuse in the future.

Regards,
Matt Campagna, Shay Gueron, Panos Kampanakis, Colm MacCarthaigh, Margaret Salter and Daniel Simon

References:

[1] S. Gueron, Intel® Advanced Encryption Standard (AES) New Instructions Set https://www.intel.com/content/dam/doc/white-paper/advanced-encryption-standard-new-instructions-set-paper.pdf (2010)
[2] S. Gueron, N. Mouha, Simpira v2: A Family of Efficient Permutations Using the AES Round Function, ASIACRYPT (1) 2016: 95-125 (also in https://eprint.iacr.org/2016/122.pdf)
[3] S. Gueron, C. MacCarthaigh, A. Weibel: SCRAM -- Short Cut Resistant AEAD Mode https://github.com/aws/s2n-tls/tree/main/scram
[4] S. Gueron, Y.Lindell: Better Bounds for Block Cipher Modes of Operation via Nonce-Based Key Derivation. CCS 2017: 1019-1036
[5] S. Gueron, A. Langley, Y. Lindell:
AES-GCM-SIV: Nonce Misuse-Resistant Authenticated Encryption
https://datatracker.ietf.org/doc/html/rfc8452
[6] CVE-2011-3389 BEAST attack in SSL 3.0 / TLS 1.0. In CBC mode, chained initialization vectors are non-random, allowing decryption of HTTPS traffic using a chosen plaintext attack.
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-3389
[7] CVE-2017-3225 A device bootloader uses a zero initialization vector during AES-CBC
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-3225
[8] CVE-2016-6485 PHP rand function to generate a random number for the initialization vector (not cryptographic RBG).
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-6485
[9] CVE-2014-5386 Release of an IV from an unseeded RBG reveals state of low-entropy state.
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-5386
[10] CVE-2020-5408 A fixed null IV with CBC mode, allowing attackers to decrypt traffic in applications that use this functionality.
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-5408

[11] CVE-2017-17704 Each message is encrypted in CBC mode and restarts with the fixed IV, leading to replay attacks of entire messages.
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-17704
[12] CVE-2017-11133 Application uses secret and the IV are generated with math.random(), not a cryptographic RBG.
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-11133
[13] CVE-2002-1697 Electronic Code Book (ECB) mode in VTun 2.0 through 2.5 uses a weak encryption algorithm that produces the same ciphertext from the same plaintext blocks, which could allow remote attackers to gain sensitive information.
https://nvd.nist.gov/vuln/detail/CVE-2002-1697
[14] CVE-2012-3458 Beaker before 1.6.4, when using PyCrypto to encrypt sessions, uses AES in ECB cipher mode, which might allow remote attackers to obtain portions of sensitive session data via unspecified vectors.
https://nvd.nist.gov/vuln/detail/CVE-2012-3458
[15] CVE-2013-7252 kwalletd in KWallet before KDE Applications 14.12.0 uses Blowfish with ECB mode instead of CBC mode when encrypting the password store, which makes it easier for attackers to guess passwords via a codebook attack.
https://nvd.nist.gov/vuln/detail/CVE-2013-7252
[16] CVE-2017-8867 Elemental Path's CogniToys Dino smart toys through firmware version 0.0.794 use AES-128 with ECB mode to encrypt voice traffic between the device and remote server, allowing a malicious user to map encrypted traffic to a particular AES key index and gaining further access to eavesdrop on privacy-sensitive voice communication of a child and their Dino device.
https://nvd.nist.gov/vuln/detail/CVE-2017-8867
[17] CVE-2017-2598 Jenkins before versions 2.44, 2.32.2 uses AES ECB block cipher mode without IV for encrypting secrets which makes Jenkins and the stored secrets vulnerable to unnecessary risks (SECURITY-304).
https://nvd.nist.gov/vuln/detail/CVE-2017-2598
[18] CVE-2016-1000344. In the Bouncy Castle JCE Provider version 1.55 and earlier the DHIES implementation allowed the use of ECB mode. This mode is regarded as unsafe and support for it has been removed from the provider.
https://nvd.nist.gov/vuln/detail/CVE-2016-1000344
[19] CVE-2016-1000352 In the Bouncy Castle JCE Provider version 1.55 and earlier the ECIES implementation allowed the use of ECB mode. This mode is regarded as unsafe and support for it has been removed from the provider.
https://nvd.nist.gov/vuln/detail/CVE-2016-1000352
[20] CVE-2020-11500 Zoom Client for Meetings through 4.6.9 uses the ECB mode of AES for

video and audio encryption. Within a meeting, all participants use a single 128-bit key.
https://nvd.nist.gov/vuln/detail/CVE-2020-11500
[21] CVE-2018-5548 On BIG-IP APM 11.6.0-11.6.3, an insecure AES ECB mode is used for orig_uri parameter in an undisclosed /vdesk link of APM virtual server configured with an access profile, allowing a malicious user to build a redirect URI value using different blocks of cipher texts.
https://nvd.nist.gov/vuln/detail/CVE-2018-5548