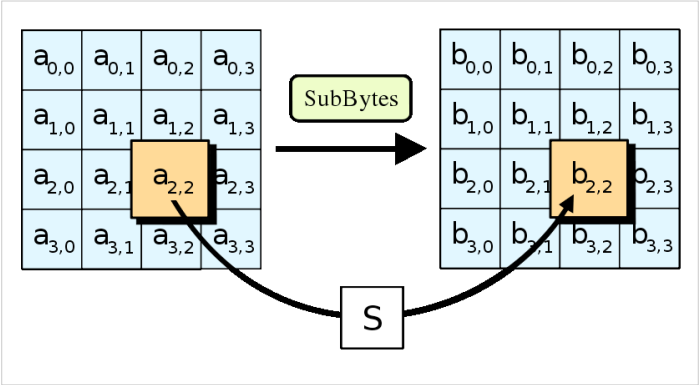
"Advanced Encryption Standard"

Wikipedia



The `SubBytes` step, one of four stages in a round of AES

General	
Designers	Vincent Rijmen, Joan Daemen
First published	1998
Derived from	Square
Successors	Anubis, Grand Cru
Certification	AES winner, CRYPTREC, NESSIE, NSA
Cipher detail	
Key sizes	128, 192 or 256 bits[1]
Block sizes	128 bits[2]
Structure	Substitution-permutation network
Rounds	10, 12 or 14 (depending on key size)
Best public cryptanalysis	
A related-key attack can break 256-bit AES with a complexity of $2^{99.5}$, which is faster than brute force but is still infeasible. 192-bit AES can also be defeated in a similar manner, but at a complexity of 2^{176} which is also infeasible. 128-bit AES is not affected by this attack.	

In cryptography, the **Advanced Encryption Standard** (**AES**) is a symmetric-key encryption standard adopted by the U.S. government. The standard comprises three block ciphers, AES-128, AES-192 and AES-256, adopted from a larger collection originally published as **Rijndael.** Each of these ciphers has a 128-bit block size, with key sizes of 128, 192 and 256 bits, respectively. The AES ciphers have been analyzed extensively and are now used worldwide, as was the case with its predecessor,[3] the Data Encryption Standard (DES).

AES was announced by National Institute of Standards and Technology (NIST) as U.S. FIPS PUB 197 (FIPS 197) on November 26, 2001 after a 5-year standardization process in which fifteen competing designs were presented and evaluated before Rijndael was selected as the most suitable (see Advanced Encryption Standard process for more details). It became effective as a Federal government standard on May 26, 2002 after approval by the Secretary of Commerce. It is available in many different encryption packages. AES is the first publicly accessible and open cipher approved by the NSA for top secret information (see Security of AES, below).

The Rijndael cipher was developed by two Belgian cryptographers, Joan Daemen and Vincent Rijmen, and submitted by them to the AES selection process.[4] **Rijndael** (Dutch pronunciation: [ˈrɛinda:l][5]) is a wordplay based upon the names of the two inventors.

Description of the cipher

AES is based on a design principle known as a Substitution permutation network. It is fast in both software and hardware.[6] Unlike its predecessor, DES, AES does not use a Feistel network.

AES has a fixed block size of 128 bits and a key size of 128, 192, or 256 bits, whereas Rijndael can be specified with block and key sizes in any multiple of 32 bits, with a minimum of 128 bits. The blocksize has a maximum of 256 bits, but the keysize has no theoretical maximum.

AES operates on a 4×4 matrix of bytes, termed the *state* (versions of Rijndael with a larger block size have additional columns in the state). Most AES calculations are done in a special finite field.

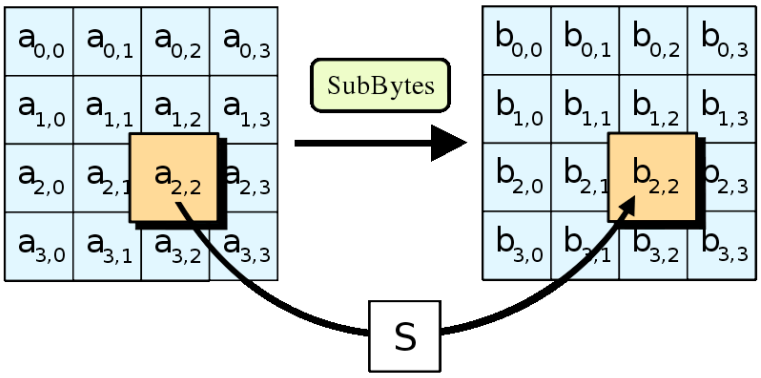
The AES cipher is specified as a number of repetitions of transformation rounds that convert the input plaintext into the final output of ciphertext. Each round consists of several processing steps, including one that depends on the encryption key. A set of reverse rounds are applied to transform ciphertext 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 Rijndael's key schedule
2. Initial Round
    1. `AddRoundKey`—each byte of the state is combined with the round key using bitwise xor
3. 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 each row of the state is shifted cyclically a certain number of steps.
    3. `MixColumns`—a mixing operation which operates on the columns of the state, combining the four bytes in each column.
    4. `AddRoundKey`
4. Final Round (no `MixColumns`)
    1. `SubBytes`
    2. `ShiftRows`
    3. `AddRoundKey`

The `SubBytes` step

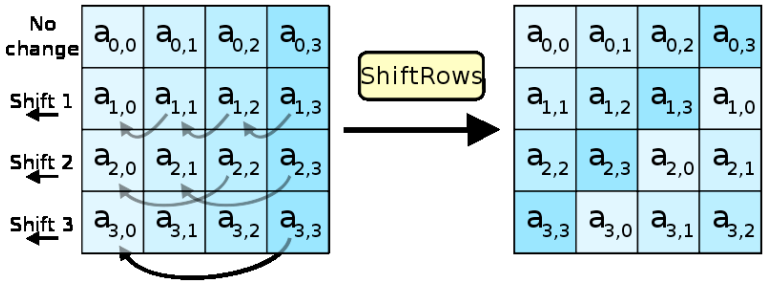
In the `SubBytes` step, each byte in the matrix is updated using an 8-bit substitution box, the Rijndael S-box. This operation provides the non-linearity in the cipher. The S-box used is derived from the multiplicative inverse over **GF**(2^8), known to have good non-linearity properties. To avoid attacks 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), and also any opposite fixed points.



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

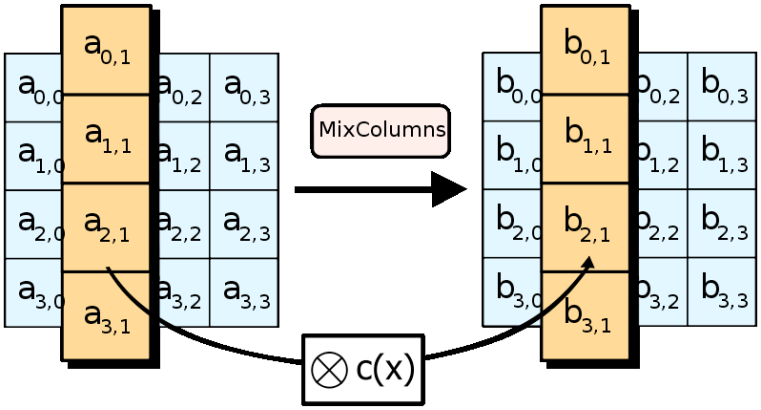
The `ShiftRows` step operates on the rows of the state; it cyclically shifts the bytes in each row by a certain offset. For AES, the first row is left unchanged. Each byte of the second row is shifted one to the left. Similarly, the third and fourth rows are shifted by offsets of two and three respectively. For the block of size 128 bits and 192 bits the shifting pattern is the same. 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). In the case of the 256-bit block, the first row is unchanged and the shifting for 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. Here Aij is from cipher text and Bij is from key.



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

During this operation, each column is multiplied by the known matrix that for the 128 bit key is

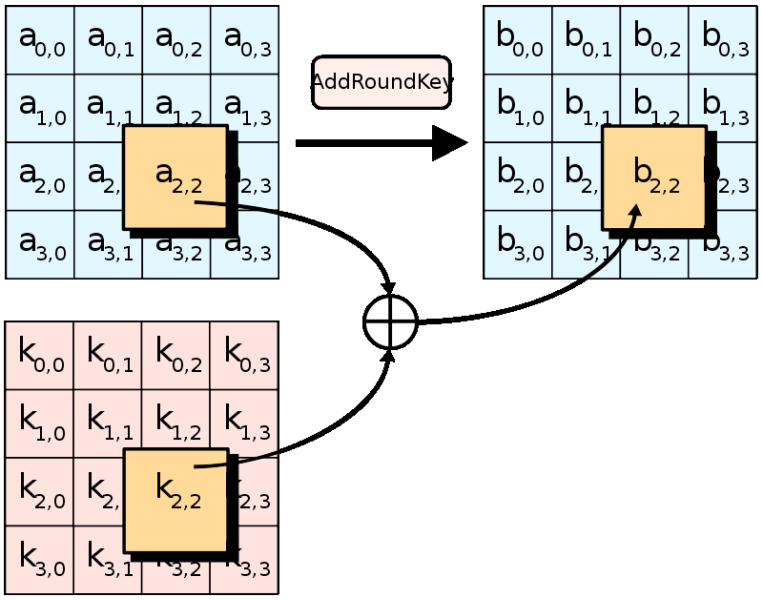
$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}.$$

The multiplication operation is defined as: multiplication by 1 means leaving unchanged, multiplication by 2 means shifting byte to the left and multiplication by 3 means shifting to the left and then performing xor with the initial unshifted value. After shifting, a conditional xor with 0x11B should be performed if the shifted value is larger than 0xFF.

In more general sense, each column is treated as a polynomial over **GF**(2^8) and is then multiplied modulo x^4+1 with a fixed polynomial $c(x) = 0x03 \cdot x^3 + x^2 + x + 0x02$. The coefficients are displayed in their hexadecimal equivalent of the binary representation of bit polynomials from **GF**(2)[x]. The `MixColumns` step can also be viewed as a multiplication by a particular MDS matrix in a finite field. This process is described further in the article Rijndael mix columns. Using the polynomial one matrix was created. Using that matrix add with o/p came from previous state.

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.



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 `SubBytes` and `ShiftRows` with `MixColumns`, and transforming them into a sequence of table lookups. This requires four 256-entry 32-bit tables, which utilizes a total of four kilobytes (4096 bytes) of memory—one kilobyte for each table. A round can now 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.[7]

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.[8]

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 may 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."[9]

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.[10]

Known attacks

For cryptographers, a cryptographic "break" is anything faster than a brute force attack - trying every possible key. Thus, an attack against a 256-bit-key AES requiring 2^{200} operations (compared to 2^{256} possible keys) would be considered a break, even though 2^{200} operations would still take far longer than the age of the universe to complete. The largest successful publicly-known brute force attack has been against a 64-bit RC5 key by distributed.net.[11]

AES has a fairly simple algebraic description.[12] 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.[13] Since then, other papers have shown that the attack as originally presented is unworkable; see XSL attack on block ciphers.

During the AES process, developers of competing algorithms wrote of Rijndael, "...we are concerned about [its] use...in security-critical applications."[14] However, at the end of the AES process, Bruce Schneier, a developer of the competing algorithm Twofish, wrote that while he thought successful academic attacks on Rijndael would be developed someday, "I do not believe that anyone will ever discover an attack that will allow someone to read Rijndael traffic."[15]

On July 1, 2009, Bruce Schneier blogged[16] about a related-key attack on the 192-bit and 256-bit versions of AES, discovered by Alex Biryukov and Dmitry Khovratovich,[17] which exploits AES's somewhat simple key schedule and has a complexity of 2^{119}. Until 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.[18] Another attack was blogged by Bruce Schneier[19] on July 30, 2009 and released as a preprint[20] 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.

In November 2009, the first known-key distinguishing attack against a reduced 8-round version of AES-128 was released as a preprint.[21] This known-key distinguishing attack is an improvement of the rebound or the start-from-the-middle attacks for 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 computation complexity of 2^{48}, and a memory complexity of 2^{32}.

In July 2010 Vincent Rijmen published an ironic paper on "chosen-key-relations-in-the-middle" attacks on AES-128.[22]

Side-channel attacks

Side-channel attacks do not attack the underlying cipher and so have nothing to do with its security as described here, but attack implementations of the cipher on 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.[23] 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), and the attack required over 200 million chosen plaintexts.[24]

In October 2005, Dag Arne Osvik, Adi Shamir and Eran Tromer presented a paper demonstrating several cache-timing attacks against AES.[25] 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 key with complexity of 2^{32}.[26]

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.[27] Like some earlier attacks this one requires the ability to run arbitrary code on the system performing the AES encryption. [28]

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 validated cryptographic modules is required by the United States Government for all unclassified uses of cryptography. 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)[29] 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.

FIPS 140-2 validation is challenging to achieve both technically and fiscally. 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) 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 (in ZIP format) [30].

Performance

Good performance (high speed and low RAM requirements) were an explicit goal of the AES selection process. Thus AES performs well on a wide variety of hardware, from 8-bit smartcards to high-performance computers.

On a Pentium Pro, AES encryption requires 18 clock cycles / byte[31] , equivalent to a throughput of about 11 MiB/s for a 200MHz processor. On a Pentium M 1.7GHz throughput is about 60 MiB/s.

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] Westlund, Harold B. (2002). "NIST reports measurable success of Advanced Encryption Standard" (http://www.findarticles.com/p/articles/mi_m0IKZ/is_3_107?pnum=2&opg=90984479). *Journal of Research of the National Institute of Standards and Technology*. .

[4] John Schwartz (October 3, 2000). "U.S. Selects a New Encryption Technique" (http://www.nytimes.com/2000/10/03/business/technology-us-selects-a-new-encryption-technique.html). *New York Times*. .

[5] "'Rijndael' pronunciation" (http://rijndael.info/audio/rijndael_pronunciation.wav). .

[6] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, Niels Ferguson, Tadayoshi Kohno, Mike Stay (May 2000). "The Twofish Team's Final Comments on AES Selection" (http://www.schneier.com/paper-twofish-final.pdf). .

[7] "Efficient software implementation of AES on 32-bit platforms". (http://www.springerlink.com/index/UVX5NQGNN55VK199.pdf) Lecture Notes in Computer Science: 2523. 2003

[8] http://code.google.com/p/byte-oriented-aes

[9] 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). . Retrieved 2011-02-15.

[10] John Kelsey, Stefan Lucks, Bruce Schneier, Mike Stay, David Wagner, and Doug Whiting, *Improved Cryptanalysis of Rijndael*, Fast Software Encryption, 2000 pp213–230 (http://www.schneier.com/paper-rijndael.html)

[11] Ou, George (April 30, 2006). "Is encryption really crackable?" (http://www.webcitation.org/5rocpRxhN). Ziff-Davis. Archived from the original (http://www.zdnet.com/blog/ou/is-encryption-really-crackable/204) on August 7, 2010. . Retrieved August 7, 2010.

[12] "Sean Murphy" (http://www.isg.rhul.ac.uk/~sean/). University of London. . Retrieved 2008-11-02.

[13] Bruce Schneier. "AES News, Crypto-Gram Newsletter, September 15, 2002" (http://www.schneier.com/crypto-gram-0209.html). . Retrieved 2007-07-27.

[14] Niels Ferguson, Richard Schroeppel, Doug Whiting (2001). "A simple algebraic representation of Rijndael" (http://www.macfergus.com/pub/rdalgeq.html) (PDF/PostScript). *Proceedings of Selected Areas in Cryptography, 2001, Lecture Notes in Computer Science*. Springer-Verlag. pp. 103–111. . Retrieved 2006-10-06.

[15] Bruce Schneier, AES Announced (http://www.schneier.com/crypto-gram-0010.html), October 15, 2000

[16] Bruce Schneier (2009-07-01). "New Attack on AES" (http://www.schneier.com/blog/archives/2009/07/new_attack_on_a.html). *Schneier on Security, A blog covering security and security technology*. . Retrieved 2010-03-11.

[17] Biryukov, Alex; Khovratovich, Dmitry (2009-12-04). "Related-key Cryptanalysis of the Full AES-192 and AES-256" (http://eprint.iacr.org/2009/317). . Retrieved 2010-03-11.

[18] 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.

[19] 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*. . Retrieved 2010-03-11.

[20] 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). . Retrieved 2010-03-11.

[21] Henri Gilbert; Thomas Peyrin (2009-11-09). "Super-Sbox Cryptanalysis: Improved Attacks for AES-like permutations" (http://eprint.iacr.org/2009/531). . Retrieved 2010-03-11.

[22] Vincent Rijmen (2010). "Practical-Titled Attack on AES-128 Using Chosen-Text Relations" (http://eprint.iacr.org/2010/337.pdf). .

[23] "Index of formal scientific papers" (http://cr.yp.to/papers.html#cachetiming). Cr.yp.to. . Retrieved 2008-11-02.

[24] Bruce Schneier. "AES Timing Attack" (http://www.schneier.com/blog/archives/2005/05/aes_timing_atta_1.html). . Retrieved 2007-03-17.

[25] Dag Arne Osvik1; Adi Shamir2 and Eran Tromer2 (2005-11-20) (PDF). *Cache Attacks and Countermeasures: the Case of AES* (http://www.wisdom.weizmann.ac.il/~tromer/papers/cache.pdf). . Retrieved 2008-11-02.

[26] Dhiman Saha, Debdeep Mukhopadhyay, Dipanwita RoyChowdhury (PDF). *A Diagonal Fault Attack on the Advanced Encryption Standard* (http://eprint.iacr.org/2009/581.pdf). . Retrieved 2009-12-08.

[27] Endre Bangerter, David Gullasch and Stephan Krenn (2010). "Cache Games – Bringing Access-Based Cache Attacks on AES to Practice" (http://eprint.iacr.org/2010/594.pdf). .

[28] http://news.ycombinator.com/item?id=1937902

[29] http://csrc.nist.gov/groups/STM/cavp/index.html

[30] http://csrc.nist.gov/groups/STM/cavp/documents/aes/KAT_AES.zip

[31] "Performance Comparisons of the AES submissions" (http://www.schneier.com/paper-aes-performance.pdf) (PDF). 1999-02-01. . Retrieved 2010-12-28.

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" (http://wiki.crypto.rub.de/Buch/sample_chapters.php), Chapter 4 of "Understanding Cryptography, A Textbook for Students and Practitioners". Springer, 2009.

External links

- Original AES-Process submission and derived code (http://members.fortunecity.it/blackvisionit/CRYPT_SOURCE.HTM)
- FIPS PUB 197: the official AES standard (http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf) (PDF file)
- AES algorithm archive information - (old, unmaintained) (http://csrc.nist.gov/archive/aes/rijndael/wsdindex.html)
- Animation of the AES encryption process (http://www.formaestudio.com/rijndaelinspector/)
- Fully Functional Animation of the AES encryption process and key expansion (128 bits) - based on the work of Enrique Zabala (previous link) (http://blog.ultrassecreto.com/wp-content/uploads/2009/06/projetofinal.html)
- Stick Figure Guide to AES (http://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html), a layman introduction to cryptography and AES.
- An in-depth description of the Advanced Encryption Standard and the maths behind it. C implementation provided. (http://www.x-n2o.com/aes-explained/)
- Accelerating AES in software by using custom instructions (http://www.ensilica.com/pdfs/A_study_of_aes_and_its_efficient_implementation_on_eSi_RISC_r1.0.pdf) (PDF file)
- AES VHDL implementation (pipelined and iterative) (http://www.isaakian.com/VHDL_page.html)

Article Sources and Contributors

Advanced Encryption Standard *Source*: http://en.wikipedia.org/w/index.php?oldid=417809828 *Contributors*: *drew, ++Martin++, 16@r, 216.150.138.xxx, 403forbidden, Adoniscik, Afog, Ahoerstemeier, Aither, Ajithabraham.m, AlephGamma, Algocu, Ali@gwc.org.uk, AlistairMcMillan, Amoss, Andrei Stroe, Andy Dingley, AniLoveBe, Ansell, Antimatter15, Apapadop, Apokrif, Arichnad, ArnoldReinhold, Arrowoftime, Arturj, Ashley Y, Ashmoo, Astronaut, Audriusa, Austin Hair, Avocade, Basawala, Batty, BenjaminGittins, Bile43113, Binba, Blackvisionit, BrokenSegue, Bryan Derksen, Bsdaemon, CDV, CSWarren, Caligatio, Centrx, ChlkDstTtr, Chrisahn, Chrylis, Ciphergoth, ClementSeveillac, Climbon, Collard, Conseguenza, Conversion script, Cralar, DMS, Daggilli, DanBealeCocks, Daniel.Cardenas, Danny, Darrien, DataWraith, Davidgothberg, Dawnseeker2000, Dchristle, Dcoetzee, Denisutku, DerEikopf, Dermeister, Df1, Dimawik, Dirkx, Dispenser, Dmitriid, Doctorhook, Donarreiskoffer, DopefishJustin, Dougher, DrHok, Drj, EamonnAG, Ed g2s, Eike, ElBenevolente, ElectroKitty, Endareth, Ennan, Ericbg05, Farnik, Fintler, Flamurai, Flib0, Frencheigh, Frysalebald, Fudoreaper, Fxbx, Gaius Cornelius, Galoubet, Gavia immer, George A. M., Gerbrant, Giftlite, Gndurant, Gogo Dodo, GoldKanga, Grauw, Greg Tyler, GregorB, Gudguy1, Guymacon, Haakon, Hadal, Hannes Röst, Harishmalgae, Haseo9999, Hashar, Hashproduct, Hede2000, Hellisp, Hermes17, Hoho, Hugh Emberson, Hydrogen Iodide, IByte, Ianneub, Imran, Inkling, Intgr, Intractable, Itusg15q4user, James mcl, James.nvc, Jaredwf, Jarhed, Jasper Chua, Jay, JayTau, Jeffz1, Jesse Viviano, Jestep, JidGom, Jmnbatista, Johnteslade, Jonathan.lampe@standardnetworks.com, Josh Liu, Julesd, KTC, Kenyon, Kravietz, Kwamikagami, L33th4x0rguy, Lee Carre, Legios, Lightmouse, LinkTiger, Loadmaster, LodeRunner, LouScheffer, Luckyherb, Lukejamesoconnor, M@, Malbrain, Manuel Anastácio, Marcel Augustus, Marcushan, Marygillwiki, Mathiastck, Matt Crypto, Matusz, Maxgrin, Maximus Rex, Melchoir, MentalForager, Meridian, Mewtu, MiG, Michael Shields, Michaelwagnercanberra, Mike Rosoft, Mike Schwartz, Minna Sora no Shita, Miserlou, Mmernex, Mmtux, Moonradar, Mordemur, Msoos, NTF, Nabokov, Nat.latos, Netsnipe, Niyazlife, Nneonneo, Norm mit, Ntsimp, Numbo3, Nuwewsco, Ojigiri, Oleg1899, Pakaran, Paranoid123, Paulb73, Paulehoffman, PizzaMan, Pne, Poor Yorick, Populus, Posix memalign, Powerlife, Prosfilaes, Pttam, Ptyantai, Puchiko, Quadell, Quantumor, Qutezuce, RP88, RainbowOfLight, Rajeshksv37, RealWorldExperience, Rich Farmbrough, Richard506, Rjwilmsi, Roadrunner, Rob*, Robert Brockway, RobertG, Rodzilla, Romualdo Juan Caruso, Ross Burgess, RoySmith, Rprpr, SColombo, SKJDh, SLi, Sam Hocevar, Samboy, Scovetta, Seb az86556, Sebastian Goll, Sfdev, Shadowborg, Sharcho, Shaul1, ShaunMacPherson, Sietse Snel, Simetrical, Singpolyma, SiobhanHansa, Sleske, Sourcejedi, Starwiz, Steve Checkoway, Stevenj, Stevertigo, Stoive, Stolsvik, Superm401, Svick, Tassedethe, Template namespace initialisation script, TenPoundHammer, Teorth, TestPilot, The Anome, The Thing That Should Not Be, Thelb4, Themfromspace, Thomas Springer, Thorwald, Threeafterthree, Tibti, Timeineurope, Timwi, Tomcully, Tomstdenis, TonyW, Torzsmokus, Vicarious, Volkan YAZICI, VvV, Whkoh, Wikiborg, Wikisux, Wingedsubmariner, Wj32, Wk muriithi, Wperdue, Ww, X-N2O, XFireRaidX, Xizhi.zhu, Zvn, শরি, 531 anonymous edits

Image Sources, Licenses and Contributors

Image:AES-SubBytes.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:AES-SubBytes.svg *License*: Public Domain *Contributors*: User:Matt Crypto

Image:AES-ShiftRows.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:AES-ShiftRows.svg *License*: Public Domain *Contributors*: User:Matt Crypto

Image:AES-MixColumns.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:AES-MixColumns.svg *License*: Public Domain *Contributors*: User:Matt Crypto

Image:AES-AddRoundKey.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:AES-AddRoundKey.svg *License*: Public Domain *Contributors*: User:Matt Crypto

License