Vidhyayana - ISSN 2454-8596
An International Multidisciplinary Peer-Reviewed E-Journal
www.vidhyayanaejournal.org
Indexed in: ROAD & Google Scholar

Design and implementation of MAES (modified Advanced Encryption Standard) algorithm in ANDROID for multimedia applications

Riddhi Somaiya

Ph.D. Scholar in Saurashtra University, Rajkot, Gujarat, India.

Dr. Atul Gonsai

Professor, Computer Science Department,

Saurashtra University, Rajkot, Gujarat, India.

Vidhyayana - ISSN 2454-8596
An International Multidisciplinary Peer-Reviewed E-Journal
www.vidhyayanaejournal.org
Indexed in: ROAD & Google Scholar

Abstract:

The most advanced and popular security protocol for wireless communication is WPA2 which uses Counter Mode with CBC-MAC (cypher block chaining message authentication code) Protocol (CCMP) for encryption. CCMP employs Advanced Encryption Standard (AES) as an encryption algorithm. AES is considered the most secure encryption algorithm available today for wireless multimedia communication. Recently there have been many modifications suggested in AES for improving its performance in terms of execution time, memory usage, power consumption, throughput, security etc...Also many new implementations of the AES algorithm in different hardware and software environments have been proposed. The efficiency of AES is an issue as it decreases when large input data like multimedia files are used. Nowadays due to the cheap and easy availability of the internet in mobile devices, large multimedia files are continuously being exchanged over the network. Many cases of unauthorized access and data theft have been observed. Also, with advancements in the Internet of Things (IoT), there is a desperate need for an efficient AES algorithm for security purposes. This paper proposes modifications in AES and compares the results of the implementation of standard AES and this modified version of AES in ANDROID devices. Results show an improvement of 70 per cent on average in the efficiency of AES for encryption and decryption of TEXT, Audio and Image files. This is achieved through modifications in Sbox and mix columns. This paper proposes an efficient AES algorithm while keeping the security level constant for wireless multimedia communication in ANDROID devices.

Keywords: AES, WPA2 protocol, CBC-MAC, WLAN, ANDROID

Introduction:

Wireless communication and especially mobile communication have become an integral part of our lives. With the advancements in hardware capacity, internet speed and the competitive nature of telecom operators, hundreds of apps requiring large multimedia communication are being used by everyone. Recently communication has shifted from only text to images, audio and video files which require large memory and bandwidth. Hence truckload of data is being exchanged every day on wireless networks. But along with all these perks, there comes a lot of security issues that these networks have to deal with. WLAN is being targeted by many fraudsters, hackers or intruders to steal important and private data using loopholes in the security of wireless communication protocols. Many security protocols have been developed to

address these issues, but none of them has been successful to address all the issues [1].

The main hurdle in improving the security of a protocol is the tradeoff between security and throughput. As we try to improve security by using complex encryption techniques, we get a reduction in throughput. So, we must take utmost care while trying to improve security as we don't want it at the cost of throughput. Hence there is a need for an efficient encryption algorithm that provides good security even with today's bulky multimedia communication [2].

Zeghid Medien, et al [3] eliminated the issue of textured zones in previous image encryption algorithms by modifying AES encryption algorithms. Pimpale Priyanka, et al [4] improved the security of the AES algorithm by making it multiple times complex using stronger diffusion and confusion techniques. Rizky Riyaldhi et al [5] were successful in optimizing the AES algorithm on an average 86%. R S Tanna, et al in [6] successfully implemented a modified version of AES algorithm in MATLAB and improved its efficiency by around 65%. Recently most of the modifications for optimization of AES are done at hardware level [7] [8] [9].

In this paper we have tried to increase the throughput of the AES algorithm while keeping its security unchanged. Initially we tried to implement standard AES algorithm in ANDROID studio 3.0.1 calculated its efficiency in terms of encryption, decryption time, network usage, cpu usage and memory usage for text, image and audio files. Then we modified the mix columns, sbox and polymatrix generation parts, implemented it and calculated the same parameters. On comparing these results we found around 70 percent improvement in the efficiency of the AES algorithm. We also implemented this modified AES in a demo chat app and performed secured communication between two android mobile phones by transmitting and receiving text, image, audio and video files.

Basic AES algorithm:

In the year 2001, Advanced Encryption Standard (AES) or Rijndael algorithm for encryption was standardized by the United States National Institute of Standards and Technology (NIST). The AES algorithm is a symmetric key algorithm and by using cryptographic keys of size 128, 192, or 256 bits it is capable of encrypting and decrypting 128 bits blocks of data [10].

Vidhyayana - ISSN 2454-8596
An International Multidisciplinary Peer-Reviewed E-Journal
www.vidhyayanaejournal.org
Indexed in: ROAD & Google Scholar

Here we have used a data block of 16 bytes that is 128 bits and a cipher key also of the same length. The AES algorithm works on a two-dimensional array of bytes which is called the State. Hence using 16 bytes of cipher key a 4 x 4 state matrix can be formed. The input data could be of any type i.e. Text, Audio or Image. We convert this data into blocks of 16 bytes each and encrypt these blocks one by one.

We are using ANDROID STUDIO 3.0.1 as the simulation tool. The complete ANDROID implementation of AES is done using JAVA programming in 3 parts:

1. AES initialization – sbox/inverse sbox generation, round constant generation, key expansion and polynomial/inverse polynomial matrix generation.

2. Cipher – Add round key, substitute byte, shift rows and mix columns.

3. Inverse cipher – inverse shift rows, substitute bytes, add round key and mix columns.

ANDROID Implementation of AES for encrypting and decrypting Text, Audio, Image and video files has shown following results:

Table 1 Performance of basic AES in ANDROID during encryption

Input Type	Execution time	CPU usage	RAM usage	Network usage
Text (25kb)	752 ms	23.33 %	9.35 MB	5.12 Kb/s
Image (245kb)	8425 ms	26.28 %	9.77 MB	30.91 Kb/s
Audio (733kb)	25562 ms	25.53 %	10.03 MB	53.57 Kb/s

Table 2 Performance of basic AES in ANDROID during decryption

Input Type	Execution time	CPU usage	RAM usage	Network usage
Text (25kb)	933 ms	24.49%	9.84 MB	7.09 Kb/s
Image (245kb)	7201 ms	25.38 %	9.90 MB	39.66 Kb/s
Audio (733kb)	15756 ms	28.47%	9.91 MB	84.04 Kb/s

Vidhyayana - ISSN 2454-8596
An International Multidisciplinary Peer-Reviewed E-Journal
www.vidhyayanaejournal.org
Indexed in: ROAD & Google Scholar

Observations:

Table 2.1and 2.2 clearly shows that as the amount of the input data to be encrypted increases, the execution time also increases exponentially. Similarly, it also shows that the execution time is also dependent on the type of input data.

Hence, we need to modify the basic AES algorithm to increase its efficiency by decreasing its execution time. For these modifications could be done in sbox/inverse sbox generation, mix columns and polynomial matrix generation part of basic AES algorithm.

During the implementation of the basic AES algorithm, we observed that the sbox and polynomial matrix was generated every single time when the algorithm was initialized. Also, the mix column operation performed a complex multiplication process every single time in each round. This was the reason for the increase in the execution time of the basic AES algorithm when size of input data was increased. Hence in order to reduce execution time we can remove the repetitive procedures in the algorithm without affecting its security. Next sections deal with the proposed modifications to achieve this.

Proposed work:

Modifications in AES initialization:

Every single time when the AES algorithm was initialized, it generated the same sbox & inverse sbox. The S-box is created by searching for the inverses of all elements of GF (28) and by applying affine transformations to all inverses. Finally, the inverse S-Box is constructed from sbox by sbox inversion. Instead of generating it every time, we used a fixed sbox/inverse sbox in the form of a 16 x 16 matrix.

Vidhyayana - ISSN 2454-8596
An International Multidisciplinary Peer-Reviewed E-Journal
www.vidhyayanaejournal.org
Indexed in: ROAD & Google Scholar

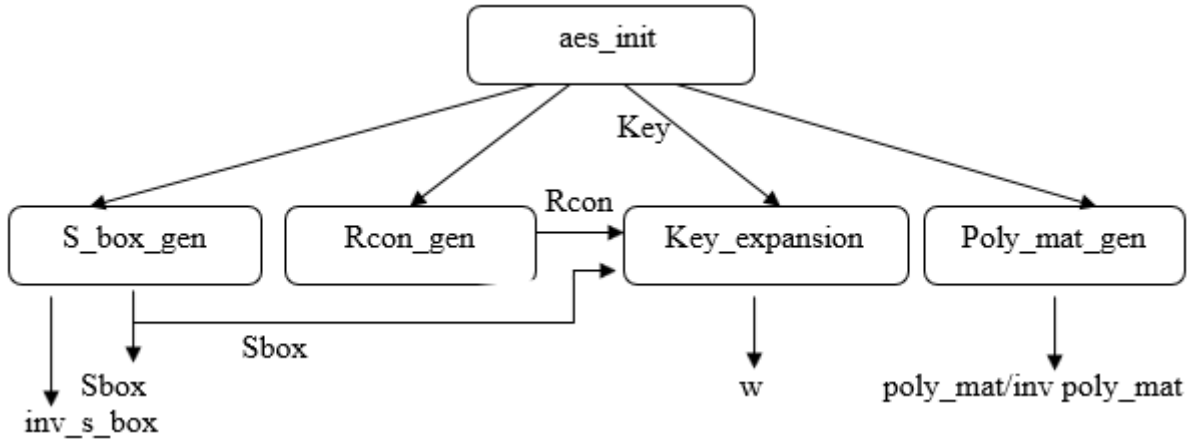

Fig. 3.1.1. Initialization process of basic AES algorithm [10]

From Fig 3.1.1 we can observe that sbox and polynomial matrices are generated each and every time the AES algorithm initializes. So we have replaced both this generation processes by fixed sbox and fixed polynomial matrix.

sbox [][]= {{99, 124, 119, 123, 242, 107, 111, 197, 48, 1, 103, 43, 254, 215, 171, 118},

{202, 130, 201, 125, 250, 89, 71, 240, 173, 212, 162, 175, 156, 164, 114, 192},

{183, 253, 147, 38, 54, 63, 247, 204, 52, 165, 229, 241, 113, 216, 49, 21},

{4, 199, 35, 195, 24, 150, 5, 154, 7, 18, 128, 226, 235, 39, 178, 117},

{9, 131, 44, 26, 27, 110, 90, 160, 82, 59, 214, 179, 41, 227, 47, 132},

{83, 209, 0, 237, 32, 252, 177, 91, 106, 203, 190, 57, 74, 76, 88, 207},

{208, 239, 170, 251, 67, 77, 51, 133, 69, 249, 2, 127, 80, 60, 159, 168},

{81, 163, 64, 143, 146, 157, 56, 245, 188, 182, 218, 33, 16, 255, 243, 210},

{205, 12, 19, 236, 95, 151, 68, 23, 196, 167, 126, 61, 100, 93, 25, 115},

{96, 19, 79, 220, 34, 42, 144, 136, 70, 238, 184, 20, 222, 94, 11, 219},

{224, 50, 58, 10, 73, 6, 36, 92, 194, 211, 172, 98, 145, 149, 228, 121},

Vidhyayana - ISSN 2454-8596
An International Multidisciplinary Peer-Reviewed E-Journal
www.vidhyayanaejournal.org
Indexed in: ROAD & Google Scholar

{231, 200, 55, 109, 141, 213, 78, 169, 108, 86, 244, 234, 101, 122, 174, 8},

{186, 120, 37, 46, 28, 166, 180, 198, 232, 221, 116, 31, 75, 189, 139, 138},

{112, 62, 181, 102, 72, 3, 246, 14, 97, 53, 87, 185, 134, 193, 29, 158},

{225, 248, 152, 17, 105, 217, 142, 148, 155, 30, 135, 233, 206, 85, 40, 223},

{140, 161, 137, 13, 191, 230, 66, 104, 65, 153, 45, 15, 176, 84, 187, 22}};

Fig. 3.1.2. Fixed 16x16 sbox matrix during AES initialization

poly_mat [] [] = {{2 ,3, 1, 1},

{1, 2, 3, 1},

{1, 1, 2, 3},

{3, 1, 1, 2}}

Fig. 3.1.3. Fixed 4x4 polynomial matrix during AES initialization

Similarly, a fixed inverse sbox and a fixed inverse polynomial matrix are used. Because of this a huge reduction in execution time can be seen.

Modifications in Mix columns:

Ciphering process is the other area where modification is done. In AES ciphering is done in terms of number of rounds. The number of rounds depends on the size of the ciphering key. The keys with 128-bit use 10 rounds, 192-bit use 12 rounds, and 256-bit use 14 rounds. Each round consists of four processes: -

1. Add round key

2. Substitute byte

3. Shift rows

4. Mix columns

Vidhyayana - ISSN 2454-8596
An International Multidisciplinary Peer-Reviewed E-Journal
www.vidhyayanaejournal.org
Indexed in: ROAD & Google Scholar

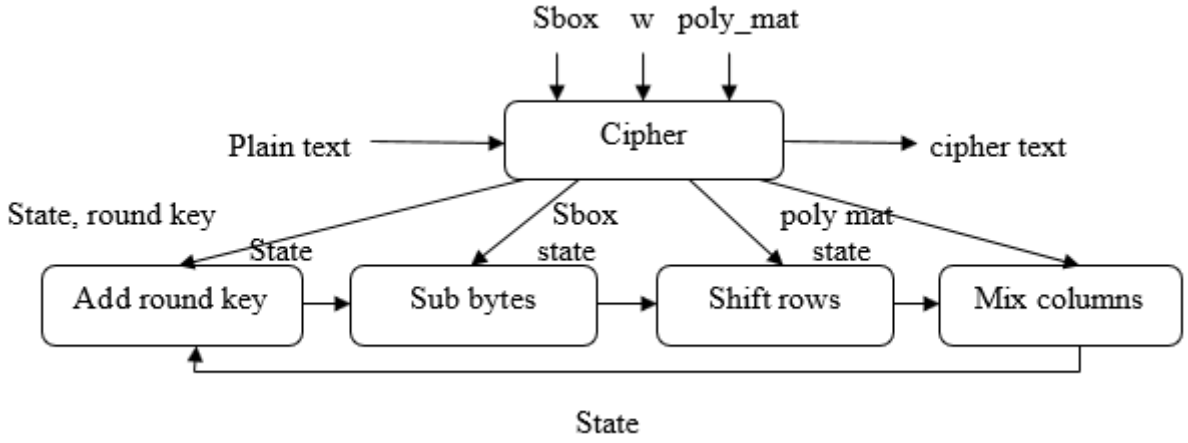

Fig. 3.2.1. Ciphering process of basic AES algorithm [10]

Mix columns consist of four stages i.e. addroundkey (), subbytes (), shiftrows (), mixcolumns () but in mixcolumns () the predefined polynomial matrix is multiplied with state matrix. In this complex multiplication process addition and shifting is done separately to each and every element. So in place of this complex process we suggest that a simple mapping process is enough. From fig 3.1.3 we can say that the values of all the elements of the polynomial matrix contain 1, 2 or 3. Hence the multiplication of any element of the state matrix is done only with any one of these three values. Any element multiplied with 1 produces the same value hence no process is required. So only two multiplication values are required i.e. with 2 & with 3.

The range of state matrices is 0 to 255. Hence just two lookup matrices are enough. One with all values 0 to 255 multiplied by 2 and another with 3. Hence we suggest two more fixed sboxes i.e. sbox2 and sbox3. Instead of multiplication by 2 or 3 we just do mapping from sbox2 or sbox3. Mapping is a highly efficient process.

In addition, we do not need function subbytes () as we are doing mapping of data in mixcolumns (). So all the data will be mapped to either sbox or sbox2 or sbox3 directly in mixcolumns () as shown in figure 3.2.2

Vidhyayana - ISSN 2454-8596
An International Multidisciplinary Peer-Reviewed E-Journal
www.vidhyayanaejournal.org
Indexed in: ROAD & Google Scholar

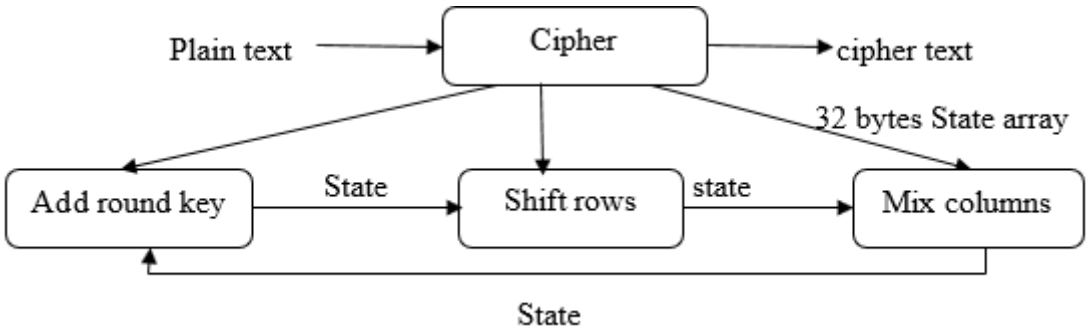

Fig. 3.2.2. Ciphering process of modified AES algorithm

```
sbox2[][]={{198, 248, 238, 246, 255, 214, 222, 145, 96, 2, 206, 86, 231, 181, 77, 236},
           {143, 31, 137, 250, 239, 178, 142, 251, 65, 179, 95, 69, 35, 83, 228, 155},
           {117, 225, 61, 76, 108, 126, 245, 131, 104, 81, 209, 249, 226, 171, 98, 42},
           {8, 149, 70, 157, 48, 55, 10, 47, 14, 36, 27, 223, 205, 78, 127, 234},
           {18, 29, 88, 52, 54, 220, 180, 91, 164, 118, 183, 125, 82, 221, 94, 19},
           {166,185, 0, 193, 64, 227, 121, 182, 212, 141, 103, 114, 148, 152, 176, 133},
           {187, 197, 79, 237, 134, 154, 102, 17, 138, 233, 4, 254, 160, 120, 37, 75},
           {162, 93, 128, 5, 63, 33, 112, 241, 99, 119, 175, 66, 32, 229, 253, 191},
           {129, 24, 38, 195, 190, 53, 136, 46, 147, 85, 252, 122, 200, 186, 50, 230},
           {192, 25, 158, 163, 68, 84, 59, 11, 140, 199, 107, 40, 167, 188, 22, 173},
           {219, 100, 116, 20, 146, 12, 72, 184, 159, 189, 67, 196, 57, 49, 211, 242},
           {213, 139, 110, 218, 1, 177, 156, 73, 216, 172, 243, 207, 202, 244, 71, 16},
           {111, 240, 74, 92, 56, 87, 115, 151, 203, 161, 232, 62, 150, 97, 13, 15},
           {224, 124, 113, 204, 144, 6, 247, 28, 194, 106, 174, 105, 23, 153, 58, 39},
```

Vidhyayana - ISSN 2454-8596
An International Multidisciplinary Peer-Reviewed E-Journal
www.vidhyayanaejournal.org
Indexed in: ROAD & Google Scholar

{217, 235, 43, 34, 210, 169, 7, 51, 45, 60, 21, 201, 135, 170, 80, 165},

{3, 89, 9, 26, 101, 215, 132, 208, 130, 41, 90, 30, 123, 168, 109, 44}};

sbox3[][]={{165, 132, 153, 141, 13, 189, 177, 84, 80, 3, 169, 125, 25, 98, 230, 154},

{69, 157, 64, 135, 21, 235, 201, 11, 236, 103, 253, 234, 191, 247, 150, 91},

{194, 28, 174, 106, 90, 65, 2, 79, 92, 244, 52, 8, 147, 115, 83, 63},

{12, 82, 101, 94, 40, 161, 15, 181, 9, 54, 155, 61, 38, 105, 205, 159},

{27, 158, 116, 46, 45, 178, 238, 251, 246, 77, 97, 206, 123, 62, 113, 151},

{245, 104, 0, 44, 96, 31, 200, 237, 190, 70, 217, 75, 222, 212, 232, 74},

{107, 42, 229, 22, 197, 215, 85, 148, 207, 16, 6, 129, 240, 68, 186, 227},

{243, 254, 192, 138, 173, 188, 72, 4, 223, 193, 117, 99, 48, 26, 14, 109},

{76, 20, 53, 47, 225, 162, 204, 57, 87, 242, 130, 71, 172, 231, 43, 149},

{160, 152, 209, 127, 102, 126, 171, 131, 202, 41, 211, 60, 121, 226, 29, 118},

{59, 86, 78, 30, 219, 10, 108, 228, 93, 110, 239, 166, 168, 164, 55, 139},

{50, 67, 89, 183, 140, 100, 210, 224, 180, 250, 7, 37, 175, 142, 233, 24},

{213, 136, 111, 114, 36, 241, 199, 81, 35, 124, 156, 33, 221, 220, 134, 133},

{144, 66, 196, 170, 216, 5, 1, 18, 163, 95, 249, 208, 145, 88, 39, 185},

{56, 19, 179, 51, 187, 112, 137, 167, 182, 34, 146, 32, 73, 255, 120, 122},

{143, 248, 128, 23, 218, 49, 198, 184, 195, 176, 119, 17, 203, 252, 214, 58}};

Fig. 3.2.3. Fixed sbox2 and sbox3

Vidhyayana - ISSN 2454-8596
An International Multidisciplinary Peer-Reviewed E-Journal
www.vidhyayanaejournal.org
Indexed in: ROAD & Google Scholar

Results:

Results of modified AES compared with basic AES are shown below.

Table 3 Comparison of basic AES and modified AES during encryption

INPUT TYPE	Execution Time		CPU Usage		RAM Usage		Network Usage	
	AES	**Modified AES**	**AES**	**Modified AES**	**AES**	**Modified AES**	**AES**	**Modified AES**
Text (25kb)	52 ms	23 ms	3.33 %	20.42 %	9.35 MB	15.46 MB	5.12 Kb/s	5.38 Kb/s
Image (245kb)	425 ms	120 ms	6.28 %	28.87 %	9.77 MB	9.61 MB	30.91 Kb/s	23.01 Kb/s
Audio (733kb)	5562 ms	358 ms	5.53 %	24.97 %	10.03 MB	9.36 MB	53.57 Kb/s	48.14 Kb/s

Table 4 Comparison of basic AES and modified AES during decryption

INPUT TYPE	Execution Time		CPU Usage		RAM Usage		Network Usage	
	AES	**Modified AES**	**AES**	**Modified AES**	**AES**	**Modified AES**	**AES**	**Modified AES**
Text (25kb)	933 ms	42 ms	24.49 %	23.91 %	9.84 MB	14.42 MB	7.09 Kb/s	5.51 Kb/s
Image (245kb)	7201 ms	473 ms	25.38 %	26.83 %	9.90 MB	10.62 MB	39.66 Kb/s	41.54 Kb/s
Audio (733kb)	15756 ms	158 ms	28.47 %	29.95 %	9.91 MB	10.31 MB	84.04 Kb/s	89.24 Kb/s

Vidhyayana - ISSN 2454-8596
An International Multidisciplinary Peer-Reviewed E-Journal
www.vidhyayanaejournal.org
Indexed in: ROAD & Google Scholar

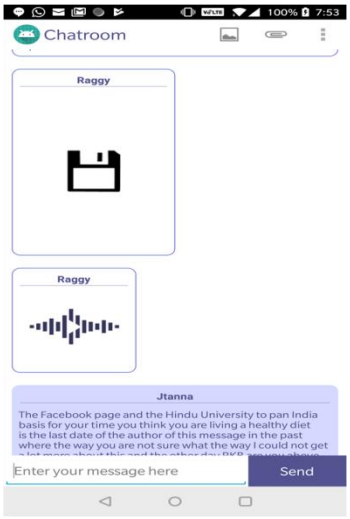

Fig 5.1 Implementation of basic and modified AES in Android chat application

We have compared basic and modified AES for Text, Image and Audio files. As shown in Table 4.1 and 4.2 we can say that in Android studio 3.0.1 there is huge improvement in execution time i.e., on an average 68.12 % in encryption, 72.96 % in decryption and overall on an average 70.24 %. Efficiency in terms of CPU usage, RAM usage and Network usage remains similar in case of basic as well as modified AES algorithm. It could also be observed that as the size of the input data was increased the improvement in execution time was also more.

Conclusion:

The AES algorithm has been successfully optimized. Efficiency in terms of execution time has been increased by replacing the process of sbox generation and polynomial matrix generation by predefined sbox and polynomial matrix. Also the complex multiplication process in the last stage of Mix columns during ciphering has been replaced by an efficient process of mapping. Results show that in Android studio 3.0.1 there is huge improvement in execution time i.e., on an average 68.12 % in encryption, 72.96 % in decryption and overall on an average 70.24 %. The only disadvantage of this method is that it requires larger memory to store 4 predefined sboxes and polynomial matrices in the form of arrays for mapping.

At last we have taken an Android chat application that can send and receive data in forms of text, image, audio and video and implemented basic AES algorithm and modified AES algorithm to encrypt and decrypt data before sending and after receiving respectively.

References:

[1]. Somaiya Riddhi C., Dr. Atul M. Gonsai, and Lect. Rashmin S. Tanna,"WLAN Security and Efficiency Issues based on Encryption Techniques" International Journal of Research in Engineering, IT and Social Sciences, ISSN 2250-0588, Impact Factor: 6.452, Volume 06 Issue 09, September 2016.

[2].Gurkas, G. Zeynep, et al. "Security mechanisms and their performance impacts on wireless local area networks." Computer Networks, 2006 International Symposium on. IEEE, 2006.

[3]. Zeghid Medien, et al. "A modified AES based algorithm for image encryption." International Journal of Computer Science and Engineering 1.1 (2007): 70-75.

[4].Pimpale Priyanka, et al. ``Modification to AES Algorithm for Complex Encryption." IJCSNS International Journal of Computer Science and Network Security (2011): 183-186.

[5].Rizky Riyaldhi et al. "Improvement of advanced encryption standard algorithm with shift row and s.Box modification mapping in mix column" 2nd International Conference on Computer Science and Computational Intelligence 2017, ICCSCI 2017, 13-14 October 2017, Bali, Indonesia

[6].Tanna Rashmin, Riddhi Tanna, and Jaykumar Bhadeshiya. "Improvement in the Execution Time of AES Algorithm by Modifications in Sbox and Mix Columns for Multimedia Applications." International Journal of Research in Engineering, IT and Social Sciences, ISSN 2250-0588, Impact Factor: 6.452, Volume 08 Issue 04, April 2018.

[7].Ahmad, N., Hasan, R., Jubadi, W.M. Design of aes s-box using combinational logic optimization. In: Industrial Electronics & Applications (ISIEA), 2010 IEEE Symposium on. IEEE; 2010, p. 696–699.

[8].Dehbaoui, A., Dutertre, J.M., Robisson, B., Tria, A. Electromagnetic transient faults injection on a hardware and a software implementations of aes in Fault Diagnosis and Tolerance in Cryptography (FDTC), 2012 Workshop on. IEEE; 2012, p. 7–15.

[9].Bai, R., Liu, H., Zhang, X. Aes and its software implementation based on arm920t. Journal of Computer Applications 2011; 31(5):1295– 1301.

[9].Federal Information Processing Standards Publication 197 November 26, 2001 announcing the "ADVANCED ENCRYPTION STANDARD (AES)".