International Journal of Information Technology and Knowledge Management
January-June 2011, Volume 4, No. 1, pp. 113-117

AES: A SYMMETRIC KEY CRYPTOGRAPHIC SYSTEM

Punita Meelu[1] & Sitender Malik[2]

With the introduction of the computer, the need for automated tools for protecting files and other information stored on the computer became evident. Security is always a major concern in the field of communication. Advanced Encryption Standard (AES) algorithm is a popular Symmetric Key Cryptographic Scheme that guarantee confidentiality and authenticity over an insecure communication channel. This paper presents the fundamental mathematics behind the AES algorithm along with a brief description of some cryptographic primitives that are commonly used in the field of communication security. It also includes several computational issue, optimization of cipher as well as the analysis of AES security aspects against different kinds of attacks including the countermeasures against these attacks.

Keywords: Cryptography, AES, Encryption, Decryption

1. Introduction

Several techniques, such as cryptography, steganography, watermarking, and scrambling, have been developed to keep data secure, private, and copyright protected. However, the need for secure transactions in ecommerce, private networks, and secure messaging has moved encryption into the commercial realm. The fundamental necessity in security is to hide information from irrelevant public or malicious attackers. This requirement has given birth to different kinds of cryptographic primitives including symmetric and asymmetric cryptography, hash functions, digital signatures, message authentication codes etc. In symmetric encryption, a key is shared between the sender and the receiver which is kept secret from the intruder[9]. Advanced encryption standard (AES) was issued as Federal Information Processing Standards (FIPS) by National Institute of Standards and Technology (NIST) as a successor to data encryption standard (DES) algorithms. Among the different kinds of symmetric algorithms, Advanced Encryption Standard (AES) is gaining popularity due to its better security and efficiency than its predecessors [10]. It was defined in Federal Information Processing Standard (FIPS) 192, published in November 2001 [7, 5]. Advanced Encryption Standard is a symmetric 128-bit, 192-bit or 256-bit block data encryption technique developed by Belgian cryptographers Joan Daemen and Vincent Rijmen[10]. It is also known an Rijndael, combining the author's names. The U.S government adopted the algorithm as its official encryption technique in 2000-10, replacing DES encryption[1]. Both the key size and the block size may be chosen to be any of 128, 192, or 256 bits in any combination. The remainder of this paper is organized as follows. In Section 2 presents the overview of AES cipher which is based on the design principle of Substitution permutation network as well as AES algorithm including encryption and decryption process. Section 3 describes the computational issues of AES with example. Section 4 we describes the Optimization of Cipher. Section 5 analyze security of AES aspects against different kinds of attacks. Finally, Section 6 gives conclusion remarks and future work.

2. Overview of AES

AES is based on a design principle known as a Substitution permutation network. It is fast in both software and hardware. 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[7]. A number of AES parameters depend on the key length shown below in Table 1.1. The blocksize has a maximum of 256 bits, but the keysize has theoretically no maximum.AES operates on a 4×4 array of bytes, termed the state (versions of Rijndael with a larger block size have additional columns in the state)[3]. 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. Number of rounds to be used depend on the length of key e.g. 10 round for 128 bit key, 12 rounds for 192-bit key and 14 rounds for 256 bit keys. Both in encryption and decryption process, the State array is modified at each round by a round function that defines four different byte-oriented transformations [1] SubBytes, Shiftrows, MixColumns and AddRoundKey.

[1,2]Department of CSE, N.C.College of Engg., Israna, Panipat
Email: [1]punita.meelu@gmail.com, [2]sitender007@gmail.com

Table 1.1
AES Parameters

Key size (words/ bytes/bits)	4/16/ 128	6/24/ 192	8/32/256
Plaintext block size (words/bytes/bits)	4/16/ 128	4/16/ 128	4/16/128
Number of rounds	10	12	14
Round key size (words/bytes/bits)	4/16/ 128	4/16/ 128	4/16/128
Expanded key size (words/bytes)	44/176	52/208	60/240

Encryption and Decryption Process

The encryption and decryption process composed of several rounds depending on the size of the cipher key where each round performs some specific functions. In this paper, we have considered encryption and decryption process for 128 bit cipher key that requires 10 different rounds to complete the process. The different steps involved in the algorithm are as follows and shown in Figure.1

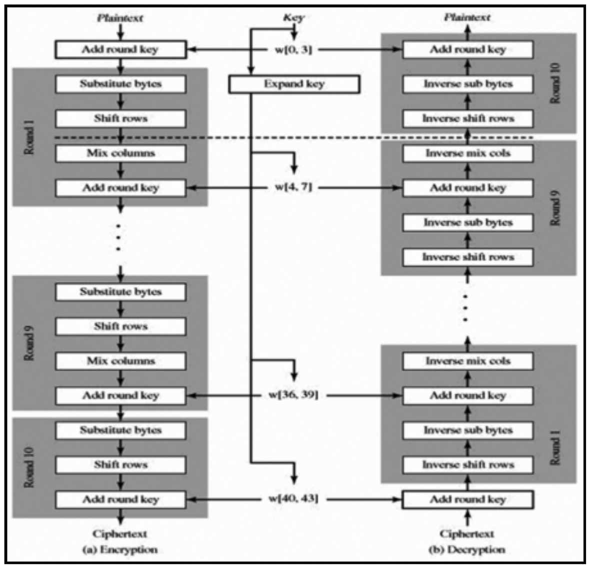

Fig. 1: AES Encryption and Decryption

Figure 1 shows the overall structure of AES. The input to the encryption and decryption algorithms is a single 128-bit block. This block is depicted as a square matrix of bytes. This block is copied into the State array, which is modified at each stage of encryption or decryption. After the final stage, State is copied to an output matrix. These operations are depicted in Figure 1(a). Similarly, the 128-bit key is depicted as a square matrix of bytes. This key is then expanded into an array of key schedule words; each word is four bytes and the total key schedule is 44 words for the 128-bit key Figure 1(b). The ordering of bytes within a matrix is by column. So, for example, the first four bytes of a 128-bit plaintext input to the encryption cipher occupy the first column of the in matrix, the second four bytes occupy the second column, and so on. Similarly, the first four bytes of the expanded key, which form a word, occupy the first column of the w matrix.

Both encryption and decryption process begins with the initial round i.e. round0 performs only the AddRound- Key transformation on the state arrray and provides the security as this is the only stage that makes use of the secret key. Initial round is followed by nine identical rounds where each round incorporates SubBytes, Shiftrows, MixColumns and AddRoundKey transformations respectively on the state array. The final round is slightly different from the other rounds as it uses only three functions other than MixColumns transformation.

AES Algorithm

The AES algorithm is a block cipher that encrypts blocks of 128, 192, or 256 bits using symmetric keys of 128, 192 or 256 bits. The standard requires that the AES must implement a symmetric block cipher with a block size of 128 bits. Electronic Codebook Mode (ECB) has been used in the design.

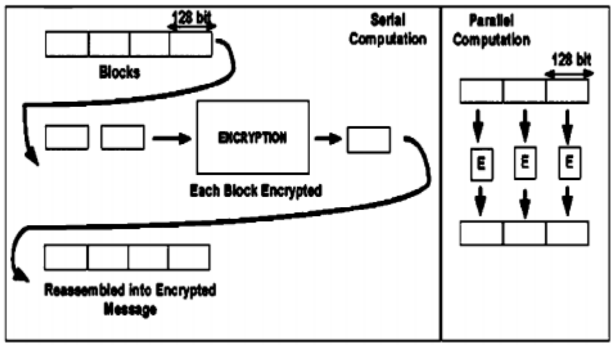

Fig. 2: ECB Operating Mode

ECB mode simply takes the piece off the input message one block (128 bits) at a time and encrypts it using the same key until no more pieces are left. This process is shown above in Figure 2, which displays the computation for both serial (one block at a time) and parallel encryption. Serial encryption and decryption supports slower data rate, whereas parallel encryption/decryption provides higher data rate using more resources.

A. Steps for Encryption and Decryption

The steps involved in the algorithm are described below:

1. Initial Round: Modulo 2 addition of the 4×4 state matrix and the round key, also represented as a 4×4 matrix, is performed.
2. Rounds:

- SubByte: A non-linear substitution step where each byte is replaced with another according to a lookup table called S-Box. The AES S-Box is a 256 entry table composed of two transformations such as multiplicative inverse in GF (2^8) and an affine Transformation. For decryption, inverse S-Box is obtained by applying inverse affine transformation followed by multiplicative inversion in GF(2^8) Inversion in the GF(2^8) field, modulo the irreducible polynomial $m(x) = x^8 + x^4 + x^3 + x + 1$.. Where Affine Transformation is a transformation consisting of multiplication by a matrix followed by the addition of a vector. Affine transformation defined as: $Y = AX^{-1} + B$, where A is a 8×8 fixed matrix and B is a 8×1 Vector.

- ShiftRows: A transition step where each row of the state is shifted cyclically to the left using 0,1,2 and 3 byte offset for encryption while for decryption, rotation is applied to the right. This transformation can be defined as follows:

  $S'_{r,c} = S_{r\,((c + \text{shift}\,(r,\,4)\,)\bmod 4}$ where shift value shift (r, 4) depends on the row number are as follows: shift(1,4) = 1; shift(2,4) = 2; shift(3,4) = 3;

- Mixcolumn: A mixing operation which operates on the columns of the state, combining the 4 bytes in each column. Each column of the state matrix is multiplied by a constant fixed matrix. The transformation can be written as the following matrix multiplication formula:

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}$$

$$\text{for } 0 \le c < 4$$

- AddRoundKey: Each byte of the state is combined with the roundkey; each round key is derived from the cipher key using a key schedule. The above 4 rounds are iteratively carried out 9 times. It is a simple bit wise XOR operation is performed between each byte of the state and the roundkey which is generated from the cipher key using Rijndael key schedule algorithm. The operation can be performed in the following way:

$$\left[S'_{0,c} S'_{1,c} S'_{2,c} S'_{3,c}\right] = \left[S_{0,c} S_{1,c} S_{2,c} S_{3,c}\right]$$
$$\oplus [W_{\text{rounder}+c}] \text{ for } 0 \le c < 4.$$

Round keys, are derived from the 128-bit user key by means of the key schedule [6]. It consists of two components: key expansion and round key selection. The total number of round keys required is equal to N, + 1 (where Nr = Number of rounds = 10). Although there are 10 rounds, eleven keys are needed because one extra key is needed in the Initial round. The key expansion algorithm uses bit-wise additions modulo 2 of 32-bit values obtained from user key combined with byte substitution, byte rotation to right and round constants (RCons) addition as shown in Figure 3.

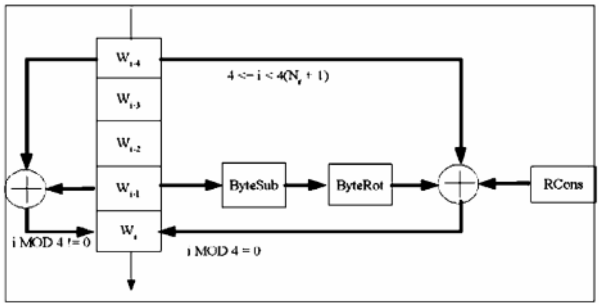

Fig.3: Key Expansion

There are two ways decryption engines can be designed; one is straightforward method and second is equivalent method, In the straightforward decryption method, the sequence of the transformations differs from that of the encryption approach. It comprises of an inverse of the final round, inverses of the nine rounds, followed by the initial data/key addition. The inverse of the round is found by inverting each of the transformations in the round. The inverse of ByteSub is obtained by applying the inverse of the affine transformation and taking the multiplicative inverse in GF(28) of the result. The same set of keys are used in encryption and decryption process, but are used in reverse order. In the equivalent approach that the decryption has the same sequence of transformations as encryption (with transformations replaced by their inverses) but with different set of keys. The decryption keys are generated by keeping the first and the last 128-bit encryption subkeys as it is and performing InvMixColumn operation on remaining intermediate 128-bit sub-keys.

3. Final Round: In this round, each of the transformations SubByte, ShiftRows, & AddRoundKey, is performed only once. It is noted that the Mixcolumns operation is not performed in this round.

3. Computational Issues of AES

In AES, each byte is considered as an element of Finite field of characteristic 2 with 8 terms, which is also known as Galois field, GF(2^8). GF(2^8) is actually a field of 256 elements and each element can be represented by a polynomial of degree 7 with coefficient {0,1}.

Addition and Subtraction: The addition of two elements is performed by the bitwise XOR operation of the coefficients for the corresponding powers in the polynomials for the two elements.

Example: $(x^7 + x^5 + x^3 + x + 1) + (x^6 + x^5 + x + 1)$

$= (x^7 + x^6 + x^3)$

Multiplication: The multiplication can be obtained by performing ordinary polynomial multiplication and then taking the remainder of it by an irreducible polynomial of degree 8, $m(x) = (x^8 + x^4 + x^3 + x + 1)$

Example: $(x^6 + x^4 + x^2 + x + 1)\ (x^7 + x + 1)$

$= (x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1)$

$= (x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 +1)$

modulo $(x^8 + x^4 + x^3 + x + 1)$

$= (x^7 + x^6 + 1)$

4. OPTIMIZATION OF 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[3]. A round can now be done with 16 table lookups and 12 32-bit XOR operations, followed by four 32-bit XOR in the AddRoundKey step. 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[11].Using a byte-oriented approach, it is possible to combine the SubBytes, ShiftRows, and MixColumns steps into a single round operation.

5. SECURITY OF AES

No major security attack has been proven successful against the AES till now. The security strength of AES is described briefly against different kinds of attacks.

5.1. Brute Force Attack

The minimum length of AES cipher key is 128 bits that provides 2^{128} possible keys. Performing exhaustive search in this huge key space is considered infeasible[8]. Thus the brute-force attack against AES with current and projected technology is considered impractical [3].

5.2. Mathematical Attack

AES uses S-box substitution table which is generated by determining the multiplicative inverse for a given number in Galois finite field and has the capability to resist the linear and differential cryptanalysis. Recently Warren D. Smith has defined an analytical method and claimed that there is a possibility of existence of a cracking algorithm that will be able to extract the AES 256 bits key from any random plaintext-ciphertext pairs [8]. However, it was just an assumption and he hasn't provided any cracking algorithm. As a result, till now, AES algorithm doesn't have any mathematical property that can be exploited by an attacker to reduce the effective key length and to gain success against AES.

5.3. Timing Attack

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. However, AES is less secure against side channels attacks. A number of side channel attacks have been proven successful including timing attack, power consumption, electromagnetic radiation, cache-collision timing attacks [2] etc. Timing attack is a ciphertext-only attack which is actually an implementation level attack that may occur in any implementation that doesn't run in fixed time, rather depends on the input. The high speed software implementation of AES is susceptible to this kind of attacks as it performs a sequence of S-box lookups that take variable time and dependent on the key. Daniel Bernstein has presented an example of such attack against a server running the OpenSSL AES implementation [4]. By selecting the cryptographic primitives which will allow efficient constant-time implementation we can avoid timing attack. However, it is extremely difficult to develop such primitives that are independent of the key as well as fast. It is also possible to circumvent timing attack by avoiding the use of S-box which will make the AES a bit slow. Another way is the addition of a delay to the faster operations to hide the timing differences but it will also cause performance penalty.

6. CONCLUSION AND FUTURE SCOPE

In this paper, we have studied AES encryption and decryption schemes and have highlighted some of the important mathematical properties as well as the security issues of AES algorithm. Since AES provides better security and has less implementation complexity, it has emerged as one of the strongest and most efficient algorithms in existence today. Hence, the optimal solution is the use of a hybrid encryption system in which typically AES is used to encrypt large data block. The future work can done for the distribution of secret key that is considered as a critical issue of AES like other symmetric encryption algorithm.

REFERENCES

[1] FIPS PUB 197: the Official AES Standard. http://www.csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[2] J. Bonneau and I. Mironov. Cache-Collision Timing Attacks Against AES. 2005.http://research.microsoft.com/users/mironov/papers/aestiming.pdf.

[3] A. J. Elbirt,W. Yip, B. Chetwynd, C. Paar, "An FPGA-based Performance Evaluation of the AES Block Cipher Candidate Algorithm Finalists", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 9, 545–557, 2001.

[4] D. J. Bernstein, "Cache-timing Attacks on AES", April 2005, http://cr.yp.to/antiforgery/cachetiming-20050414.pdf.

[5] V. R. Joan Daemen, "AES Proposal: Rijndael, Version 2, AES Submission", 1999.http://csrc.nist.gov/CryptoToolkit/aes/rijndael/Rijndaelammended. pdf.

[6] John Carroll, Computer Security, 3rd ed., Butterworth-Heinemann 1997.

[7] Joan Daemen, Steve Borg and Vincent Rijmen, "The Design of Rijndael: AES The Advanced Encryption Standard", Springer, 2002.

[8] W. D. Smith. 1. AES Seems Weak. 2. Linear Time Secure Cryptography, http://eprint.iacr.org/2007/248.pdf, 2007.

[9] William Stallings, "Cryptography and Network Security Principles and Practices", Prentice Hall, November 16, 2005.

[10] Kofahi, N.A. Turki Al-Somani Khalid Al-Zamil, "Performance Evaluation of Three Encryption/decryption Algorithms, Circuits and Systems", 2003, MWSCAS '03. Proceedings of the 46th IEEE International Midwest Symposium on, 2:790–793, 27-30 December 2003.

[11] P. Paillier and J. Villar, "Trading One-wayness Against Chosen Ciphertext Security in Factoring-based Encryption", Advances in Cryptology-Asiacrypt, Pages 553–558, 2006.