Oriental Journal of Computer Science & Technology Vol. **3**(1), 83-88 (2010)

Analysis of AES hardware and software implementation

R. VELAYUTHAM[1] and D. MANIMEGALAI[2]

Department of Computer Science and Engineering, Einstein College of Engineering, Tirunelveli, (India).
Department of IT, National Engineering College, Kovilpatti, (India).

(Received: April 12, 2010; Accepted: June 04, 2010)

ABSTRACT

In November 2001 NIST published Rijndael as the proposed algorithm for AES(Advanced Encryption Standard). The result of new attack methods shows that there may be some missing part in the design of S-box and key schedule with AES algorithm. The problem is the weakness of linearity existing in the S-box and key schedule. In order to keep away from the new attacks and implement the AES in software and hardware provides higher level of security and faster encryption speed; we analyze in detail the AES algorithm and propose a new implementation scheme for increasing complexity of nonlinear transformation in design of S-box. Implementation scheme with Java and the use of reconfigurable coprocessor as a cryptography hardware is proposed.

Keywords: AES Hardware, Software implementation.

INTRODUCTION

Initially the AES algorithm was believed of much more security and of no weakness in the ideas of most people. However, recently cryptanalysts have also obtained some breaking methods on the AES. This paper first analyzes the AES algorithm and point out the weakness of linearity existing in the design of S-box and key schedule from the standpoint of new attack methods. Secondly, it proposes to improve the design of S-box to increase its nonlinearity and complexity in the implementation scheme for AES. The S-box used in this scheme has stronger resistance against the new attacks than the standard one. Thirdly, this scheme is used in the LAN for secured communication. Fourthly, it proposes hardware design of AES where parallel processing and pipelining is possible. Hardware systems offer superior performance with higher throughput. Helion Technology claims that speed exceeding 16 Gbps for FPGA and 25 Gbps for ASIC design is available.

Rijndael Algorithm

Rijndael algorithm is a symmetric block cipher with a block length of 128 bits and supports key lengths of 128, 192 and 256 bits. The minimum key length is 128 bits. Both block length and key length can be extended very easily to multiples of 32 bits. The procedures of AES encryption and decryption in Cipher Block Chaining (CBC) Mode [14] are shown in Figure 1 and 2, respectively. In CBC (Cipher Block Chaining) mode the input to the encryption algorithm is the XOR of the next 64 bits of plaintext and the preceding 64 bits of cipher text. It finds its application in general purpose block oriented transmission and authentication.

In Figure 1, the block size is 128 bits according to AES specification at present. The number of rounds (Nr) depends on the length of main keys (Nk) and the number of block columns (Nb), i.e. Nr = Nk + Nb + abs (Nk-Nb), where Nk = 4. e.g. if Nk = 4 and Nb = 8, then Nr = 14. The round transformation is composed of four different transformations.

- *SubBytes Transformation: Uses an S-box to perform a byte-by-byte substitution of the block.*
- *Shift rows: A simple permutation.*
- *Mix columns: A substitution.*
- *Add round key: A simple bitwise XOR of the current block with a portion of the expanded key.*

In the Rijndael algorithm, all steps are invertible. The decryption is shown in Figure 2. It is classified in to the straightforward decryption algorithm and the equivalent decryption algorithm. For SubBytes transformation, ShiftRows transformation, MixColumns transformation and AddRoundKey addition, an inverse function is used in the decryption algorithm. Decryption algorithm make use of the expanded key in the reverse order.

The Problem of Aes Security

Rijndael has been designed to have very strong resistance against the classical approximation attacks, such as linear cryptanalysis, differential cryptanalysis etc. However since Rijndael is derived form Square algorithm, and is very algebraic, new algebraic and improved differential attacks have appeared.

(1) Strength against Known Attacks

The AES specifies three key sizes, 128, 192 and 256 bits. Their number of possible keys is 3.4 x 1038, 6.2 x 1057, and 1.1 x 1077, respectively. In comparison, DES keys are 56 bits long, which means there are approximately 7.2 x 1016 possible DES keys. Thus, there are on the order of 1021 times more AES 128-bit keys than DES 56-bit keys. AES with 128-bit keys has stronger resistance to an exhaustive key search than DES. Although classic differential and linear attacks are invalid for the AES, they have been extended in several ways for recent years and new attacks have been published that are relative to them. The newest attack combined boomerang and the rectangle attack with related-key differentials uses the weaknesses of few nonlinear transformations in the key schedule algorithm of ciphers, and can break some reduced-round versions of AES. For example, it can break 192-bit 9-round AES by using 256 different related keys. The Square attack is also valid for Rijndael, as Rijndael inherits many properties form Square. The original Square attack can break round-reduced variants of Rijndael up to 6 or 7 rounds (i.e. AES-128 and AES-192) faster than an exhaustive key search .

(2) Weakness of the existing S-box and key schedule

S-box is only one component to implement nonlinear transformation in the AES. The cryptographic strength of the AES depends strongly on the choice of S-box. Many cryptographists have discovered that there are some weaknesses in the design of the existing S-box. For example, Y. B. Wang proved that the S-box has the properties of short periods and bad distribution, which may be a fatal weakness for AES. The new way of equivalent generating the S-box was found by Y. A. Zhang and D. G. Feng . It may be a great help to the algebraic attack. In order to make up the weakness of the existing S-box and improve the nonlinearity of S-box, W. Millan proposed to use the iterated hill climbing be used for self-inverse S-box[15] . J. M. Liu, etc thought that the algebraic expression of AES S-box is very simple and fewer terms are involved [16]. They proposed to adopt a new algebraic method to design the S-box. In comparison with the cipher itself, the Rijndael key schedule appears to be more of a linear design. It has a much slower diffusion structure than the cipher, and contains relatively few nonlinear elements. It can almost be described as a collection of 32 linear feedback shift registers LFSRs, working in parallel. This implies that for related keys, i.e., pairs of unknown keys with known differences, one can in part predict the differences of the individual round keys.

Proposed Methodology

i) Software

In order to reduce the cost of implementation, we adopt the method of software to implement AES algorithm. In addition, Java is an object oriented programming language with many interesting security features (e.g. sandbox paradigm, byte code verification) [17]. Hence, it is proposed to implement AES in Java. Apart from its security, the efficiency of AES is of main interest for application developers. In this scheme, the speeds of data encryption and decryption are selected as the performance indices to evaluate the efficiency of AES. The speed of data encryption is defined as the quotient obtained by dividing the bit

number of plaintext input by corresponding encryption time in second. So is the speed of data decryption. The scheme of the proposed implementation uses ten data members as shown in Fig. 3. The cipher system involves three important methods, i.e. Key expansion shown in Fig. 4, Cipher and InvCipher shown in Fig. 5. They implement the function of the key schedule, data encryption and data decryption, respectively.

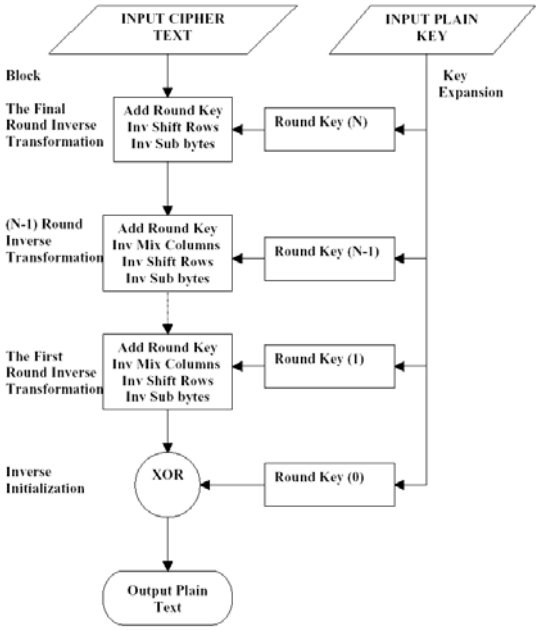

Fig. 1 Encryption

```
public enum KeyLength {128 bits,i92 bits,256 bits}
//Select the length of cipher key
private int Nb;//Block size in word
private int Nk;//Key size in word
private int Nr;//Number of rounds

private byte[] Key;//Cipher Key array
private byte[,] Sbox;//S-Box
private byte[,] InvSbox;//Inverse S-Box
private byte[,] w;//Key schedule table
private byte[,] Rcon;//Round constants table
private byte[] State;// Intermediate Cipher result
pictured as array
```

Fig. 3 Data Members Used in the Proposed

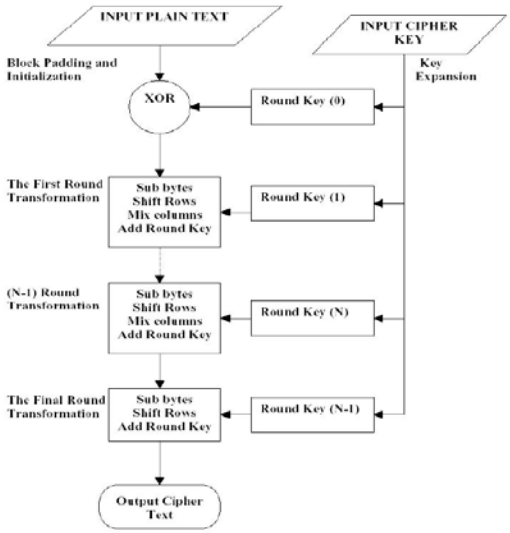

Fig. 2 Decryption

Scheme:

```
Public void KeyExpansion()
{
        this.w=new byte[Nb*(Nr+1),4];
        for (int row=0;row<Nk;++row)
        {
        this.w[row,0]=this.key[4*row];
this.w[row,1]=this.key[4*row+1};
        this.w[row,2]=this.key[4*row+2];
        this.w[row,3]=this.key[4*row+3];
}
byte[] temp=new byte[4];
for (int row=Nk;row<Nb*(Nr+1);++row)
{
temp [0]=this.w[row-1,0];
temp [1]=this.w[row-1,1];
temp [2]=this.w[row-1,2];
temp [3]=this.w[row-1,3];
if (row%Nk==0)
{
temp=SubWord(RotWord(temp));
temp [0]=
(byte)((int)temp[0]^(int)this.Rcon
[row/Nk,0]);
temp [1]=
(byte)((int)temp[1]^(int)this.Rcon
[row/Nk,1]);
```

```
temp [2]= (byte)((int)temp[2]^(int)this.Rcon [row/
Nk,2]);
temp [3]=   (byte)((int)temp[3]^(int)this.Rcon
[row/Nk,3]);
     }
     else if (Nk>6 && (row%Nk==4))
     {
           temp=SubWord(temp);
      }
       this.w[row,0]= (byte)((int) this.w[row-
Nk,0]^(int) temp[0]);
    this.w[row,1]= (byte)((int) this.w[row-Nk,1]^(int)
temp[1]);
     this.w[row,2]= (byte)((int) this.w[row-Nk,2]^(int)
temp[2]);
this.w[row,3]= (byte)((int) this.w[row-Nk,3]^(int)
temp[3]);
        }
}
```

Fig. 4 Method of Key Expansion in the Proposed Scheme

```
public void Encrypt(byte[] input,byte[] output)
{
this.State= new  byte[4,Nb]; //state=input
for(int i=0;i<(4*Nb);++i)
{
          this.State[i%4,i/4]=input[i];
}
AddRoundKey(0);
for(int round=1; round<=(Nr-1); ++round)
{
          SubBytes();
          ShiftRows();
          MixColumns();
          AddRoundKey(round);
}
SubBytes();
ShiftRows();
AddRoundKey(Nr);
for(int i=0;i<(4*Nb);++i) //output=state
{
          output[i]=this.state[i%4,i/4];
}
}
```

Fig. 5 Methods of Data Encryption and Decryption in the Proposed Scheme

There are two S-boxes to be designed in the proposed implementation scheme. Users can select either of the two according to their will. Where, one of them is a standard S-box described in. The other is an implemented S-box. In order to increase the nonlinearity and computational complexity of S-box, the improved S-box is designed by using the combination of the iterated hill climbing algorithm and a new algebraic method. It may have stronger resistance against new attacks than the standard S-box. How to increase the nonlinearity of key schedule is one of the important problems that we want to solve in the future. The default algorithm in our scheme is AES with 256 bits keys. AES with 256 bits keys has the highest security margin among three standard AES variants. Users can also customize an AES by using an improved S-box or / and expanded rounds [18]. Adding more rounds to Rijndael may increase the security margin to protect from new attacks. The speed of data encryption for AES using Java implementation is over that of RSA (45.8kb/s) using hardware implementation. The experimental results of Java implementation that our scheme is feasible, and has a good performance of encryption and decryption speed.

ii) Hardware:

Currently, several hardware implementation methods have been designed and published. There are many design choices encountered during hardware implementation of AES. In reality, these choices will be limited to its applications and budget. From the perspective of performance, major decision lies in the tradeoff between area and speed. For example, fast system is obtained at a cost of increased area, and vice versa. Before looking into different hardware architectures, basic hardware concepts are defined.

Pipelining: Replicating rounds and placing registers in between - Increases throughput.

Iterative Looping: One round of hardware design, which forces the algorithm to reuse the same hardware.

Loop unrolling: Refers to the process of unrolling multiple rounds.

Latency: An elapsed time between start to finish of encryption.

Although efficiency of hardware implementation was one of the evaluation criteria for choosing AES, only few hardware designs are presented for FPGA or ASIC platforms [9]. Analysis of routing was not mentioned in most papers. For FPGA target, routing placement is predetermined within the FPGA architecture and this is the cause for greater area in FPGAs compared to ASIC designs. On contrary, area of ASIC designs would be greatly affected by routing overhead, where the minimum bus length is 128 bit. ASIC's floorplan in [9] reports that the area estimations of routing were off by a factor of two. Previous experience with layout CAD tools helped me to realize the complexity of routing problems and its effect on chip area. In order to achieve optimized ASIC hardware design of AES, efficient routing algorithm is mandatory.

FPGA Coprocessor

Previous sections examined stand alone AES hardware implementations. Following section describes the design which supports a coprocessor. Organization of this architecture consists of a CPU with an aid of a FPGA coprocessor. Coprocessor design integrates software and hardware into a single system, along with the reconfigurable capability of FPGA. Generally, the CPU controls the overall system operations while FPGA is responsible for calculations involving extensive computations. Moreover, FPGA is reconfigurable that can be reprogrammed in few milliseconds. As an example, consider an embedded system with a coprocessor, connected over a network using an Ethernet. Depending on different situations, the FPGA can be reprogrammed dynamically according to the real-time status [11]. This allows FPGA to dynamically adjust to satisfy its surrounding requests.

Hardware platform that was researched is an embedded system by Wind River. The physical setup of this board includes an IBM PowerPC and Xilinx FPGA daughter card connected through a custom peripheral bus [12]. Many issues arise for the mixed system. First, communication between the processor and the FPGA must be managed. Type of communication protocol, management of control signals, and handling of crossing data needs to be developed. Second, debugging and simulating the coprocessor design should support the combination of software and hardware. Lastly, limited speed of the bus, connecting CPU and FPGA should be efficiently utilized. Designer should avoid degrading the overall system's performance by analyzing these parameters. Speed of the bus is the major bottleneck imposed on coprocessor systems. This is similar to the memory gap between processor and memory in the PC industry. For dividing the AES system, one idea would be to allow the processor to compute shift operation and assign rest to the FPGA.

CPU - ShiftRow and Control

FPGA - SubByte, MixColumn, and KeyAddition

Shift operations in hardware represent mere wiring. Moving this function may benefit the hardware implementation due to reduced amount of wire interconnects. In other words, decreased wire parasitic and smaller routing overhead. Moreover, most of the AES function would operate on the FPGA and thus minimal communication occurs between the CPU and the FPGA.

Conlusion

AES is a new cryptographic algorithm that can be used to protect electronic data. Its security has attracted cryptographist's attentions. The methods of new attacks welled that the design of existing S-box some weaknesses. The principal weakness is the problem of linearity in the S-box and key schedule. It is necessary to improve nonlinear transformations in the design of S-box and key schedule in order to protect from new attacks. Some measures against new attacks were adopted by improving the complexity of nonlinear transformation of S-box in the proposed implementation scheme such that it increases the security. To increase the data encryption and decryption time the implemented software design is embedded with FPGA. Such that the proposed work mix of software and hardware design generates an acceptable speed of data encryption and decryption and also provides security.

REFERENCES

1. A. Elbirt, Reconfigurable Computing for Symmetric-Key Algorithms, Ph.D. thesis, Department of Electrical Engineering, Worcester Polytechnic Institute, 2002.
2. P.Gutmann, “An open-sourcecryptographic coprocessor,”
3. K. Gaj and P. Chodowiec, \Hardware performance of the AES finalists-Survey and Analysis of results,” .
4. Helion Technology,” Tech. Rep., Helion, 2003
5. W. Trappe and L. Washington, Introduction to Cryptography with Coding Theory, Prentice Hall, New Jersey, 2002.
6. S. Morioka and A. Satoh, “An optimized s-box circuit architecture for low power aes design,” in Cryptographic Hardware and Embedded Systems - CHES 2002, C . K. Koc and C. Paar, Eds. Aug 13-15, 2002, Forth International Workshop, Redwood Shores, USA, pp. 172-186, Springer-Verlag.
7. B. Megarajan and S. Park, “Hardware implementation of aes (rijndael),” 2002, Webpage.
8. J.V. McCanny M. McLoone, “High performance single-chip fpga rijndael algorithm implementation,” in Cryptographic Hardware and Embedded Systems - CHES 2001, C.K.Koc and C. Paar, Eds. May 14-16, 2001, Third International Workshop, Paris, France, pp. 65-76, Springer-Verlag.
9. A. Lutz, J. Treichler, F. Frkaynak, H. Kaeslin, G. Basler, A. Erni, S. Reichmuth, P. Rommens, S. Oetiker, and W. Fichtner, “2gbit/s hardware realizations of rijndael and serpent: A comparative analysis,” in Cryptographic Hardware and Embedded Systems - CHES 2002, C_ . K. Ko_c and C. Paar, Eds. Aug 13-15, 2002, Forth International Workshop, Redwood Shores, USA, pp. 144-158, Springer-Verlag.
10. J. Golic and C. Tymen, Multiplicative masking and power analysis of aes,” in Cryptographic Hardware and Embedded Systems - CHES 2002, C_ . K. Ko_c and C. Paar, Eds. Aug 13-15,2002, Forth International Workshop, Redwood Shores, USA, pp. 198-212, Springer-Verlag.
11. “Re-Configurable Computing,” Tech. Rep., Wind River, August 2002, White Paper.
12. “Hardware Reference Designs for SBC405GP,” Tech. Rep., Wind River, 2001.
13. “Real World Experiences Designing For Mixed CPU + FPGA Systems,” Tech. Rep., Celoxica, August 2002, White Paper.
14. William Stallings, “Cryptography and network Security”, Third Edition.
15. W. Millan, “How to Improve the Nonlinearity of Bijective S-boxes,” Lecture Notes in Computer Science, vol. 1438, pp.181 - 192, Berlin: Springer-Verlag, 1998.
16. J. M. Liu, B. D. Wei, and X.G. Cheng,, “An AES S-Box to Increase Complexity and Cryptographic Analysis,”..Proc. of the 19th International Conference on Advanced Information Networking and Applications, pp. 724-728., Taiwan, China, 2005.
17. www.wikipedia.org
18. www.csrc.nist.gov/publications