WIKIPEDIA
The Free Encyclopedia

Advanced Encryption Standard

The **Advanced Encryption Standard** (**AES**), also known by its original name **Rijndael** (Dutch pronunciation: [ˈrɛindaːl]),[5] is a specification for the encryption of electronic data established by the US National Institute of Standards and Technology (NIST) in 2001.[6]

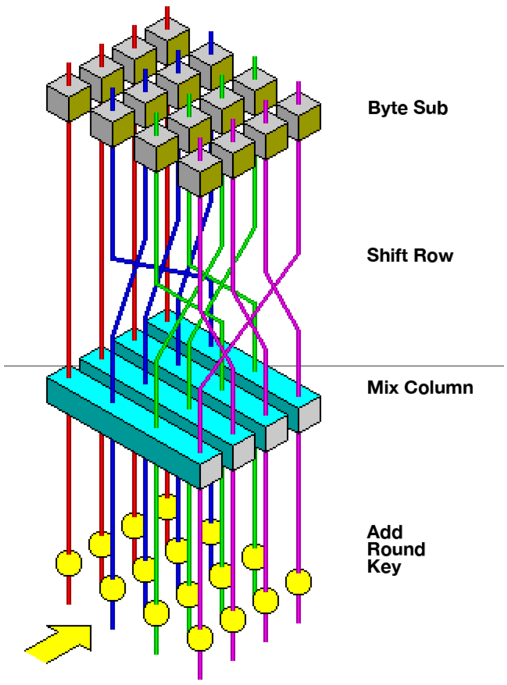
AES is a variant of the Rijndael block cipher[5] developed by two Belgian cryptographers, Joan Daemen and Vincent Rijmen, who submitted a proposal[7] to NIST during the AES selection process.[8] Rijndael is a family of ciphers with different key and block sizes. For AES, NIST selected three members of the Rijndael family, each with a block size of 128 bits, but three different key lengths: 128, 192 and 256 bits.

AES has been adopted by the US government. It supersedes the Data Encryption Standard (DES),[9] which was published in 1977. The algorithm described by AES is a symmetric-key algorithm, meaning the same key is used for both encrypting and decrypting the data.

In the United States, AES was announced by the NIST as US FIPS PUB 197 (FIPS 197) on November 26, 2001.[6] This announcement followed a five-year standardization process in which fifteen competing designs were presented and evaluated, before the Rijndael cipher was selected as the most suitable.

AES is included in the ISO/IEC 18033-3 standard. AES became effective as a US federal government standard on May 26, 2002, after approval by US Secretary of Commerce Donald Evans. AES is available in many different encryption packages, and is the first (and only) publicly accessible cipher approved by the US National Security Agency (NSA) for top secret information when used in an NSA approved cryptographic module.

Advanced Encryption Standard (Rijndael)



Visualization of the AES round function

General	
Designers	Joan Daemen, Vincent Rijmen
First published	1998
Derived from	Square
Successors	Anubis, Grand Cru, Kalyna
Certification	AES winner, CRYPTREC, NESSIE, NSA
Cipher detail	
Key sizes	128, 192 or 256 bits[note 1]
Block sizes	128 bits[note 2]
Structure	Substitution–permutation network
Rounds	10, 12 or 14 (depending on key size)
Best public cryptanalysis	

Definitive standards

The Advanced Encryption Standard (AES) is defined in each of:

- FIPS PUB 197: Advanced Encryption Standard (AES)[6]
- ISO/IEC 18033-3: Block ciphers[10]

Description of the ciphers

AES is based on a design principle known as a substitution–permutation network, and is efficient in both software and hardware.[11] Unlike its predecessor DES, AES does not use a Feistel network. AES is a variant of Rijndael, with a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits. By contrast, Rijndael *per se* is specified with block and key sizes that may be any multiple of 32 bits, with a minimum of 128 and a maximum of 256 bits. Most AES calculations are done in a particular finite field.

Attacks have been published that are computationally faster than a full brute-force attack, though none as of 2023 are computationally feasible.[1]

For AES-128, the key can be recovered with a computational complexity of $2^{126.1}$ using the biclique attack. For biclique attacks on AES-192 and AES-256, the computational complexities of $2^{189.7}$ and $2^{254.4}$ respectively apply. Related-key attacks can break AES-192 and AES-256 with complexities $2^{99.5}$ and 2^{176} in both time and data, respectively.[2]

Another attack was blogged[3] and released as a preprint[4] in 2009. This attack is against AES-256 that uses only two related keys and 2^{39} time to recover the complete 256-bit key of a 9-round version, or 2^{45} time for a 10-round version with a stronger type of related subkey attack, or 2^{70} time for an 11-round version.

AES operates on a 4 × 4 column-major order array of 16 bytes b_0, b_1, ..., b_{15} termed the *state*:[note 3]

$$\begin{bmatrix} b_0 & b_4 & b_8 & b_{12} \\ b_1 & b_5 & b_9 & b_{13} \\ b_2 & b_6 & b_{10} & b_{14} \\ b_3 & b_7 & b_{11} & b_{15} \end{bmatrix}$$

The key size used for an AES cipher specifies the number of transformation rounds that convert the input, called the plaintext, into the final output, called the ciphertext. The number of rounds are as follows:

- 10 rounds for 128-bit keys;
- 12 rounds for 192-bit keys;
- 14 rounds for 256-bit keys.

Each round consists of several processing steps, including one that depends on the encryption key itself. A set of reverse rounds are applied to transform ciphertext back into the original plaintext using the same encryption key.

High-level description of the algorithm

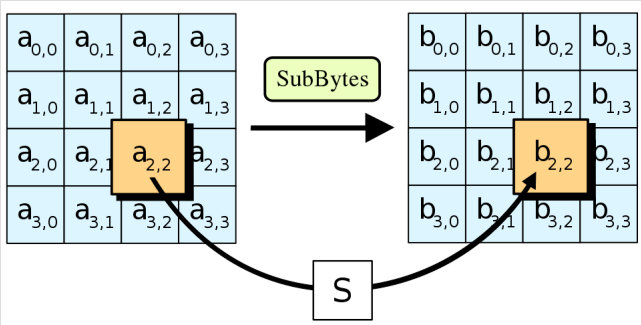
1. `KeyExpansion` – round keys are derived from the cipher key using the AES key schedule. AES requires a separate 128-bit round key block for each round plus one more.
2. Initial round key addition:

1. `AddRoundKey` – each byte of the state is combined with a byte of the round key using bitwise xor.

3. 9, 11 or 13 rounds:
    1. `SubBytes` – a non-linear substitution step where each byte is replaced with another according to a lookup table.
    2. `ShiftRows` – a transposition step where the last three rows of the state are shifted cyclically a certain number of steps.
    3. `MixColumns` – a linear mixing operation which operates on the columns of the state, combining the four bytes in each column.
    4. `AddRoundKey`
4. Final round (making 10, 12 or 14 rounds in total):
    1. `SubBytes`
    2. `ShiftRows`
    3. `AddRoundKey`

The `SubBytes` step

In the `SubBytes` step, each byte $a_{i,j}$ in the *state* array is replaced with a `SubByte` $S(a_{i,j})$ using an 8-bit substitution box. Before round 0, the *state* array is simply the plaintext/input. This operation provides the non-linearity in the cipher. The S-box used is derived from the multiplicative inverse over GF(2^8), known to have good non-linearity properties. To avoid attacks based on simple algebraic properties, the S-box is constructed by combining the inverse function with an invertible affine transformation. The S-box is also chosen to avoid any fixed points (and so is a derangement), i.e., $S(a_{i,j}) \neq a_{i,j}$, and also any opposite fixed points, i.e., $S(a_{i,j}) \oplus a_{i,j} \neq \mathrm{FF}_{16}$. While performing the decryption, the `InvSubBytes` step (the inverse of `SubBytes`) is used, which requires first taking the inverse of the affine transformation and then finding the multiplicative inverse.



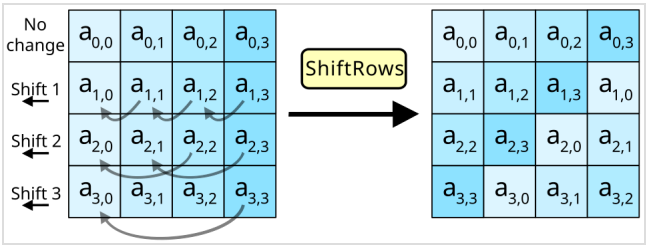
In the `SubBytes` step, each byte in the state is replaced with its entry in a fixed 8-bit lookup table, S; $b_{ij} = S(a_{ij})$.

The `ShiftRows` step

The `ShiftRows` step operates on the rows of the state; it cyclically shifts the bytes in each row by a certain offset. For AES, the first row is left unchanged. Each byte of the second row is shifted one to the left. Similarly, the third and fourth rows are shifted by offsets of two and three respectively.[note 4] In this way, each column of the output state of the `ShiftRows` step is composed of bytes from each column of the input state. The importance of this step is to avoid the columns being encrypted independently, in which case AES would degenerate into four independent block ciphers.

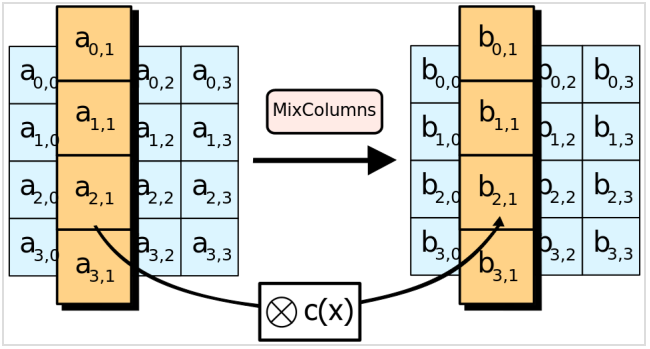
The `MixColumns` step

In the `MixColumns` step, the four bytes of each column of the state are combined using an invertible linear transformation. The `MixColumns` function takes four bytes as input and outputs four bytes, where each input byte affects all four output bytes. Together with `ShiftRows`, `MixColumns` provides diffusion in the cipher.



In the `ShiftRows` step, bytes in each row of the state are shifted cyclically to the left. The number of places each byte is shifted differs incrementally for each row.

During this operation, each column is transformed using a fixed matrix (matrix left-multiplied by column gives new value of column in the state):



In the `MixColumns` step, each column of the state is multiplied with a fixed polynomial $c(x)$.

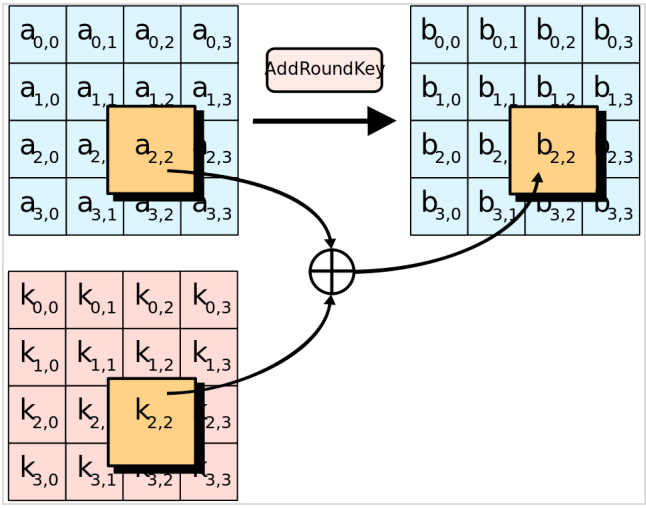
$$\begin{bmatrix} b_{0,j} \\ b_{1,j} \\ b_{2,j} \\ b_{3,j} \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} a_{0,j} \\ a_{1,j} \\ a_{2,j} \\ a_{3,j} \end{bmatrix} \qquad 0 \leq j \leq 3$$

Matrix multiplication is composed of multiplication and addition of the entries. Entries are bytes treated as coefficients of polynomial of order x^7. Addition is simply XOR. Multiplication is modulo irreducible polynomial $x^8 + x^4 + x^3 + x + 1$. If processed bit by bit, then, after shifting, a conditional XOR with $1B_{16}$ should be performed if the shifted value is larger than FF_{16} (overflow must be corrected by subtraction of generating polynomial). These are special cases of the usual multiplication in $GF(2^8)$.

In more general sense, each column is treated as a polynomial over $GF(2^8)$ and is then multiplied modulo $01_{16} \cdot z^4 + 01_{16}$ with a fixed polynomial $c(z) = 03_{16} \cdot z^3 + 01_{16} \cdot z^2 + 01_{16} \cdot z + 02_{16}$. The coefficients are displayed in their hexadecimal equivalent of the binary representation of bit polynomials from $GF(2^8)[x]$. The `MixColumns` step can also be viewed as a multiplication by the shown particular MDS matrix in the finite field $GF(2^8)$. This process is described further in the article Rijndael MixColumns.

The AddRoundKey

In the AddRoundKey step, the subkey is combined with the state. For each round, a subkey is derived from the main key using Rijndael's key schedule; each subkey is the same size as the state. The subkey is added by combining of the state with the corresponding byte of the subkey using bitwise XOR.



In the AddRoundKey step, each byte of the state is combined with a byte of the round subkey using the XOR operation (⊕).

Optimization of the cipher

On systems with 32-bit or larger words, it is possible to speed up execution of this cipher by combining the SubBytes and ShiftRows steps with the MixColumns step by transforming them into a sequence of table lookups. This requires four 256-entry 32-bit tables (together occupying 4096 bytes). A round can then be performed with 16 table lookup operations and 12 32-bit exclusive-or operations, followed by four 32-bit exclusive-or operations in the AddRoundKey step.[12] Alternatively, the table lookup operation can be performed with a single 256-entry 32-bit table (occupying 1024 bytes) followed by circular rotation operations.

Using a byte-oriented approach, it is possible to combine the SubBytes, ShiftRows, and MixColumns steps into a single round operation.[13]

Security

The National Security Agency (NSA) reviewed all the AES finalists, including Rijndael, and stated that all of them were secure enough for US Government non-classified data. In June 2003, the US Government announced that AES could be used to protect classified information:

> The design and strength of all key lengths of the AES algorithm (i.e., 128, 192 and 256) are sufficient to protect classified information up to the SECRET level. TOP SECRET information will require use of either the 192 or 256 key lengths. The implementation of AES in products intended to protect national security systems and/or information must be reviewed and certified by NSA prior to their acquisition and use.[14]

AES has 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys.

Known attacks

For cryptographers, a cryptographic "break" is anything faster than a brute-force attack – i.e., performing one trial decryption for each possible key in sequence . A break can thus include results that are infeasible with current technology. Despite being impractical, theoretical breaks can

sometimes provide insight into vulnerability patterns. The largest successful publicly known brute-force attack against a widely implemented block-cipher encryption algorithm was against a 64-bit RC5 key by distributed.net in 2006.[15]

The key space increases by a factor of 2 for each additional bit of key length, and if every possible value of the key is equiprobable; this translates into a doubling of the average brute-force key search time with every additional bit of key length. This implies that the effort of a brute-force search increases exponentially with key length. Key length in itself does not imply security against attacks, since there are ciphers with very long keys that have been found to be vulnerable.

AES has a fairly simple algebraic framework.[16] In 2002, a theoretical attack, named the "XSL attack", was announced by Nicolas Courtois and Josef Pieprzyk, purporting to show a weakness in the AES algorithm, partially due to the low complexity of its nonlinear components.[17] Since then, other papers have shown that the attack, as originally presented, is unworkable; see XSL attack on block ciphers.

During the AES selection process, developers of competing algorithms wrote of Rijndael's algorithm "we are concerned about [its] use ... in security-critical applications."[18] In October 2000, however, at the end of the AES selection process, Bruce Schneier, a developer of the competing algorithm Twofish, wrote that while he thought successful academic attacks on Rijndael would be developed someday, he "did not believe that anyone will ever discover an attack that will allow someone to read Rijndael traffic."[19]

By 2006, the best known attacks were on 7 rounds for 128-bit keys, 8 rounds for 192-bit keys, and 9 rounds for 256-bit keys.[20]

Until May 2009, the only successful published attacks against the full AES were side-channel attacks on some specific implementations. In 2009, a new related-key attack was discovered that exploits the simplicity of AES's key schedule and has a complexity of 2^{119}. In December 2009 it was improved to $2^{99.5}$.[2] This is a follow-up to an attack discovered earlier in 2009 by Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolić, with a complexity of 2^{96} for one out of every 2^{35} keys.[21] However, related-key attacks are not of concern in any properly designed cryptographic protocol, as a properly designed protocol (i.e., implementational software) will take care not to allow related keys, essentially by constraining an attacker's means of selecting keys for relatedness.

Another attack was blogged by Bruce Schneier[3] on July 30, 2009, and released as a preprint[22] on August 3, 2009. This new attack, by Alex Biryukov, Orr Dunkelman, Nathan Keller, Dmitry Khovratovich, and Adi Shamir, is against AES-256 that uses only two related keys and 2^{39} time to recover the complete 256-bit key of a 9-round version, or 2^{45} time for a 10-round version with a stronger type of related subkey attack, or 2^{70} time for an 11-round version. 256-bit AES uses 14 rounds, so these attacks are not effective against full AES.

The practicality of these attacks with stronger related keys has been criticized,[23] for instance, by the paper on chosen-key-relations-in-the-middle attacks on AES-128 authored by Vincent Rijmen in 2010.[24]

In November 2009, the first known-key distinguishing attack against a reduced 8-round version of AES-128 was released as a preprint.[25] This known-key distinguishing attack is an improvement of the rebound, or the start-from-the-middle attack, against AES-like permutations, which view two consecutive rounds of permutation as the application of a so-called Super-S-box. It works on the 8-round version of AES-128, with a time complexity of 2^{48}, and a memory complexity of 2^{32}. 128-bit AES uses 10 rounds, so this attack is not effective against full AES-128.

The first key-recovery attacks on full AES were by Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger, and were published in 2011.[26] The attack is a biclique attack and is faster than brute force by a factor of about four. It requires $2^{126.2}$ operations to recover an AES-128 key. For AES-192 and AES-256, $2^{190.2}$ and $2^{254.6}$ operations are needed, respectively. This result has been further improved to $2^{126.0}$ for AES-128, $2^{189.9}$ for AES-192, and $2^{254.3}$ for AES-256 by Biaoshuai Tao and Hongjun Wu in a 2015 paper,[27] which are the current best results in key recovery attack against AES.

This is a very small gain, as a 126-bit key (instead of 128 bits) would still take billions of years to brute force on current and foreseeable hardware. Also, the authors calculate the best attack using their technique on AES with a 128-bit key requires storing 2^{88} bits of data. That works out to about 38 trillion terabytes of data, which was more than all the data stored on all the computers on the planet in 2016.[28] A paper in 2015 later improved the space complexity to 2^{56} bits,[27] which is 9007 terabytes (while still keeping a time complexity of approximately 2^{126}).

According to the Snowden documents, the NSA is doing research on whether a cryptographic attack based on tau statistic may help to break AES.[29]

At present, there is no known practical attack that would allow someone without knowledge of the key to read data encrypted by AES when correctly implemented.

Side-channel attacks

Side-channel attacks do not attack the cipher as a black box, and thus are not related to cipher security as defined in the classical context, but are important in practice. They attack implementations of the cipher on hardware or software systems that inadvertently leak data. There are several such known attacks on various implementations of AES.

In April 2005, D. J. Bernstein announced a cache-timing attack that he used to break a custom server that used OpenSSL's AES encryption.[30] The attack required over 200 million chosen plaintexts.[31] The custom server was designed to give out as much timing information as possible (the server reports back the number of machine cycles taken by the encryption operation). However, as Bernstein pointed out, "reducing the precision of the server's timestamps, or eliminating them from the server's responses, does not stop the attack: the client simply uses round-trip timings based on its local clock, and compensates for the increased noise by averaging over a larger number of samples."[30]

In October 2005, Dag Arne Osvik, Adi Shamir and Eran Tromer presented a paper demonstrating several cache-timing attacks against the implementations in AES found in OpenSSL and Linux's `dm-crypt` partition encryption function.[32] One attack was able to obtain an entire AES key after only 800 operations triggering encryptions, in a total of 65 milliseconds. This attack requires the attacker to be able to run programs on the same system or platform that is performing AES.

In December 2009 an attack on some hardware implementations was published that used differential fault analysis and allows recovery of a key with a complexity of 2^{32}.[33]

In November 2010 Endre Bangerter, David Gullasch and Stephan Krenn published a paper which described a practical approach to a "near real time" recovery of secret keys from AES-128 without the need for either cipher text or plaintext. The approach also works on AES-128 implementations that use compression tables, such as OpenSSL.[34] Like some earlier attacks, this one requires the ability to run unprivileged code on the system performing the AES encryption, which may be achieved by malware infection far more easily than commandeering the root account.[35]

In March 2016, C. Ashokkumar, Ravi Prakash Giri and Bernard Menezes presented a side-channel attack on AES implementations that can recover the complete 128-bit AES key in just 6–7 blocks of plaintext/ciphertext, which is a substantial improvement over previous works that require between 100 and a million encryptions.[36] The proposed attack requires standard user privilege and key-retrieval algorithms run under a minute.

Many modern CPUs have built-in hardware instructions for AES, which protect against timing-related side-channel attacks.[37][38]

Quantum attacks

AES-256 is considered to be quantum resistant, as it has similar quantum resistance to AES-128's resistance against traditional, non-quantum, attacks at 128 bits of security. AES-192 and AES-128 are not considered quantum resistant due to their smaller key sizes. AES-192 has a strength of 96 bits against quantum attacks and AES-128 has 64 bits of strength against quantum attacks, making them both insecure.[39][40]

NIST/CSEC validation

The Cryptographic Module Validation Program (CMVP) is operated jointly by the United States Government's National Institute of Standards and Technology (NIST) Computer Security Division and the Communications Security Establishment (CSE) of the Government of Canada. The use of cryptographic modules validated to NIST FIPS 140-2 is required by the United States Government for encryption of all data that has a classification of Sensitive but Unclassified (SBU) or above. From NSTISSP #11, National Policy Governing the Acquisition of Information Assurance: "Encryption products for protecting classified information will be certified by NSA, and encryption products intended for protecting sensitive information will be certified in accordance with NIST FIPS 140-2."[41]

The Government of Canada also recommends the use of FIPS 140 validated cryptographic modules in unclassified applications of its departments.

Although NIST publication 197 ("FIPS 197") is the unique document that covers the AES algorithm, vendors typically approach the CMVP under FIPS 140 and ask to have several algorithms (such as Triple DES or SHA1) validated at the same time. Therefore, it is rare to find cryptographic modules that are uniquely FIPS 197 validated and NIST itself does not generally take the time to list FIPS 197

validated modules separately on its public web site. Instead, FIPS 197 validation is typically just listed as an "FIPS approved: AES" notation (with a specific FIPS 197 certificate number) in the current list of FIPS 140 validated cryptographic modules.

The Cryptographic Algorithm Validation Program (CAVP)[42] allows for independent validation of the correct implementation of the AES algorithm. Successful validation results in being listed on the NIST validations page.[43] This testing is a pre-requisite for the FIPS 140-2 module validation. However, successful CAVP validation in no way implies that the cryptographic module implementing the algorithm is secure. A cryptographic module lacking FIPS 140-2 validation or specific approval by the NSA is not deemed secure by the US Government and cannot be used to protect government data.[41]

FIPS 140-2 validation is challenging to achieve both technically and fiscally.[44] There is a standardized battery of tests as well as an element of source code review that must be passed over a period of a few weeks. The cost to perform these tests through an approved laboratory can be significant (e.g., well over US$30,000)[44] and does not include the time it takes to write, test, document and prepare a module for validation. After validation, modules must be re-submitted and re-evaluated if they are changed in any way. This can vary from simple paperwork updates if the security functionality did not change to a more substantial set of re-testing if the security functionality was impacted by the change.

Test vectors

Test vectors are a set of known ciphers for a given input and key. NIST distributes the reference of AES test vectors as AES Known Answer Test (KAT) Vectors.[note 5]

Performance

High speed and low RAM requirements were some of the criteria of the AES selection process. As the chosen algorithm, AES performed well on a wide variety of hardware, from 8-bit smart cards to high-performance computers.

On a Pentium Pro, AES encryption requires 18 clock cycles per byte (cpb),[45] equivalent to a throughput of about 11 MiB/s for a 200 MHz processor.

On Intel Core and AMD Ryzen CPUs supporting AES-NI instruction set extensions, throughput can be multiple GiB/s.[46] On an Intel Westmere CPU, AES encryption using AES-NI takes about 1.3 cpb for AES-128, and 1.8 cpb for AES-256.[47]

Implementations

See also

- AES modes of operation
- Disk encryption
- Whirlpool – hash function created by Vincent Rijmen and Paulo S. L. M. Barreto
- List of free and open-source software packages

Notes

1. Key sizes of 128, 160, 192, 224, and 256 bits are supported by the Rijndael algorithm, but only the 128, 192, and 256-bit key sizes are specified in the AES standard.
2. Block sizes of 128, 160, 192, 224, and 256 bits are supported by the Rijndael algorithm for each key size, but only the 128-bit block size is specified in the AES standard.
3. Large-block variants of Rijndael use an array with additional columns, but always four rows.
4. Rijndael variants with a larger block size have slightly different offsets. For blocks of sizes 128 bits and 192 bits, the shifting pattern is the same. Row n is shifted left circular by $n-1$ bytes. For a 256-bit block, the first row is unchanged and the shifting for the second, third and fourth row is 1 byte, 3 bytes and 4 bytes respectively—this change only applies for the Rijndael cipher when used with a 256-bit block, as AES does not use 256-bit blocks.
5. The AES Known Answer Test (KAT) Vectors are available in Zip format within the NIST site here (http://csrc.nist.gov/groups/STM/cavp/documents/aes/KAT_AES.zip) Archived (https://web.archive.org/web/20091023001419/http://csrc.nist.gov/groups/STM/cavp/documents/aes/KAT_AES.zip) 2009-10-23 at the Wayback Machine

References

1. "Biclique Cryptanalysis of the Full AES" (https://web.archive.org/web/20160306104007/http://research.microsoft.com/en-us/projects/cryptanalysis/aesbc.pdf) (PDF). Archived from the original (http://research.microsoft.com/en-us/projects/cryptanalysis/aesbc.pdf) (PDF) on March 6, 2016. Retrieved May 1, 2019.
2. Alex Biryukov and Dmitry Khovratovich, *Related-key Cryptanalysis of the Full AES-192 and AES-256*, "Related-key Cryptanalysis of the Full AES-192 and AES-256" (https://eprint.iacr.org/2009/317). Table 1. Archived (https://web.archive.org/web/20090928014006/http://eprint.iacr.org/2009/317) from the original on 2009-09-28. Retrieved 2010-02-16.
3. Bruce Schneier (2009-07-30). "Another New AES Attack" (http://www.schneier.com/blog/archives/2009/07/another_new_aes.html). *Schneier on Security, A blog covering security and security technology*. Archived (https://web.archive.org/web/20091005183132/http://www.schneier.com/blog/archives/2009/07/another_new_aes.html) from the original on 2009-10-05. Retrieved 2010-03-11.
4. Alex Biryukov; Orr Dunkelman; Nathan Keller; Dmitry Khovratovich; Adi Shamir (2009-08-19). "Key Recovery Attacks of Practical Complexity on AES Variants With Up To 10 Rounds" (https://eprint.iacr.org/2009/374). Archived (https://web.archive.org/web/20100128050656/http://eprint.iacr.org/2009/374) from the original on 28 January 2010. Retrieved 2010-03-11.

5. Daemen, Joan; Rijmen, Vincent (March 9, 2003). "AES Proposal: Rijndael" (http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf#page=1) (PDF). National Institute of Standards and Technology. p. 1. Archived (https://web.archive.org/web/20130305143117/http://csrc.nist.gov/archive/aes/rijndael/Rijndael-ammended.pdf#page=1) (PDF) from the original on 5 March 2013. Retrieved 21 February 2013.
6. "Announcing the ADVANCED ENCRYPTION STANDARD (AES)" (https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197-upd1.pdf) (PDF). *Federal Information Processing Standards Publication 197*. United States National Institute of Standards and Technology (NIST). November 26, 2001. Archived (https://web.archive.org/web/20240823165748/https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197-upd1.pdf) (PDF) from the original on August 23, 2024. Retrieved August 26, 2024.
7. Joan Daemen and Vincent Rijmen (September 3, 1999). "AES Proposal: Rijndael" (https://web.archive.org/web/20070203204845/https://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndael.pdf) (PDF). Archived from the original (http://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndael.pdf) (PDF) on February 3, 2007.
8. Schwartz, John (October 3, 2000). "U.S. Selects a New Encryption Technique" (https://www.nytimes.com/2000/10/03/business/technology-us-selects-a-new-encryption-technique.html). *The New York Times*. Archived (https://web.archive.org/web/20170328215407/http://www.nytimes.com/2000/10/03/business/technology-us-selects-a-new-encryption-technique.html) from the original on March 28, 2017.
9. Westlund, Harold B. (2002). "NIST reports measurable success of Advanced Encryption Standard" (https://web.archive.org/web/20071103105501/http://findarticles.com/p/articles/mi_m0IKZ/is_3_107?pnum=2&opg=90984479). *Journal of Research of the National Institute of Standards and Technology*. Archived from the original (http://www.findarticles.com/p/articles/mi_m0IKZ/is_3_107?pnum=2&opg=90984479) on 2007-11-03.
10. "ISO/IEC 18033-3: Information technology – Security techniques – Encryption algorithms – Part 3: Block ciphers" (http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=54531). Archived (https://web.archive.org/web/20131203003348/http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=54531) from the original on 2013-12-03.
11. Bruce Schneier; John Kelsey; Doug Whiting; David Wagner; Chris Hall; Niels Ferguson; Tadayoshi Kohno; et al. (May 2000). "The Twofish Team's Final Comments on AES Selection" (http://www.schneier.com/paper-twofish-final.pdf) (PDF). Archived (https://web.archive.org/web/20100102041117/http://schneier.com/paper-twofish-final.pdf) (PDF) from the original on 2010-01-02.
12. Bertoni, Guido; Breveglieri, Luca; Fragneto, Pasqualina; MacChetti, Marco; Marchesin, Stefano (2003). "Efficient Software Implementation of AES on 32-Bit Platforms" (https://doi.org/10.1007%2F3-540-36400-5_13). *Cryptographic Hardware and Embedded Systems - CHES 2002*. Lecture Notes in Computer Science. Vol. 2523. pp. 159–171. doi:10.1007/3-540-36400-5_13 (https://doi.org/10.1007%2F3-540-36400-5_13). ISBN 978-3-540-00409-7.
13. "byte-oriented-aes – A public domain byte-oriented implementation of AES in C – Google Project Hosting" (https://code.google.com/p/byte-oriented-aes). Archived (https://web.archive.org/web/20130720155538/http://code.google.com/p/byte-oriented-aes/) from the original on 2013-07-20. Retrieved 2012-12-23.
14. Lynn Hathaway (June 2003). "National Policy on the Use of the Advanced Encryption Standard (AES) to Protect National Security Systems and National Security Information" (http://csrc.nist.gov/groups/ST/toolkit/documents/aes/CNSS15FS.pdf) (PDF). Archived (https://web.archive.org/web/20101106122007/http://csrc.nist.gov/groups/ST/toolkit/documents/aes/CNSS15FS.pdf) (PDF) from the original on 2010-11-06. Retrieved 2011-02-15.
15. Ou, George (April 30, 2006). "Is encryption really crackable?" (https://www.zdnet.com/article/is-encryption-really-crackable/). Ziff-Davis. Archived (https://web.archive.org/web/20100808173034/http://www.zdnet.com/blog/ou/is-encryption-really-crackable/204) from the original on August 8, 2010. Retrieved August 7, 2010.

16. "Sean Murphy" (http://www.isg.rhul.ac.uk/~sean/). University of London. Archived (https://web.archive.org/web/20090131145521/http://www.isg.rhul.ac.uk/~sean/) from the original on 2009-01-31. Retrieved 2008-11-02.
17. Bruce Schneier. "AES News, Crypto-Gram Newsletter, September 15, 2002" (http://www.schneier.com/crypto-gram-0209.html). Archived (https://web.archive.org/web/20070707105715/http://www.schneier.com/crypto-gram-0209.html) from the original on 7 July 2007. Retrieved 2007-07-27.
18. Niels Ferguson; Richard Schroeppel; Doug Whiting (2001). "A simple algebraic representation of Rijndael" (https://web.archive.org/web/20061104080748/http://www.macfergus.com/pub/rdalgeq.html). *Proceedings of Selected Areas in Cryptography, 2001, Lecture Notes in Computer Science*. Springer-Verlag. pp. 103–111. CiteSeerX 10.1.1.28.4921 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.28.4921). Archived from the original (http://www.macfergus.com/pub/rdalgeq.html) (PDF/PostScript) on 4 November 2006. Retrieved 2006-10-06.
19. Bruce Schneier, AES Announced (http://www.schneier.com/crypto-gram-0010.html) Archived (https://web.archive.org/web/20090201005720/http://www.schneier.com/crypto-gram-0010.html) 2009-02-01 at the Wayback Machine, October 15, 2000
20. John Kelsey, Stefan Lucks, Bruce Schneier, Mike Stay, David Wagner, and Doug Whiting, *Improved Cryptanalysis of Rijndael*, Fast Software Encryption, 2000 pp213–230 "Academic: Improved Cryptanalysis of Rijndael - Schneier on Security" (http://www.schneier.com/paper-rijndael.html). Archived (https://web.archive.org/web/20070223215007/http://www.schneier.com/paper-rijndael.html) from the original on 2007-02-23. Retrieved 2007-03-06.
21. Nikolić, Ivica (2009). "Distinguisher and Related-Key Attack on the Full AES-256". *Advances in Cryptology - CRYPTO 2009*. Lecture Notes in Computer Science. Vol. 5677. Springer Berlin / Heidelberg. pp. 231–249. doi:10.1007/978-3-642-03356-8_14 (https://doi.org/10.1007%2F978-3-642-03356-8_14). ISBN 978-3-642-03355-1.
22. Alex Biryukov; Orr Dunkelman; Nathan Keller; Dmitry Khovratovich; Adi Shamir (2009-08-19). "Key Recovery Attacks of Practical Complexity on AES Variants With Up To 10 Rounds" (http://eprint.iacr.org/2009/374). Archived (https://web.archive.org/web/20100128050656/http://eprint.iacr.org/2009/374) from the original on 28 January 2010. Retrieved 2010-03-11.
23. Agren, Martin (2012). *On Some Symmetric Lightweight Cryptographic Designs*. Dissertation, Lund University. pp. 38–39.
24. Vincent Rijmen (2010). "Practical-Titled Attack on AES-128 Using Chosen-Text Relations" (http://eprint.iacr.org/2010/337.pdf) (PDF). *IACR Cryptology ePrint Archive*. Archived (https://web.archive.org/web/20100702184311/http://eprint.iacr.org/2010/337.pdf) (PDF) from the original on 2010-07-02.
25. Henri Gilbert; Thomas Peyrin (2009-11-09). "Super-Sbox Cryptanalysis: Improved Attacks for AES-like permutations" (http://eprint.iacr.org/2009/531). *IACR Cryptology ePrint Archive*. Archived (https://web.archive.org/web/20100604095754/http://eprint.iacr.org/2009/531) from the original on 2010-06-04. Retrieved 2010-03-11.
26. Bogdanov, Andrey; Khovratovich, Dmitry; Rechberger, Christian (2011). "Biclique Cryptanalysis of the Full AES". In Lee, Dong Hoon; Wang, Xiaoyun (eds.). *Advances in Cryptology – ASIACRYPT 2011*. Lecture Notes in Computer Science. Vol. 7073. pp. 344–371. doi:10.1007/978-3-642-25385-0_19 (https://doi.org/10.1007%2F978-3-642-25385-0_19). ISBN 978-3-642-25385-0.
27. Tao, Biaoshuai; Wu, Hongjun (2015). "Improving the Biclique Cryptanalysis of AES". In Foo, Ernest; Stebila, Douglas (eds.). *Information Security and Privacy*. Lecture Notes in Computer Science. Vol. 9144. pp. 39–56. doi:10.1007/978-3-319-19962-7_3 (https://doi.org/10.1007%2F978-3-319-19962-7_3). ISBN 978-3-319-19962-7.
28. Jeffrey Goldberg (2011-08-18). "AES Encryption isn't Cracked" (https://web.archive.org/web/20150108165723/https://blog.agilebits.com/2011/08/18/aes-encryption-isnt-cracked/). Archived from the original (https://blog.agilebits.com/2011/08/18/aes-encryption-isnt-cracked/) on 8 January 2015. Retrieved 30 December 2014.

29. "Prying Eyes: Inside the NSA's War on Internet Security" (http://www.spiegel.de/international/germany/inside-the-nsa-s-war-on-internet-security-a-1010361.html). *Spiegel Online*. Hamburg, Germany. 28 December 2014. Archived (https://web.archive.org/web/20150124202809/http://www.spiegel.de/international/germany/inside-the-nsa-s-war-on-internet-security-a-1010361.html) from the original on 24 January 2015. Retrieved 4 September 2015.
30. "Index of formal scientific papers" (http://cr.yp.to/papers.html#cachetiming). Cr.yp.to. Archived (https://web.archive.org/web/20080917042758/http://cr.yp.to/papers.html#cachetiming) from the original on 2008-09-17. Retrieved 2008-11-02.
31. Bruce Schneier (17 May 2005). "AES Timing Attack" (http://www.schneier.com/blog/archives/2005/05/aes_timing_atta_1.html). Archived (https://web.archive.org/web/20070212015727/http://www.schneier.com/blog/archives/2005/05/aes_timing_atta_1.html) from the original on 12 February 2007. Retrieved 2007-03-17.
32. Dag Arne Osvik; Adi Shamir; Eran Tromer (2005-11-20). "Cache Attacks and Countermeasures: the Case of AES" (http://www.wisdom.weizmann.ac.il/~tromer/papers/cache.pdf) (PDF). *The Cryptographer's Track at RSA Conference 2006*. Lecture Notes in Computer Science. Vol. 3860. pp. 1–20. doi:10.1007/11605805_1 (https://doi.org/10.1007%2F11605805_1). ISBN 978-3-540-31033-4. Archived (https://web.archive.org/web/20060619221046/http://www.wisdom.weizmann.ac.il/%7Etromer/papers/cache.pdf) (PDF) from the original on 2006-06-19. Retrieved 2008-11-02.
33. Dhiman Saha; Debdeep Mukhopadhyay; Dipanwita RoyChowdhury. "A Diagonal Fault Attack on the Advanced Encryption Standard" (http://eprint.iacr.org/2009/581.pdf) (PDF). *IACR Cryptology ePrint Archive*. Archived (https://web.archive.org/web/20091222070135/http://eprint.iacr.org/2009/581.pdf) (PDF) from the original on 22 December 2009. Retrieved 2009-12-08.
34. Endre Bangerter; David Gullasch & Stephan Krenn (2010). "Cache Games – Bringing Access-Based Cache Attacks on AES to Practice" (http://eprint.iacr.org/2010/594.pdf) (PDF). *IACR Cryptology ePrint Archive*. Archived (https://web.archive.org/web/20101214092512/http://eprint.iacr.org/2010/594.pdf) (PDF) from the original on 2010-12-14.
35. "Breaking AES-128 in realtime, no ciphertext required" (http://news.ycombinator.com/item?id=1937902). Hacker News. Archived (https://web.archive.org/web/20111003193004/http://news.ycombinator.com/item?id=1937902) from the original on 2011-10-03. Retrieved 2012-12-23.
36. Ashokkumar, C.; Giri, Ravi Prakash; Menezes, Bernard (12 May 2016). *Highly Efficient Algorithms for AES Key Retrieval in Cache Access Attacks*. 2016 IEEE European Symposium on Security and Privacy (EuroS&P). Saarbruecken, Germany. pp. 261–275. doi:10.1109/EuroSP.2016.29 (https://doi.org/10.1109%2FEuroSP.2016.29).
37. Mowery, Keaton; Keelveedhi, Sriram; Shacham, Hovav (19 October 2012). *Are AES x86 cache timing attacks still feasible?* (https://web.archive.org/web/20170809152309/http://cseweb.ucsd.edu/~kmowery/papers/aes-cache-timing.pdf) (PDF). CCS'12: the ACM Conference on Computer and Communications Security. Raleigh, North Carolina, USA. pp. 19–24. doi:10.1145/2381913.2381917 (https://doi.org/10.1145%2F2381913.2381917). Archived from the original (https://cseweb.ucsd.edu/~kmowery/papers/aes-cache-timing.pdf) (PDF) on 2017-08-09.
38. "Securing the Enterprise with Intel AES-NI" (https://www.intel.in/content/dam/doc/white-paper/enterprise-security-aes-ni-white-paper.pdf) (PDF). *Intel Corporation*. Archived (https://web.archive.org/web/20130331041411/http://www.intel.in/content/dam/doc/white-paper/enterprise-security-aes-ni-white-paper.pdf) (PDF) from the original on 2013-03-31. Retrieved 2017-07-26.
39. Bonnetain, Xavier; Naya-Plasencia, María; Schrottenloher, André (11 June 2019). "Quantum Security Analysis of AES" (https://inria.hal.science/hal-02397049/document). *IACR Transactions on Symmetric Cryptology*. **2019** (2): 55–93. doi:10.13154/tosc.v2019.i2.55-93 (https://doi.org/10.13154%2Ftosc.v2019.i2.55-93).
40. O'Shea, Dan (April 26, 2022). "AES-256 joins the quantum resistance" (https://www.fierceelectronics.com/electronics/aes-256-joins-quantum-resistance). *Fierce Electronics*. Retrieved September 26, 2023.

41. "NSTISSP No. 11, Revised Fact Sheet, National Information Assurance Acquisition Policy" (https://web.archive.org/web/20120421103818/http://www.cnss.gov/Assets/pdf/nstissp_11_fs.pdf) (PDF). Archived from the original (http://www.cnss.gov/Assets/pdf/nstissp_11_fs.pdf) (PDF) on 2012-04-21. Retrieved 2012-05-29.
42. "NIST.gov – Computer Security Division – Computer Security Resource Center" (http://csrc.nist.gov/groups/STM/cavp/index.html). Csrc.nist.gov. Archived (https://web.archive.org/web/20130102044410/http://csrc.nist.gov/groups/STM/cavp/index.html) from the original on 2013-01-02. Retrieved 2012-12-23.
43. "Validated FIPS 140-1 and FIPS 140-2 Cryptographic Modules" (https://web.archive.org/web/20141226152243/http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140val-all.htm). Archived from the original (http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/140val-all.htm) on 2014-12-26. Retrieved 2014-06-26.
44. OpenSSL, openssl@openssl.org. "OpenSSL's Notes about FIPS certification" (https://web.archive.org/web/20130102203126/http://www.openssl.org/docs/fips/fipsnotes.html). Openssl.org. Archived from the original (https://openssl.org/docs/fips/fipsnotes.html) on 2013-01-02. Retrieved 2012-12-23.
45. Schneier, Bruce; Kelsey, John; Whiting, Doug; Wagner, David; Hall, Chris; Ferguson, Niels (1999-02-01). "Performance Comparisons of the AES submissions" (http://www.schneier.com/paper-aes-performance.pdf) (PDF). Archived (https://web.archive.org/web/20110622084238/http://www.schneier.com/paper-aes-performance.pdf) (PDF) from the original on 2011-06-22. Retrieved 2010-12-28.
46. "AMD Ryzen 7 1700X Review" (https://www.vortez.net/articles_pages/amd_ryzen_7_1700x_review,7.html).
47. "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) (PDF). May 2010.

- Courtois, Nicolas; Pieprzyk, Josef (2003). "Cryptanalysis of Block Ciphers with Overdefined Systems of Equations" (https://books.google.com/books?id=OZ1qCQAAQBAJ&pg=PA268). In Zheng, Yuliang (ed.). *Advances in Cryptology – ASIACRYPT 2002: 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1–5, 2002, Proceedings*. Springer. pp. 268–287. ISBN 978-3-540-36178-7.
- Daemen, Joan; Rijmen, Vincent (2002). *The Design of Rijndael: AES – The Advanced Encryption Standard* (https://books.google.com/books?id=tfjd6icCUoYC&pg=PR4). Springer. ISBN 978-3-540-42580-9.
- Paar, Christof; Pelzl, Jan (2009). *Understanding Cryptography: A Textbook for Students and Practitioners* (https://books.google.com/books?id=f24wFELSzkoC&pg=PA87). Springer. pp. 87–122. ISBN 978-3-642-04101-3. alternate link (https://archive.today/20130105232834/http://wiki.crypto.rub.de/Buch/sample_chapters.php) (companion web site contains online lectures on AES)

External links

- "256bit key – 128bit block – AES" (http://embeddedsw.net/Cipher_Reference_Home.html). *Cryptography – 256 bit Ciphers: Reference source code and submissions to international cryptographic designs contests*. EmbeddedSW.
- "Advanced Encryption Standard (AES)" (https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197-upd1.pdf) (PDF). *Federal Information Processing Standards*. 26 November 2001. doi:10.6028/NIST.FIPS.197 (https://doi.org/10.6028%2FNIST.FIPS.197). 197.
- AES algorithm archive information – (old, unmaintained) (http://csrc.nist.gov/archive/aes/rijndael/wsdindex.html)

- "Part 3: Block ciphers" (https://webstore.iec.ch/preview/info_isoiec18033-3%7Bed2.0%7Den.pdf) (PDF). *Information technology – Security techniques – Encryption algorithms* (2nd ed.). ISO. 2010-12-15. ISO/IEC 18033-3:2010(E). Archived (https://ghostarchive.org/archive/20221009/https://webstore.iec.ch/preview/info_isoiec18033-3%7Bed2.0%7Den.pdf) (PDF) from the original on 2022-10-09.

- Animation of Rijndael (http://www.formaestudio.com/rijndaelinspector/archivos/Rijndael_Animation_v4_eng.swf) – AES deeply explained and animated using Flash (by Enrique Zabala / University ORT / Montevideo / Uruguay). This animation (in English, Spanish, and German) is also part of CrypTool 1 (menu Indiv. Procedures → Visualization of Algorithms → AES).
- HTML5 Animation of Rijndael (https://formaestudio.com/rijndaelinspector/archivos/Rijndael_Animation_v4_eng-html5.html) – Same Animation as above made in HTML5.
- AES Demo in Excel (https://infsec.de/aes-in-excel-eng/) - Example implementation and demonstration in Excel (without macros) by Tim Wambach.
