Global Information Assurance Certification Paper

The Advanced Encryption Standard: A Review of the Finalists
Kathleen Flood
October 9, 2000

On the second of October 2000 the National Institute of Standards and Technology (NIST) of the U.S. Department of Commerce announced the selection for the new encryption standard, the Advanced Encryption Standard, as the replacement for the Data Encryption Standard (DES). For more than two decades, DES has been the cryptographic algorithm used to protect the "sensitive" U.S. government communications. Although the Triple DES algorithm provides a stronger encryption than DES, it does so with a reduction in performance and an increased cost of implementation. NIST identified the need to select a new "faster, cheaper, better" standard that would provide greater security well into the twenty-first century. This paper will briefly review the history of the AES, the five finalists' algorithms, and the NIST's evaluation and selection of the new proposed standard.

Background

NIST announced its intention to seek a replacement for DES in January 1997, and in September of the same year it made the formal call for algorithms. As noted in the call for algorithms, the submissions had to be symmetric key block cipher algorithms supporting key sizes of 128, 192, and 256 bits and block sizes of 128 bits in order to be considered for further evaluation. Once having met those requirements, the algorithms would be evaluated based on security, licensing requirements, computational efficiency, memory requirements, flexibility, hardware and software suitability, and simplicity. It was hoped that the collaborative effort of academia, industry, cryptography community, and the U.S. government would provide the amount of international exposure, scrutiny, and cryptanalysis necessary to ensure the strength of the selection and to uncover possible weaknesses. The first round of evaluation and analysis of the 15 initial proposals began in August 1998. The five finalists were announced by NIST in August 1999.

The Finalists

The NIST noted that it was difficult to measure and compare the security of the algorithms but an attempt was made to do so using simplified variants of the actual algorithms with unit of measurement being a security margin. According to the Technical Details of the Round 2 Analysis as presented in NIST's Report "In general, the results will be biased against algorithms that attract greater scrutiny in a limited analysis period. This could plausibly occur, for example, if a particular algorithm is simpler, or at least appears to be simpler, to analyze against certain attacks. Another factor could be the ancestry of the algorithm and its constituent techniques, and the existence of previous attacks upon which to build. The proposed measure would tend to favor novel techniques for resisting attacks, techniques that have not yet stood the test of time." [NIST] Even though NIST found that all five finalists had "adequate security for the AES", three algorithms, MARS, Serpent, and Twofish, were noted as having a high security margin.

MARS

This algorithm was submitted by a team from IBM, the developers of Lucifer, which with input from the National Security Agency became the original DES. As noted in IBM's submission paper, "It is designed to take advantage of the powerful operations supported in today's computers, resulting in a much improved security/performance tradeoff over existing ciphers. As a result, MARS offers better security than triple DES while running significantly faster than single DES. The key for MARS may be from 4 to 39 words in length." [IBM] The algorithm used a pre-whitening key addition, rounds (iterations) of keyed transformations, rounds of unkeyed mixing which used S-boxes (substitution tables), addition, and an XOR operation, and finally a post-whitening key subtraction.

In the NIST's assessment of MARS, it found that in the software implementation on the various platforms (8-bit, 32-bit, 64-bit processors and digital signal processors), the algorithm achieved average speed performance, in all three key sizes, for encryption and decryption functions and for key setup. In restricted-space environments, that is, environments with limited ROM and RAM, MARS had greater resource requirements, particularly for ROM, which could present a problem for an application such as a smart card. In terms of the performance in a hardware implementation, MARS was assessed to have an above average area requirement and a below average throughput, yielding a below average efficiency. Efficiency was computed as [Throughput / Area]. Since MARS' encryption and decryption functions were similar, there was no significant difference between those functions in their speed and space used. MARS' subkey computation and storage created additional resource demands, an issue in memory-restricted environments. To its credit, MARS has the flexibility to support a wide range of key sizes, from 128 to 448 bits.

RC6

The RC6 algorithm was submitted by the RSA Laboratories. In the abstract of its submission paper, RSA stated that "RC6 is an evolutionary improvement of RC5, designed to meet the requirements of the Advanced Encryption Standard (AES). [It] makes essential use of data-dependent rotations, […] four working registers instead of two, and the inclusion of integer multiplication as an additional primitive operation. The use of multiplication greatly increases the diffusion achieved per round, allowing for greater security, fewer rounds, and increased throughput." [RSA]

The NIST found that RC6's performance to be above average in encryption and decryption speed for 128-bit keys (notably well in 32-bit platforms), average for key setup speed, and consistent for all three key sizes. RC6 had a low ROM requirement but a high RAM requirement since it did not have an on-the-fly subkey computation for the decryption process. Its throughput in a hardware implementation was found to be average, and not dependent upon the key size. As with MARS, the encryption and decryption processes for RC6 were similar enough to require no significant additional area and speed for implementation. As noted above, on-the-fly subkey computation for decryption was not supported, but was supported for encryption. The parameter characteristics of the algorithm support key sizes larger than 256 bits.

Serpent

This algorithm was designed by Ross Anderson, Eli Biham, and Lars Knudsen. In explaining their design philosophy, the international Serpent team noted, "Its design is highly conservative, yet still allows a very efficient implementation. It uses S-boxes similar to those of DES in a new structure that simultaneously allows a more rapid avalanche, a more efficient bitslice implementation, and an easy analysis that enables us to demonstrate its security against all known types of attack. …We did not feel it appropriate to use novel and untested ideas in a cipher which, if accepted after a short review period, will be used to protect enormous volumes of financial transactions, health records and government information over a period of decades." [ABK] The algorithm used an initial permutation, followed by 32 rounds each consisting of a keyed mixing, an S-box pass, and a linear transformation, followed by a final permutation.

In the NIST's software implementation tests, the Serpent algorithm had the slowest speed for encryption and decryption for 128-bit keys, with a consistent performance for all three key sizes. However, Serpent had an average key setup speed. Its implementation had low ROM and RAM requirements, making it favorable for restricted-space environments. Despite its slower performance in software implementation, Serpent was found to have the highest throughput in hardware in non-feedback mode, and second best in feedback mode. In addition, its efficiency was determined to be very good. Although the algorithm's speed did not vary much between the encryption and decryption process, the two processes were quite different, sharing very few hardware resources. Serpent supported on-the-fly subkey computation for encryption and decryption and supported key sizes up to 256 bits.

Twofish

A team from Counterpane Systems, Hi/fn, and University of California Berkeley submitted the Twofish algorithm. Echoing the Serpent team's view of conservative design, the Twofish team explained, "We used well-studied design elements throughout the algorithm. We started with a Feistel networks, probably the most studied block cipher structure, instead of something newer. …Complicated round functions are harder to analyze and rely on more ad-hoc arguments for security. …In Twofish, we tried to create a simple round function and then iterate it more than enough times for security." [SKW] The algorithm used a Feistel structure composed of key-dependent S-boxes, a fixed maximum distance matrix, a pseudo-Hadamard transform, and bitwise rotations.

In software implementation testing, NIST noted that Twofish had generally average speed for encryption and decryption for 128-bit keys, although the results were mixed across platforms. Its key setup speed was slower, with performance degrading with increased key sizes. Due to its low ROM and RAM requirements, the algorithm was found to be suitable for restricted-space environments. In hardware implementation testing, Twofish had average throughput and efficiency. Since the encryption and decryption functions were very similar, its efficiency was not greatly effected. The algorithm demonstrated key agility by supporting on-the-fly subkey computation for both encryption and decryption. It also supported variable key sizes up to 256 bits.

And the winner is…

The Rijndael algorithm was designed by Joan Daemen and Vincent Rijmen. The Dutch team noted in their discussion of the design rationale, three criteria: "Resistance against all known attacks; Speed and code compactness on a wide range of platforms; Design simplicity." [DR] They also gave credit to another block cipher, Square, as an influence for Rijndael's design. In the algorithm each regular round involved four steps or layers. The first layer was a byte-

oriented S-box pass. The next layer involved shifting rows of the array, followed by a layer in which the columns were mixed using a matrix multiplication. In the final layer the subkey addition was applied to the array. The extra final round omitted the Mix Column step, but was otherwise the same as a regular round.

The following pseudo C notation describing a round was provided in the submission paper:

```
Round(State,RoundKey)

{

ByteSub(State);

ShiftRow(State);

MixColumn(State);

AddRoundKey(State,RoundKey);

}
```

The pseudo notation for the Rijndael cipher:

```
Rijndael(State,CipherKey)

{

KeyExpansion(CipherKey,ExpandedKey) ;

AddRoundKey(State,ExpandedKey);

For( i=1 ; i<Nr ; i++ ) Round(State,ExpandedKey + Nb*i) ;

FinalRound(State,ExpandedKey + Nb*Nr);
```

where Nr = number of rounds, Nb = block length / 32 [DR] The number of rounds vary according to the size of the key and the block.

In NIST's evaluation Rijndael was found to have above average encryption and decryption speed for 128-bit keys, with an even performance across platforms; however the performance was diminished when larger key sizes were used. Its key setup was the fastest among the algorithms. Rijndael was found also to be very suitable for restricted-space environments since the ROM and RAM requirements were quite low. In the hardware implementation tests its efficiency was found to be very good and its throughput was the best in feedback mode and second best in non-feedback mode. Unlike some of the other algorithms, Rijndael's encryption and decryption functions were not similar, with a significant increase in space requirements when both functions were implemented. The key setup was also slower for decryption than it was for encryption. Rijndael did not have the key agility of the Twofish algorithm, supporting on-the-fly subkey computation for encryption only. The algorithm supported the required key sizes but with its flexibility could support various block and key sizes as well as the number of rounds. A good consistent performance across the evaluation criteria influenced the NIST's final selection of this algorithm as the AES.

What's Next

A draft Federal Information Processing Standard (FIPS) for the AES will soon be published, allowing for a three-month period of public review and comment. If after that period any revisions are necessary, NIST will make them and then propose the standard to the Secretary of Commerce for adoption as an official FIPS. The present expected date is Summer 2001. Although DES will be phased out of use, Triple DES for now will continue to be an approved algorithm for U.S. Government use.

References

[ABK] Anderson, R., Biham, E., Knudsen, L. "Serpent: A Proposal for the Advanced Encryption Standard". June 1998. URL: http://www.cl.cam.ac.uk/~rja14/serpent.html (3 October 2000).

[DR] Daemen, J.,Rijmen, V. "AES Proposal: Rijndael", version 2. 9 March 1999. URL: http://www.esat.kuleuven.ac.be/~rijmen/rijndael/ (3 October 2000).

[IBM] Burwick, C., Coppersmith, D., et al. "MARS – a candidate cipher for AES" 20 August 1999. URL: http://www.research.ibm.com/security/mars.html (3 October 2000).

[NIST] National Institute of Standards and Technology, U.S. Department of Commerce. "Report on the Development of the Advanced Encryption Standard (AES)." Advanced Encryption Standard (AES) Development Effort. 2 October 2000. URL: http://csrc.nist.gov/encryption/aes/ (3 October 2000).

[RSA] Rivest, R., Robshaw, M., et al. "The RC6[tm] Block Cipher". June 1998. URL: http://www.rsasecurity.com/rsalabs/aes/index.html (3 October 2000).

[SKW] Schneier, B., Kelsey, J. , et al. "Twofish: A 128-Bit Block Cipher". 15 June 1998. URL: http://www.counterpane.com/twofish-paper.html (3 October 2000).

National Institute of Standards and Technology, U.S. Department of Commerce. "Announcing Request for Candidate Algorithm Nominations for the Advanced Encryption Standard (AES)." Federal Register, Volume 62, Number 177. 12 September 1997. URL: http://csrc.nist.gov/encryption/aes/pre-round1/aes_9709.htm (7 October 2000)

National Institute of Standards and Technology, U.S. Department of Commerce. "Overview of the AES Development Effort.")." Advanced Encryption Standard (AES) Development Effort. August 1999. URL: http://csrc.nist.gov/encryption/aes/ (3 October 2000)

National Security Agency. "Hardware Performance Simulations of Round 2 Advanced Encryption Standard Algorithms." Round 2 Analysis. 15 May 2000. URL: http://csrc.nist.gov/encryption/aes/round2/r2anlsys.htm (3 October 2000).

Seifried, K. "Advanced Encryption Standard Released", 3 October 2000. URL: http://securityportal.com/articles/aes20001003.html (3 October 2000).

Savard, J. "Towards the 128-bit Era: AES Candidates." A Cryptographic Compendium. URL: http://home.ecn.ab.ca/~jsavard/crypto.htm (4 October 2000).

Last Updated: May 27th, 2017

Upcoming Training

SANS Stockholm 2017	Stockholm, Sweden	May 29, 2017 - Jun 03, 2017	Live Event
SANS San Francisco Summer 2017	San Francisco, CA	Jun 05, 2017 - Jun 10, 2017	Live Event
Security Operations Center Summit & Training	Washington, DC	Jun 05, 2017 - Jun 12, 2017	Live Event
SANS Houston 2017	Houston, TX	Jun 05, 2017 - Jun 10, 2017	Live Event
Community SANS Ottawa SEC401	Ottawa, ON	Jun 05, 2017 - Jun 10, 2017	Community SANS
SANS Rocky Mountain 2017 - SEC401: Security Essentials Bootcamp Style	Denver, CO	Jun 12, 2017 - Jun 17, 2017	vLive
Community SANS Portland SEC401	Portland, OR	Jun 12, 2017 - Jun 17, 2017	Community SANS
SANS Secure Europe 2017	Amsterdam, Netherlands	Jun 12, 2017 - Jun 20, 2017	Live Event
SANS Rocky Mountain 2017	Denver, CO	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Charlotte 2017	Charlotte, NC	Jun 12, 2017 - Jun 17, 2017	Live Event
SANS Minneapolis 2017	Minneapolis, MN	Jun 19, 2017 - Jun 24, 2017	Live Event
SANS Columbia, MD 2017	Columbia, MD	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS Cyber Defence Canberra 2017	Canberra, Australia	Jun 26, 2017 - Jul 08, 2017	Live Event
SANS Paris 2017	Paris, France	Jun 26, 2017 - Jul 01, 2017	Live Event
SANS London July 2017	London, United Kingdom	Jul 03, 2017 - Jul 08, 2017	Live Event
Cyber Defence Japan 2017	Tokyo, Japan	Jul 05, 2017 - Jul 15, 2017	Live Event
SANS Cyber Defence Singapore 2017	Singapore, Singapore	Jul 10, 2017 - Jul 15, 2017	Live Event
Community SANS Minneapolis SEC401	Minneapolis, MN	Jul 10, 2017 - Jul 15, 2017	Community SANS
SANS Los Angeles - Long Beach 2017	Long Beach, CA	Jul 10, 2017 - Jul 15, 2017	Live Event
Community SANS Phoenix SEC401	Phoenix, AZ	Jul 10, 2017 - Jul 15, 2017	Community SANS
SANS Munich Summer 2017	Munich, Germany	Jul 10, 2017 - Jul 15, 2017	Live Event
Mentor Session - SEC401	Macon, GA	Jul 12, 2017 - Aug 23, 2017	Mentor
Mentor Session - SEC401	Ventura, CA	Jul 12, 2017 - Sep 13, 2017	Mentor
Community SANS Atlanta SEC401	Atlanta, GA	Jul 17, 2017 - Jul 22, 2017	Community SANS
Community SANS Colorado Springs SEC401	Colorado Springs, CO	Jul 17, 2017 - Jul 22, 2017	Community SANS
SANSFIRE 2017	Washington, DC	Jul 22, 2017 - Jul 29, 2017	Live Event
SANSFIRE 2017 - SEC401: Security Essentials Bootcamp Style	Washington, DC	Jul 24, 2017 - Jul 29, 2017	vLive
Community SANS Charleston SEC401	Charleston, SC	Jul 24, 2017 - Jul 29, 2017	Community SANS
Community SANS Fort Lauderdale SEC401	Fort Lauderdale, FL	Jul 31, 2017 - Aug 05, 2017	Community SANS
SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Prague 2017	Prague, Czech Republic	Aug 07, 2017 - Aug 12, 2017	Live Event