Advanced Encryption Standard

The **Advanced Encryption Standard** (**AES**), also known as **Rijndael**[4][5] (its original name), is a specification for the encryption of electronic data established by the U.S. National Institute of Standards and Technology (NIST) in 2001.[6]

AES is a subset of the Rijndael cipher[5] developed by two Belgian cryptographers, Joan Daemen and Vincent Rijmen, who submitted a proposal to NIST during the AES selection process.[7] 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 U.S. government and is now used worldwide. It supersedes the Data Encryption Standard (DES),[8] 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 U.S. 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 (see Advanced Encryption Standard process for more details).

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

The name *Rijndael* (Dutch pronunciation: [ˈrɛindaːl]) is a play on the names of the two inventors (Joan Daemen and Vincent Rijmen).

1 Definitive standards

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

- FIPS PUB 197: Advanced Encryption Standard (AES)[6]
- ISO/IEC 18033-3: Information technology — Security techniques — Encryption algorithms — Part 3: Block ciphers [9]

2 Description of the cipher

AES is based on a design principle known as a substitution-permutation network, combination of both substitution and permutation, and is fast in both software and hardware.[10] Unlike its predecessor DES, AES does not use a Feistel network. AES is a variant of Rijndael which has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits. By contrast, the Rijndael specification *per se* is specified with block and key sizes that may be any multiple of 32 bits, both with a minimum of 128 and a maximum of 256 bits.

AES operates on a 4 × 4 column-major order matrix of bytes, termed the *state*, although some versions of Rijndael have a larger block size and have additional columns in the state. Most AES calculations are done in a special finite field.

For instance, if there are 16 bytes, $b_0, b_1, ..., b_{15}$, these bytes are represented as this matrix:

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

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

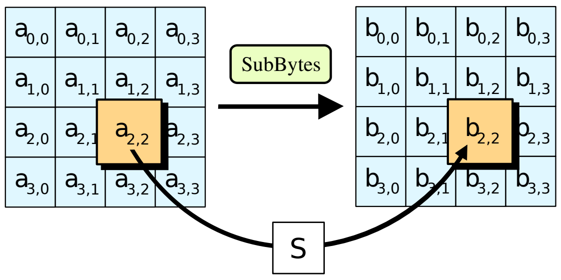
- 10 cycles of repetition for 128-bit keys.
- 12 cycles of repetition for 192-bit keys.
- 14 cycles of repetition for 256-bit keys.

Each round consists of several processing steps, each containing four similar but different stages, 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.

2.1 High-level description of the algorithm

1. KeyExpansions—round keys are derived from the cipher key using Rijndael's key schedule. AES requires a separate 128-bit round key block for each round plus one more.
2. InitialRound
   (a) AddRoundKey—each byte of the state is combined with a block of the round key using bitwise xor.
3. Rounds
   (a) SubBytes—a non-linear substitution step where each byte is replaced with another according to a lookup table.
   (b) ShiftRows—a transposition step where the last three rows of the state are shifted cyclically a certain number of steps.
   (c) MixColumns—a mixing operation which operates on the columns of the state, combining the four bytes in each column.
   (d) AddRoundKey
4. Final Round (no MixColumns)
   (a) SubBytes
   (b) ShiftRows
   (c) AddRoundKey.

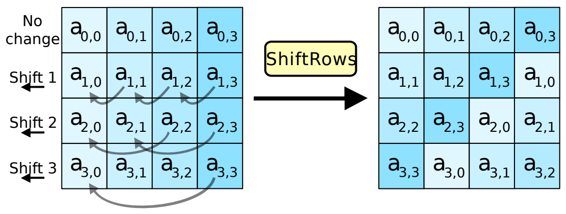
2.2 The SubBytes step



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

In the SubBytes step, each byte $a_{i,j}$ in the *state* matrix is replaced with a SubByte $S(a_{i,j})$ using an 8-bit substitution box, the Rijndael S-box. 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., $S(a_{i,j}) \oplus a_{i,j} \neq \text{0xFF}$. While performing the decryption, Inverse SubBytes step is used, which requires first taking the affine transformation and then finding the multiplicative inverse (just reversing the steps used in SubBytes step).

2.3 The ShiftRows step



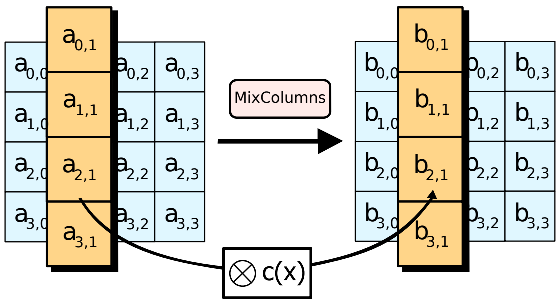
In the ShiftRows step, bytes in each row of the state are shifted cyclically to the left. The number of places each byte is shifted differs for each row.

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. 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. In this way, each column of the output state of the ShiftRows step is composed of bytes from each column of the input state. (Rijndael variants with a larger block size have slightly different offsets). 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. The importance of this step is to avoid the columns being linearly independent, in which case, AES degenerates into four independent block ciphers.

2.4 The MixColumns step

Main article: Rijndael mix columns
In the MixColumns step, the four bytes of 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.

During this operation, each column is transformed using a fixed matrix (matrix 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 $c(x)$.

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

Matrix multiplication is composed of multiplication and addition of the entries. Entries are 8 bit 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 0x1B should be performed if the shifted value is larger than 0xFF (overflow must be corrected by subtraction of generating polynomial). These are special cases of the usual multiplication in $\mathbf{GF}(2^8)$.

In more general sense, each column is treated as a polynomial over $\mathbf{GF}(2^8)$ and is then multiplied modulo x^4+1 with a fixed polynomial $c(x) = 0\text{x}03 \cdot x^3 + x^2 + x + 0\text{x}02$. The coefficients are displayed in their hexadecimal equivalent of the binary representation of bit polynomials from $\mathbf{GF}(2)[x]$. The MixColumns step can also be viewed as a multiplication by the shown particular MDS matrix in the finite field $\mathbf{GF}(2^8)$. This process is described further in the article Rijndael mix columns.

2.5 The AddRoundKey step

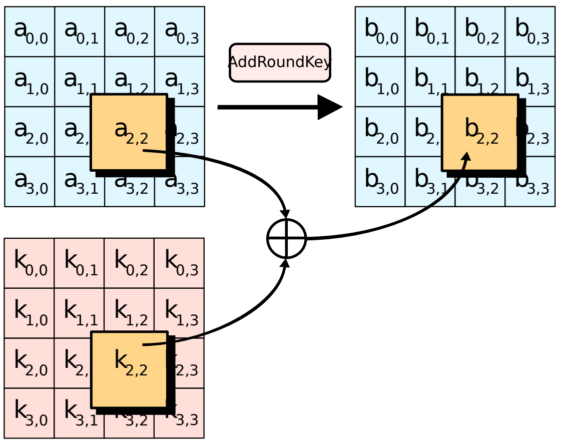
In the AddRoundKey step, the subkey is combined with the state. For each round, a subkey is derived from the main key using Rijndael's key schedule; each subkey is the same size as the state. The subkey is added by combining each byte of the state with the corresponding byte of the subkey using bitwise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

2.6 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, and utilizes a total of four kilobytes (4096 bytes) of memory — one kilobyte for each table. A round can then be done with 16 table lookups and 12 32-bit exclusive-or operations, followed by four 32-bit exclusive-or operations in the AddRoundKey step.[11]

If the resulting four-kilobyte table size is too large for a given target platform, the table lookup operation can be performed with a single 256-entry 32-bit (i.e. 1 kilobyte) table by the use of circular rotates.

Using a byte-oriented approach, it is possible to combine the SubBytes, ShiftRows, and MixColumns steps into a single round operation.[12]



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

3 Security

Until May 2009, the only successful published attacks against the full AES were side-channel attacks on some specific implementations. The National Security Agency (NSA) reviewed all the AES finalists, including Rijndael, and stated that all of them were secure enough for U.S. Government non-classified data. In June 2003, the U.S. 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.[13]

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

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.[14]

3.1 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 (see Cryptanalysis). A break can thus include results that are infeasible with current technology; however, theoretical though impractical breaks can illuminate vulnerability patterns in some cases. The largest successful publicly known brute force attack against any block-cipher encryption was against a 64-bit RC5 key by distributed.net in 2006.[15]

The key space to be searched by brute force increases by a factor of 2 for each additional bit of key length (assuming, importantly, random choice of keys) which alone increases the degree of difficulty for a brute force search very rapidly. Mere key length is not, however, regarded as sufficient for security against attack, for there are ciphers with very long keys which have been found vulnerable.

AES has a fairly simple algebraic description.[16] In 2002, a theoretical attack, termed the "XSL attack", was announced by Nicolas Courtois and Josef Pieprzyk, purporting to show a weakness in the AES algorithm due to its simple description.[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, "...we are concerned about [its] use...in security-critical applications.”[18] However, in October 2000 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 does not “believe that anyone will ever discover an attack that will allow someone to read Rijndael traffic.”[19]

On July 1, 2009, Bruce Schneier blogged[20] about a related-key attack on the 192-bit and 256-bit versions of AES, discovered by Alex Biryukov and Dmitry Khovratovich,[21] which exploits AES’s somewhat simple key schedule and has a complexity of 2^{119}. In December 2009 it was improved to $2^{99.5}$. 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.[22] 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, forcing key choice to be as random as possible.

Another attack was blogged by Bruce Schneier[23] on July 30, 2009 and released as a preprint[24] 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 aren't effective against full AES.

The practicality of these attacks with stronger related keys has been criticized,[25] for instance, by the paper on “chosen-key-relations-in-the-middle” attacks on AES-128 authored by Vincent Rijmen in 2010.[26]

In November 2009, the first known-key distinguishing attack against a reduced 8-round version of AES-128 was released as a preprint.[27] 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-Sbox. 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 isn't effective against full AES-128.

The first key-recovery attacks on full AES were due to Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger, and were published in 2011.[28] 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,[29] 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 (though this has later been improved to 2^{56},[29] which is 9 petabytes). That works out to about 38 trillion terabytes of data, which is more than all the data stored on all the computers on the planet in 2016. As such this is a seriously impractical attack which has no practical implication on AES security.[30]

According to the Snowden documents, the NSA is doing research on whether a cryptographic attack based on tau statistic may help to break AES.[31]

At present, there are no known practical attacks that would allow anyone to read correctly implemented AES encrypted data.

3.2 Side-channel attacks

Side-channel attacks do not attack the underlying cipher, and thus are not related to cipher security in the usually discussed context, though they may be important in practice. They attack implementations of the cipher on hardware or software systems which inadvertently leak data. There are several such known attacks on certain 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.[32] The attack required over 200 million chosen plaintexts.[33] 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.”[32]

In October 2005, Dag Arne Osvik, Adi Shamir and Eran Tromer presented a paper demonstrating several cache-timing attacks against AES.[34] 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}.[35]

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.[36] 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.[37]

4 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.”[38]

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)[39] allows for independent validation of the correct implementation of the AES algorithm at a reasonable cost. Successful validation results in being listed on the NIST validations page. This testing is a pre-requisite for the FIPS 140-2 module validation described below. 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.[38]

FIPS 140-2 validation is challenging to achieve both technically and fiscally.[40] 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 $30,000 US)[40] 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.

5 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 (in ZIP format).

6 Performance

High speed and low RAM requirements were 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,[41] equivalent to a throughput of about 11 MB/s for a 200 MHz processor. On a 1.7 GHz Pentium M throughput is about 60 MB/s.

On Intel Core i3/i5/i7 and AMD APU and FX CPUs supporting AES-NI instruction set extensions, throughput can be over 700 MB/s per thread.[42]

7 Implementations

Main article: AES implementations

8 See also

- Disk encryption
- Whirlpool – hash function created by Vincent Rijmen and Paulo S. L. M. Barreto
- Rijndael key schedule
- Rijndael S-box

9 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, but only the 128-bit block size is specified in the AES standard.

[3] “Biclique Cryptanalysis of the Full AES” (PDF). Retrieved July 23, 2013.

[4] “Rijndael”. Retrieved March 9, 2015.

[5] Daemen, Joan; Rijmen, Vincent (March 9, 2003). “AES Proposal: Rijndael” (PDF). National Institute of Standards and Technology. p. 1. Retrieved 21 February 2013.

[6] “Announcing the ADVANCED ENCRYPTION STANDARD (AES)" (PDF). *Federal Information Processing Standards Publication 197*. United States National Institute of Standards and Technology (NIST). November 26, 2001. Retrieved October 2, 2012.

[7] John Schwartz (October 3, 2000). “U.S. Selects a New Encryption Technique”. *New York Times*.

[8] Westlund, Harold B. (2002). “NIST reports measurable success of Advanced Encryption Standard”. *Journal of Research of the National Institute of Standards and Technology*.

[9] “ISO/IEC 18033-3: Information technology — Security techniques — Encryption algorithms — Part 3: Block ciphers”.

[10] 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” (PDF).

[11] “Efficient software implementation of AES on 32-bit platforms”. Lecture Notes in Computer Science: 2523. 2003

[12] “byte-oriented-aes - A public domain byte-oriented implementation of AES in C - Google Project Hosting”. Code.google.com. Retrieved 2012-12-23.

[13] Lynn Hathaway (June 2003). “National Policy on the Use of the Advanced Encryption Standard (AES) to Protect National Security Systems and National Security Information” (PDF). Retrieved 2011-02-15.

[14] John Kelsey, Stefan Lucks, Bruce Schneier, Mike Stay, David Wagner, and Doug Whiting, *Improved Cryptanalysis of Rijndael*, Fast Software Encryption, 2000 pp213–230

[15] Ou, George (April 30, 2006). “Is encryption really crackable?". Ziff-Davis. Archived from the original on August 7, 2010. Retrieved August 7, 2010.

[16] “Sean Murphy”. University of London. Retrieved 2008-11-02.

[17] Bruce Schneier. “AES News, Crypto-Gram Newsletter, September 15, 2002”. Archived 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”. *Proceedings of Selected Areas in Cryptography, 2001, Lecture Notes in Computer Science*. Springer-Verlag. pp. 103–111. Archived from the original (PDF/PostScript) on 4 November 2006. Retrieved 2006-10-06.

[19] Bruce Schneier, AES Announced, October 15, 2000

[20] Bruce Schneier (2009-07-01). “New Attack on AES”. *Schneier on Security, A blog covering security and security technology*. Archived from the original on 8 February 2010. Retrieved 2010-03-11.

[21] Biryukov, Alex; Khovratovich, Dmitry (2009-12-04). “Related-key Cryptanalysis of the Full AES-192 and AES-256”. Retrieved 2010-03-11.

[22] Nikolić, Ivica (2009). “Distinguisher and Related-Key Attack on the Full AES-256”. *Advances in Cryptology – CRYPTO 2009*. Springer Berlin / Heidelberg. pp. 231–249. doi:10.1007/978-3-642-03356-8_14. ISBN 978-3-642-03355-1.

[23] Bruce Schneier (2009-07-30). “Another New AES Attack”. *Schneier on Security, A blog covering security and security technology*. Retrieved 2010-03-11.

[24] 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". Archived from the original on 28 January 2010. Retrieved 2010-03-11.

[25] Agren, Martin (2012). *On Some Symmetric Lightweight Cryptographic Designs*. Dissertation, Lund University. pp. 38–39.

[26] Vincent Rijmen (2010). "Practical-Titled Attack on AES-128 Using Chosen-Text Relations" (PDF).

[27] Henri Gilbert; Thomas Peyrin (2009-11-09). "Super-Sbox Cryptanalysis: Improved Attacks for AES-like permutations". Retrieved 2010-03-11.

[28] Andrey Bogdanov; Dmitry Khovratovich & Christian Rechberger (2011). "Biclique Cryptanalysis of the Full AES" (PDF).

[29] Biaoshuai Tao & Hongjun Wu (2015). "Improving the Biclique Cryptanalysis of AES".

[30] Jeffrey Goldberg. "AES Encryption isn't Cracked". Retrieved 30 December 2014.

[31] SPIEGEL ONLINE, Hamburg, Germany (28 December 2014). "Inside the NSA's War on Internet Security". *SPIEGEL ONLINE*. Retrieved 4 September 2015.

[32] "Index of formal scientific papers". Cr.yp.to. Retrieved 2008-11-02.

[33] Bruce Schneier. "AES Timing Attack". Archived from the original on 12 February 2007. Retrieved 2007-03-17.

[34] Dag Arne Osvik; Adi Shamir; Eran Tromer (2005-11-20). "Cache Attacks and Countermeasures: the Case of AES" (PDF). Retrieved 2008-11-02.

[35] Dhiman Saha; Debdeep Mukhopadhyay; Dipanwita RoyChowdhury. "A Diagonal Fault Attack on the Advanced Encryption Standard" (PDF). Archived (PDF) from the original on 22 December 2009. Retrieved 2009-12-08.

[36] Endre Bangerter; David Gullasch & Stephan Krenn (2010). "Cache Games – Bringing Access-Based Cache Attacks on AES to Practice" (PDF).

[37] "Breaking AES-128 in realtime, no ciphertext required | Hacker News". News.ycombinator.com. Retrieved 2012-12-23.

[38] http://www.cnss.gov/Assets/pdf/nstissp_11_fs.pdf

[39] "NIST.gov - Computer Security Division - Computer Security Resource Center". Csrc.nist.gov. Retrieved 2012-12-23.

[40] OpenSSL, openssl@openssl.org. "OpenSSL's Notes about FIPS certification". Openssl.org. Retrieved 2012-12-23.

[41] Schneier, Bruce; Kelsey, John; Whiting, Doug; Wagner, David; Hall, Chris; Ferguson, Niels (1999-02-01). "Performance Comparisons of the AES submissions" (PDF). Retrieved 2010-12-28.

[42] McWilliams, Grant (6 July 2011). "Hardware AES Showdown - VIA Padlock vs. Intel AES-NI vs. AMD Hexacore". Retrieved 2013-08-28.

10 References

- Nicolas Courtois, Josef Pieprzyk, "Cryptanalysis of Block Ciphers with Overdefined Systems of Equations". pp267–287, ASIACRYPT 2002.
- Joan Daemen, Vincent Rijmen, "The Design of Rijndael: AES – The Advanced Encryption Standard." Springer, 2002. ISBN 3-540-42580-2.
- Christof Paar, Jan Pelzl, "The Advanced Encryption Standard", Chapter 4 of "Understanding Cryptography, A Textbook for Students and Practitioners". (companion web site contains online lectures on AES), Springer, 2009.

11 External links

- 256bit Ciphers - AES Reference implementation and derived code
- FIPS PUB 197: the official AES standard (PDF file)
- AES algorithm archive information – (old, unmaintained)
- Preview of ISO/IEC 18033-3
- Animation of Rijndael – 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).

12 Text and image sources, contributors, and licenses

12.1 Text

12.2 Images

12.3 Content license