International Journal of Application or Innovation in Engineering & Management (IJAIEM)
Web Site: www.ijaiem.org Email: editor@ijaiem.org, editorijaiem@gmail.com
Volume 2, Issue 4, April 2013 **ISSN 2319 - 4847**

VLSI Implementation of Advanced Encryption Standard using Rijndael Algorithm

Aditya Agarwal

Assistant Professor, Electronics and Communication Engineering
SRM University, NCR Campus, Ghaziabad, India

ABSTRACT

Now a daysAdvanced Encryption Standard (AES) is the most efficient public key encryption system based on Rijndael Algorithm that can be used to create faster and efficient cryptographic keys. AES generates keys through the properties of the Rijndael Algorithm instead of conventional method of the key generation. Althoughmany encryption algorithms can be relatively efficiently implemented in software on general-purpose or embedded processors, there is still a need for special purpose cryptographic processors. First of all, high throughput applications, such as the encryption of the physical layer of Internet traffic, require an ASIC that does not affect the data throughput. For example, software implementation of the Rijndael algorithm on a Pentium 200 Pro yields a throughput of around 100 Mbits/sec, which is too slow for high-end Internet routers. Moreover, in terms of mobile application like cellular phones, PDA's, etc., software implementation on general-purpose processors consumes much more power than special purpose ASIC's do. Last of all, it is often the case that applications require the encryption logic be physically isolated from the rest of the system, so that the encryption can be secured more easily. In this case a hardware accelerator is a more suitable solution as well.

Keywords: Cryptography, Rijndael, Encryption, Decryption, Cypher, Inverse cypher.

1. INTRODUCTION

Advanced Encryption Standard (AES) was announced by National Institute of Standards and Technology (NIST) as U.S. FIPS PUB 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 [1][2]. It became effective as a standard May 26, 2002. As of 2009, AES is one of the most popular algorithms used in symmetric key cryptography [6]. 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 [22].This standard specifies the Rijndaelalgorithm, a symmetric block cipher that can process datablocks of 128 bits using cipher keyswith lengths of 128, 192 and 256 –bits [9][10].

TheAES specifies a cryptographic algorithm that can be used to protect electronic data. The AES algorithm is asymmetric block cipher that can encrypt (encipher) and decrypt (decipher) information.Encryption converts data to an unintelligible form called ciphertext; decrypting the ciphertextconverts the data back into its original form, called plaintext.It can widely use for electronic commerce, secure communication, etc.

Computational efficiency: The evaluation of computational efficiency will be applicable to both hardware and software implementations. Round 1 analysis by NIST will focus primarily on software implementations and specifically on one key-block size combination, more attention will be paid to hardware implementations and other supported key-block size combinations during Round 2 analysis [21]. Computational efficiency essentially refers to the speed of the algorithm. AES Rijndael shows very good software performance. Rijndael's key setup time is fast[6][10].Like software, hardware implementations can be optimized for speed or for size. Rijndael has the highest throughput of any of the finalists for feedback modes and second highest for non-feedback modes. For the 192 and 256-bit key sizes, throughput falls in standard and unrolled implementations because of the additional number of rounds [4]. For fully pipelined implementations, the area requirement increases, but the throughput is unaffected.

2. CONDITIONS OF AES

2.1 AES: Inputs and Outputs

The input and output for the AES algorithm each consist of sequencesof 128 bits (digits with values of 0 or 1). These sequences will sometimes be referred to as blocks and the number of bits they contain will be referred to as their length. The Cipher Keyfor the AES algorithm is a sequence of 128, 192 or 256 bits [6][9].

2.2 Bytes

The basic unit for processing in the AES algorithm is a byte**,** a sequence of eight bits treated as a single entity. The input, output and Cipher Key bit sequences are processed as arrays of bytes that are formed by dividing these sequences into groups of eight contiguous bits to form arrays of bytes[3][10]. For an input, output or Cipher Key denoted by a, the bytes in the resulting array will be referenced using one of the two forms, $a_{n,}$ where n is Key length.

2.3 The State

The AES algorithm's operations are performed on a two-dimensional array of bytes called the State. The State consists of four rows of bytes, each containing *Nb*bytes, where *Nb*is the block length divided by 32. In the State array denoted by the symbol *s*, each individual byte has two indices, with its row number *r* in the range 0 <*r* < 4 and its column number *c* in the range 0 < *c* < *Nb*. This allows an individual byte of the State to be referred to as either $s_{r,c}$. For this standard, *Nb*= 4, i.e., 0 < *c* < 4 [11][19].

At the start of the Cipher and Inverse Cipher the input – the array of bytes $in_0, in_1, \ldots in_{15}$ – is copied into the State array. The Cipher or Inverse Cipher operations are then conducted on this State array, after which its final value is copied to the output – the array of bytes $out_0, out_1, \ldots out_{15}$.

2.4AESAlgorithm Specification

For the AES algorithm, the length of the input block, the output block and the State is 128 bits. This is represented by *Nb*= 4, which reflects the number of 32-bit words (number of columns) in the State.For the AES algorithm, the length of the Cipher Key, *K*, is 128, 192, or 256 bits [6]. The key length is represented by *Nk*= 4, 6, or 8, which reflects the number of 32-bit words (number of columns) in the Cipher Key. For the AES algorithm, the number of rounds to be performed during the execution of the algorithm is dependent on the key size. The number of rounds is represented by *Nr*, where *Nr* = 10 when *Nk*= 4, *Nr* = 12 when *Nk*= 6, and *Nr* = 14 when *Nk*= 8 [5][8][9]].

For both its Cipher and Inverse Cipher, the AES algorithm uses a round function that is composed of four different byte-oriented transformations: 1) byte substitution using a substitution table (S-box), 2) shifting rows of the State array by different offsets, 3) mixing the data within each column of the State array, and 4) adding a Round Key to the State [9].

3. AES ALGORITHM

3.1 Cipher

At the start of the Cipher, the input is copied to the State array. After an initial Round Key addition, the State array is transformed by implementing a round function 10, 12, or 14 times (depending on the key length), with the final round differing slightly from the first ***Nr*** -1 rounds[9]. The final State is then copied to the output. The round function is parameterized using a key schedule that consists of a one-dimensional array of four-byte words derived using the Key Expansion routine.The Cipher is described as the individual transformations -**SubBytes ( )**, **ShiftRows ( )**, **MixColumns ( )**, and **AddRoundKey ( )** – process the State [4][9].

3.1.1 SubBytes( )Transformation

The **SubBytes( )** transformation is a non-linear byte substitution that operates independently on each byte of the State using a substitution table (S-box). This S-box, which is invertible, is constructed by composing two transformations:

1. Take the multiplicative inverse in the finite field GF(2^8), the element {00} is mapped to itself.
2. Apply the following affine transformation (over GF(2^8))[6][19]:

$$b_i = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i$$

for $0 \leq i < 8$ where *bi* is the i_{th} bit of the byte, and *ci* is the i_{th}bit of a byte *c* with the value {63} or {01100011}. Here and elsewhere, a prime on a variable (e.g., *b*') indicates that the variable is to be updated with the value on the right.

3.1.2 ShiftRows( ) Transformation

In the **ShiftRows( )** transformation, the bytes in the last three rows of the State are cyclically shifted over different numbers of bytes. The first row, *r* = 0, is not shifted.

$$s'_{r,c} = s_{r,(c+shift\,(r,Nb)) \bmod N\,b} \quad \text{for } 0 < r < 4 \quad \text{and} \quad 0 \leq c < \boldsymbol{Nb},$$

where the shift value *shift*(*r*,*Nb*) depends on the row number, *r*, as follows (recall that ***Nb***= 4):

$$shift\,(1,4) = 1\,; \quad shift\,(2,4) = 2\,; \quad shift\,(3,4) = 3\,;$$

This has the effect of moving bytes to "lower" positions in the row (i.e., lower values of *c* in a given row), while the "lowest" bytes wrap around into the "top" of the row (i.e., higher values of *c* in a given row)[9][20].

3.1.3 MixColumns( ) Transformation

The **MixColumns( )** transformation operates on the State column-by-column, treating each column as a four-term polynomial. The columns are considered as polynomials over GF(2^8) and multiplied modulo x^4+1 with a fixed polynomial *a*(x), given by

$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$

3.1.4 AddRoundKey( ) Transformation

In the **AddRoundKey( )** transformation, a Round Key is added to the State by a simple bitwise XOR operation. Each Round Key consists of ***Nb*** words from the key schedule. Those *Nb* words are each added into the columns of the State, such that

$$[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}] = [s_{0,c}, s_{1,c}, s_{2,c}, s_{3,c}] \oplus [w_{round * nb + c}] \text{ for } 0 \le c < Nb$$

where [*wi*] are the key schedule words described below, and *round* is a value in the range $0 \le round < Nr$. In the Cipher, the initial Round Key addition occurs when *round* = 0, prior to the first application of the round function. The application of the AddRoundKey( )transformation to the *Nr*rounds of the Cipher occurs when $1 \le round \le Nr$ *[3][6]*.

3.2 Key Expansion

The AES algorithm takes the Cipher Key, *K*, and performs a Key Expansion [6] routine to generate a key schedule. The Key Expansion generates a total of *Nb*(*Nr* + 1) words: the algorithm requires an initial set of ***Nb***words, and each of the ***Nr*** rounds requires ***Nb***words of key data. The resulting key schedule consists of a linear array of 4-byte words, denoted [*wi*], with *i*in the range $0 \le i < Nb(Nr + 1)$[11].

SubWord( ) is a function that takes a four-byte input word and applies the S-box to each of the four bytes to produce an output word. The function **RotWord()** takes a word $[a_0, a_1, a_2, a_3]$ as input, performs a cyclic permutation, and returns the word $[a_1, a_2, a_3, a_0]$. The round constant word array, **Rcon [i]**, contains the values given by $[x^{i-1}, \{00\}, \{00\}, \{00\}]$, with x^{i-1} being powers of *x* (*x* is denoted as {02}) in the field GF(2^8), (note hat*i*starts at 1, not 0).

Every following word, **w[i]**, is equal to the XOR of the previous word, **w[i -1]**, and the word ***Nk***positions earlier, **w[i -*Nk*]**,For words in positions that are a multiple of ***Nk***, a transformation is applied to **w[i -1]** prior to the XOR, followed by an XOR with a round constant, **Rcon [i]**. This transformation consists of a cyclic shift of the bytes in a word (**RotWord( )**), followed by the application of a table lookup to all four bytes of the word (**SubWord( )**) [8][9][10].

3.3 Inverse Cipher

The Cipher transformations can be inverted and then implemented in reverse order to produce a straightforward Inverse Cipher for the AES algorithm. The individual transformations used in the Inverse Cipher [6]-**InvShiftRows()**, **InvSubBytes()**, **InvMixColumns()**, and **AddRoundKey()** – process the State[6][21].

3.3.1 InvShiftRows() Transformation

InvShiftRows() is the inverse of the **ShiftRows()** transformation. The bytes in the last three rows of the State are cyclically shifted over different numbers of bytes. The first row, $r = 0$, is not shifted. The bottom three rows are cyclically shifted by ***Nb-shift*** (r, *Nb)* bytes, where the shift value *shift(r,Nb)* depends on the row number[4].

Specifically, the **InvShiftRows()** transformation proceeds as

$$S'_{r, ,(c+shift\ (r,Nb)\ \mathrm{mod} Nb)} = S_{r,c} \qquad \text{for } 0 < r < 4 \quad \text{and} \quad 0 \le c < Nb$$

3.3.2 InvSubBytes() Transformation

InvSubBytes() is the inverse of the byte substitution transformation, in which the inverse S-box applied to each byte of the State. This is obtained by applying the inverse of the affine transformation followed by taking the multiplicative inverse in GF(2^8)[8].

3.3.3 InvMixColumns( ) Transformation

InvMixColumns() is the inverse of the **MixColumns()** transformation. **InvMixColumns()** operates on the State column-by-column, treating each column as a four- term polynomial[10]. The columns are considered as polynomials over GF(2^8) and multiplied modulo $x^4 + 1$ with a fixed polynomial $a^{-1}(x)$, given by

$a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}$.

3.3.4 Inverse AddRoundKey( ) Transformation

AddRoundKey(), which was described earlier, is its own inverse, since it only involves an application of the XOR operation.

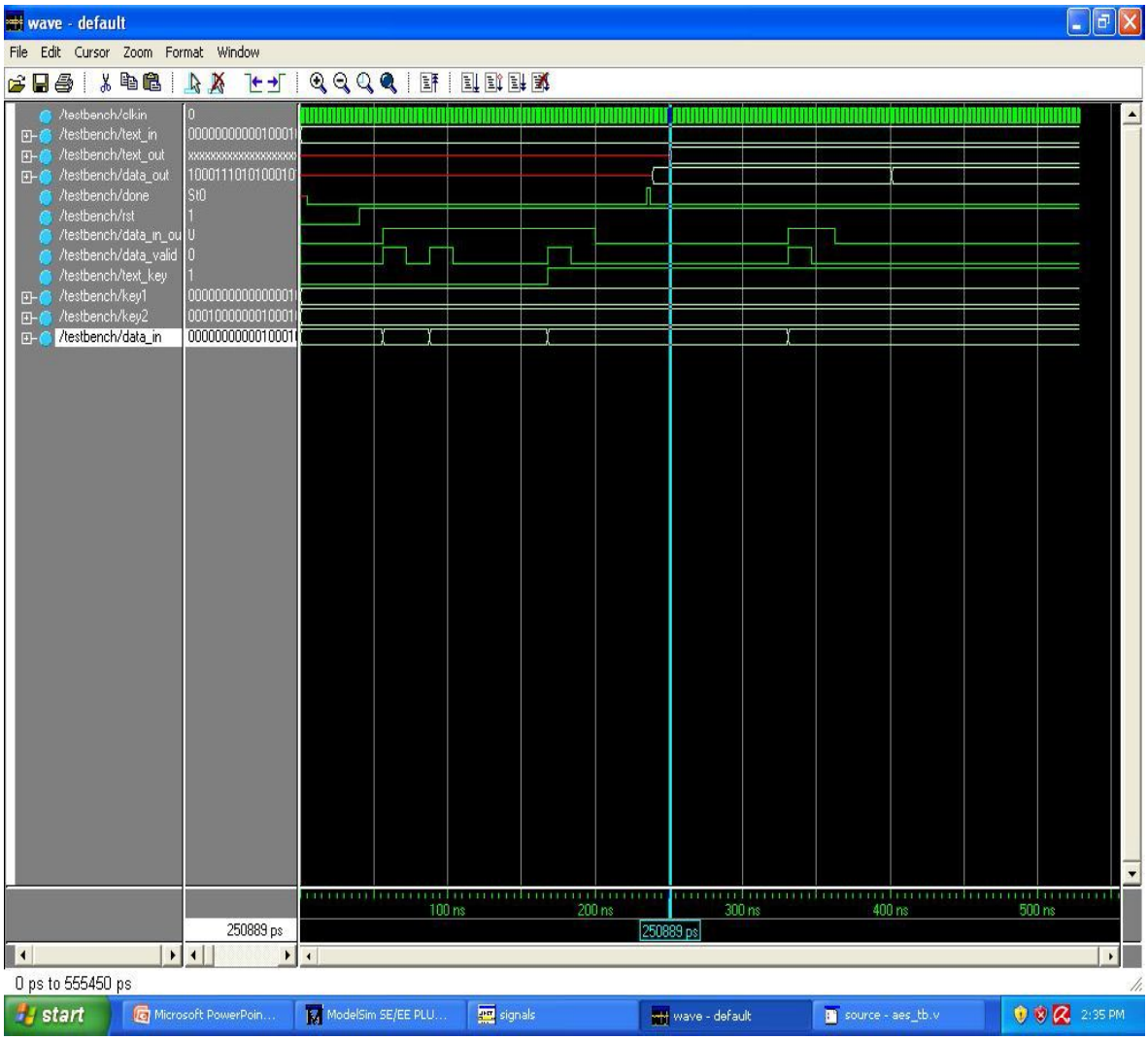
3.3.5 Equivalent Inverse Cipher

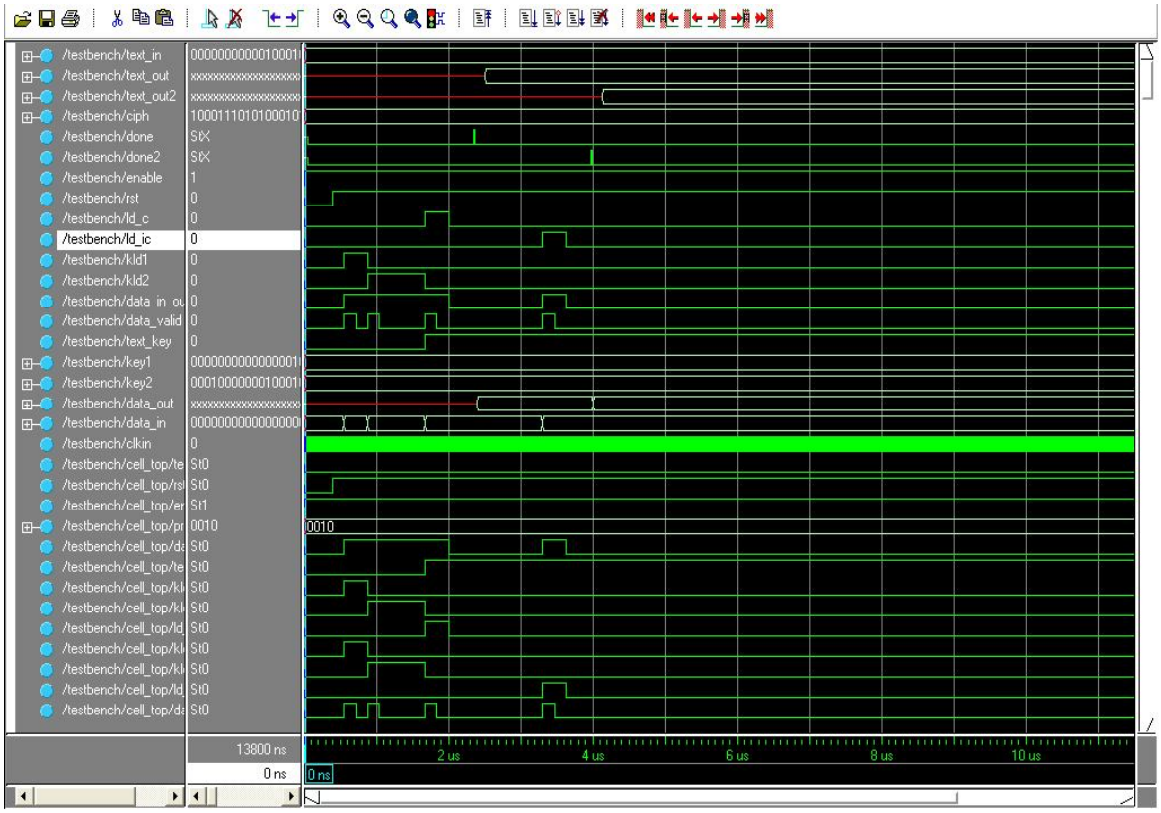
AES allow for an Equivalent Inverse Cipher that has the same sequence of transformations as the Cipher (with the transformations replaced by their inverses). This is accomplished with a change in the key schedule.The two properties that allow for this Equivalent Inverse Cipher are as follows:

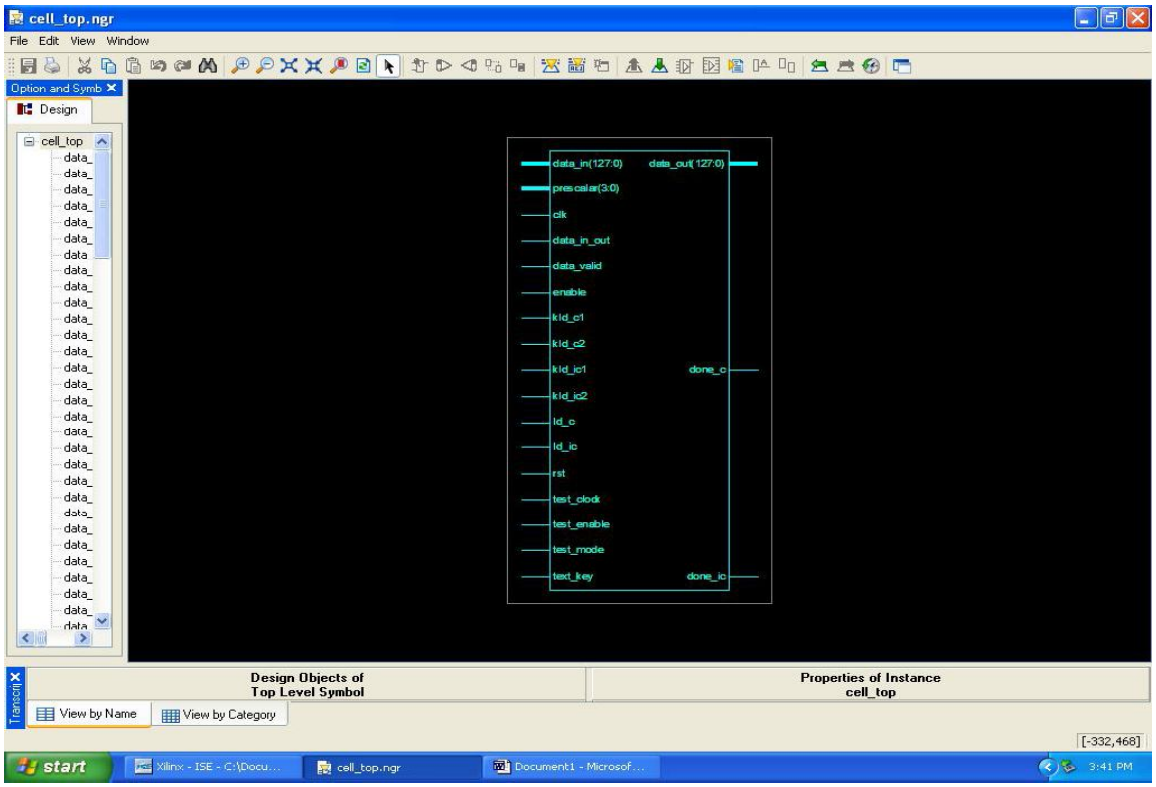
1. The **SubBytes( )** and **ShiftRows( )** transformations commute; that is, a **SubBytes( )** transformation immediately followed by a **ShiftRows( )**transformation is equivalent to a **ShiftRows( )** transformation immediately followed by a **SubBytes( )** transformation. The same is true for their inverses, **InvSubBytes( )** and **InvShiftRows.**
2. The column mixing operations - **MixColumns( )** and **InvMixColumns( )** – arelinear with respect to the column input, which means

InvMixColumns (state XOR Round Key) =InvMixColumns (state) XOR InvMixColumns (Round Key).
The equivalent inverse cipher is defined by reversing the order of the **InvSubBytes( )** and **InvShiftRows( )** transformations and byreversing the order of the **AddRoundKey( )** and **InvMixColumns( )** transformations used in the "round loop" afterfirst modifying the decryption key schedule for *round* = 1 to *Nr*-1 using the **InvMixColumns( )** transformation. The first and last *Nb*words of the decryption keyschedule shall *not* be modified in this manner [9][10][22].

4. Experimental Results:







5. APPLICATION OF ADVANCED ENCRYPTION STANDARD

Many devices are constrained devices that have small and limited storage and computational power, for constraineddevices AES can also be applied [15][16][17][20].

- For wireless communication devices like PDA's multimedia cellular phones AES can apply.

- It can be used for security of Smart cards, wireless sensor networks, wireless mesh Networks.
- AES have high computational efficiency, so as to be usable in high speed applications, such as broad band links.
- AES is very well suited for restricted-space environments where either encryption or decryption is implemented. It has very low RAM and ROM requirements.
- Web servers that need to handle many encryption sessions.
- Any kind application where security is needed for our current cryptosystems.

6. CONCLUSION

Advanced Encryption Standard offers the highest strength-per-key-bit of any known public-key system of first generation techniques. AES offers the high level of security with strong key system, computational power is high. Integrated circuit space is limited for smart card, wireless devices. The development of standards is a very important for the use of a cryptosystem. Standards help to ensure security and interoperability of different implementations of one cryptosystem.AESperforms encryption and decryption very well across a variety of platforms, including 8-bit and 64-bit platforms, and DSPs. AES Rijndael's high inherent parallelism facilitates the efficient use of processor resources, resulting in very good software performance. Rijndael's key setup time is fast. There are several major organizations that develop standards like International Standards Organization (ISO), American National Standards Institute (ANSI), Institute of Electrical and Electronics Engineers (IEEE), Federal Information Processing Standards (FIPS).The most important for security in information technology are the in addition secure communication, Advanced Encryption System (AES) enabling technology for numerous wireless communication systems.

References:

[1] AES page available via http://www.nist.gov/CryptoToolkit.4
[2] Computer Security Objects Register (CSOR): http://csrc.nist.gov/csor/.
[3] J. Daemen and V. Rijmen, AES Proposal: Rijndael, AES Algorithm Submission,September 3, 1999, available at [1].
[4] J. Daemen and V. Rijmen, The block cipher Rijndael, Smart Card research andApplications, LNCS 1820, Springer-Verlag, pp. 288-296.
[5] B. Gladman's AES related home page http://fp.gladman.plus.com/cryptography_technology/.
[6] FIPS PUB 197: The official Advanced Encryption Standard
[7] A. Menezes, P. van Oorschot, and S. Vanstone, Handbook of Applied Cryptography,CRC Press, New York, 1997, p. 81-83.
[8] J. Nechvatal, et. al., Report on the Development of the Advanced Encryption Standard(AES), National Institute of Standards and Technology, October 2, 2000, available at [1].
[9]http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
[10] William Stallings, Cryptography and Network Security, principles and practices, 4th Edition
[11] Whitfield Diffie and Martin Hellman, "New Directions in Cryptography", IEEETransactions on Information Theory, vol. IT-22, Nov. 1976, pp: 644–654. (pdf)
[12] AJ Menezes, PC van Oorschot, and SA Vanstone, Handbook of Applied Cryptography. ISBN 0-8493- 8523-7.
[13]A. Lee, NIST Special Publication 800-21, Guideline for Implementing Cryptography in the Federal Government, National Institute of Standards and Technology, November 1999.
[14] S. Brands, "Untraceable Off-line Cash in Wallets with Observers", In Advances in Cryptology — Proceedings of CRYPTO, Springer-Verlag, 1994.
[15] avies, D.W.; W.L. Price (1989). Security for computer networks, 2nd ed..John Wiley & Sons.
[16] Biham, Eli, Orr Dunkelman, Nathan Keller: Enhancing Differential- linear Cryptanalysis. ASIACRYPT 2002: pp254–266
[17] Junod, Pascal. "On the Complexity of Matsui's Attack."Selected Areas in Cryptography, 2001, pp199–211.
[18] Kaliski, Burton S., Matt Robshaw: Linear Cryptanalysis Using Multiple Approximations. CRYPTO 1994: pp26–39
[19] NIST Laboratories. National Institute of Standards and Technology.Retrieved on 2009-08-13.
[20] Nicolas Courtois, Josef Pieprzyk, "Cryptanalysis of Block Ciphers with Over defined Systems of Equations". pp267–287, ASIACRYPT 2002.
[21] Joan Daemen and Vincent Rijmen, "The Design of Rijndael: AES - The Advanced Encryption Standard." Springer-Verlag, 2002.ISBN 3-540-42580-2.
[22] National Security Agency.United States Signals Intelligence Directive 18. National Security Agency July 27, 1993. Last access date March 23, 2007.