May-June 2020
ISSN: 0193-4120 Page No. 3694-3699

Data Comparison among Encrypted Data using AES and Two Fish Algorithm on Cloud Computing Security

SP.Chokkalingam[1], E. Srimathi[2]
[1]Professor, Department of CSE, Saveetha School of Engineering,
Saveetha Institute of Medical and Technical Sciences, cho_mas@yahoo.com
[2]Research Scholar, Saveetha School of Engineering,
Saveetha Institute of Medical and Technical Sciences, srimamca4@gmail.com

Article Info
Volume 83
Page Number: 3694-3699
Publication Issue:
May-June 2020

Article History
Article Received: *19 August 2019*
Revised: *27 November 2019*
Accepted: *29 January 2020*
Publication: *12 May 2020*

Abstract
Now a day's cloud security is an important aspects in cloud computing. The cloud usage among the public user increases rapidly. Due to security risk, data reliability and data privacy is lost among the cloud user. In this research we compared two different security algorithms such as AES and TwoFish algorithm with different parameter techniques. The five different parameters which are used to compare these algorithms such as Key length, Key strength, Time taken for Encryption, Encrypting quality by applying different attack such as (Partial attack, Dictionary attack, Brute Force attack, Time taken for Decryption). These two algorithms are most probably used among organization and users on promising manner and this entire algorithm can be worked in 256 bit length. By knowing the result of this parameters we can identify which algorithm is best for security.

Keywords: *Advanced Encryption Standards (AES) and TwoFish.*

1. Introduction

Cloud computing security alludes to the controls that must be actualized so as to counteract the loss of information, data or assets having a place with a cloud administrations supplier or its clients. Cloud arrangement suppliers must execute an assortment of security controls for the SaaS (programming as assistance) solutions, PaaS (stage as a help) arrangements and the IaaS (framework as a help) arrangements that they give to their clients or customers.

Similarly as with conventional PC or information security, there are numerous bases that must be secured, in order to guarantee a protected figuring condition. PC security (counting distributed computing security)can be actualized by taking the accompanying safety efforts (as fitting): limiting access to applications and framework assets, logging access and utilization of uses and frameworks; and controlling and observing access to physical processing assets like servers and server farms, and so on.

A significant number of the security techniques that cloud specialist co-ops actualize are driven by protection, consistence or legitimate principles for overseeing purchaser information that particular enterprises must stick to. In any occasion, security breaches can bring about major money related misfortunes for any association, as exorbitant fines and lawful costs might be brought about; nearby lost trust in the association's capacity to ensure client information, which can prompt further misfortunes in business income.

1.1 Advanced Encryption Standard (AES)

AES is a symmetric block chipper based encryption; it uses 128-bits (10 rounds encryption), 192-bits (12 rounds encryption) and 256-bits (14 rounds encryption)

1.1.1 Encryption with AES

The encryption period of AES can be broken into three stages: the underlying round, the principle adjusts, and the last round. The entirety of the stages utilizes similar sub-tasks in various blends as pursues:

May-June 2020
ISSN: 0193-4120 Page No. 3694-3699

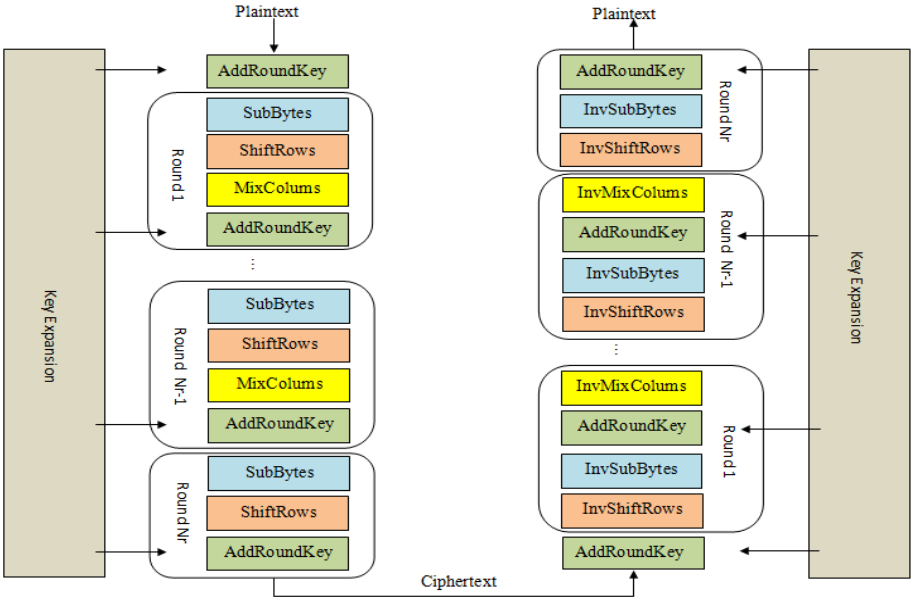


Figure 1: AES Encryption Process

i. Initial Round: AddRoundKey
ii. Main Rounds: SubBytes, ShiftRows, MixColumns, AddRoundKey.
iii. Final Round: SubBytes, ShiftRows, AddRoundKey
AddRoundKey: The AddRoundKey activity is the main period of AES encryption that straightforwardly works on the AES round key. In this activity, the contribution to the round is select order with the round key.

SubBytes: The SubBytes period of AES includes parting the contribution to bytes and going each through a Substitution Box or S-Box. In contrast to DES, AES utilizes a similar S-Box for all bytes. The AES S-Box executes opposite increase in Galois Field 28. The AES S-Box is appeared in the Table below.

Table 1: Input Bytes of AES Substitution Box

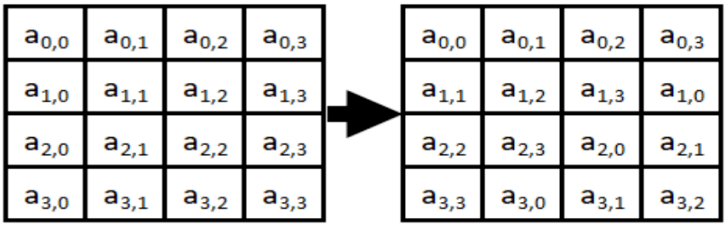
	0	1	2	3	4	5	6	7	8	9	0a	0b	0c	0d	0e	0f
0	63	7c	77	7b	f2	6b	6f	c5	30	1	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	4	c7	23	c3	18	96	5	9a	7	12	80	e2	eb	27	b2	75
40	9	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	0	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	2	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	6	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	8
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	3	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

To peruse this Table, the byte input is broken into two 4-piece parts. The principal half decides the line and the subsequent half decides the segment.
ShiftRows: In the ShiftRows period of AES, each line of the 128-piece inside condition of the figure is moved. The lines in this stage allude to the standard portrayal of the inward state in AES, which is a 4x4 network where every cell contains a byte. Bytes of the interior state are put in the grid crosswise over lines from left to right and down segments.

In the ShiftRows activity, every one of these columns is moved to one side by a set sum: their line number beginning with zero. The top line isn't moved in any way, the following column is moved by one, etc. This is shown in the Figure bellow.

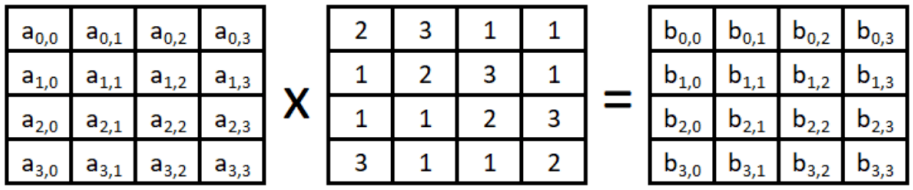
May-June 2020
ISSN: 0193-4120 Page No. 3694-3699

Table 2: Shifting the Rows and Columns



MixColumns: Like the ShiftRows period of AES, the MixColumns stage gives dissemination by blending the contribution around. Dissimilar to ShiftRows, MixColumns performs tasks parting the grid by segments rather than columns.

Table 3: Mixing the Rows and Columns



1.1.2 Decryption with AES

To unscramble an AES-encoded ciphertext, it is important to fix each phase of the encryption activity in the switch request in which they were applied. The three phase of unscrambling are as per the following:

i. Inverse Final Round: AddRoundKey, ShiftRows, SubBytes

ii. Inverse Main Round: AddRoundKey, MixColumns, ShiftRows, SubBytes

iii. Inverse Initial Round: AddRoundKey

The four tasks in AES encryption, just the AddRoundKey activity is its own opposite (since it is a selective or). To fix AddRoundKey, it is just important to extend the whole AES key calendar (indistinguishably from encryption) and afterward utilize the suitable key in the select or.

The other three activities require a reverse activity to be characterized and utilized. The primary activity to be fixed is ShiftRows. The Inverse ShiftRows activity is indistinguishable from the ShiftRows activity with the exception of that pivots are made to one side rather than to one side.

Table 4: Subbytes Operation in S-Box Matrix.

	1	2	3	4	5	6	7	8	9	0a	0b	0c	0d	0e	0f	
0	52	9	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
10	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
20	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
30	8	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
40	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
50	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
60	90	d8	ab	0	8c	bc	d3	0a	f7	e4	58	5	b8	b3	45	6
70	d0	2c	1e	8f	ca	3f	0f	2	c1	af	bd	3	1	13	8a	6b
80	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
90	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a0	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b0	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c0	1f	dd	a8	33	88	7	c7	31	b1	12	10	59	27	80	ec	5f
d0	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e0	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f0	17	2b	4	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

The last inverse operation is done on MixColumns in inverse format in Galois Field 2^8 as follows:

May-June 2020
ISSN: 0193-4120 Page No. 3694-3699

Table 5: MixColumns Inverse Operation.

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

X

14	11	13	9
9	14	11	13
13	9	14	11
11	13	9	14

=

$b_{0,0}$	$b_{0,1}$	$b_{0,2}$	$b_{0,3}$
$b_{1,0}$	$b_{1,1}$	$b_{1,2}$	$b_{1,3}$
$b_{2,0}$	$b_{2,1}$	$b_{2,2}$	$b_{2,3}$
$b_{3,0}$	$b_{3,1}$	$b_{3,2}$	$b_{3,3}$

1.4 TwoFish

Encryption and Decryption with TwoFish:

Twofish is a symmetric square figure; a solitary key is utilized for encryption and decoding. Twofish has a square size of 128 bits, and acknowledges a key of any length up to 256 bits. (NIST required the calculation to acknowledge 128-, 192-, and 256-piece keys.) Twofish is quick on both 32-piece and 8-piece CPUs (keen cards, implanted chips, and so forth), and in equipment. What's more, it's adaptable; it tends to be utilized in organize applications where keys are changed as often as possible and in applications where there is practically zero RAM and ROM accessible.

Twofish uses square figuring. Twofish uses a solitary key of any length up to 256 bits and is said to be effective both for programming that runs in littler processors, for example, those in shrewd cards and for implanting in equipment. It enables implementers to exchange off encryption speed, key arrangement time, and code size to adjust execution. Planned by Bruce Schneier's Counterpane Systems, Twofish is unpatented, permit free, and uninhibitedly accessible for use

Twofish is quick on both 32-piece and 8-piece CPUs (shrewd cards, installed chips, and so forth), and in equipment. Also, it's adaptable; it very well may be utilized in arrange applications where keys are changed every now and again and in applications where there is next to zero RAM and ROM accessible.

.

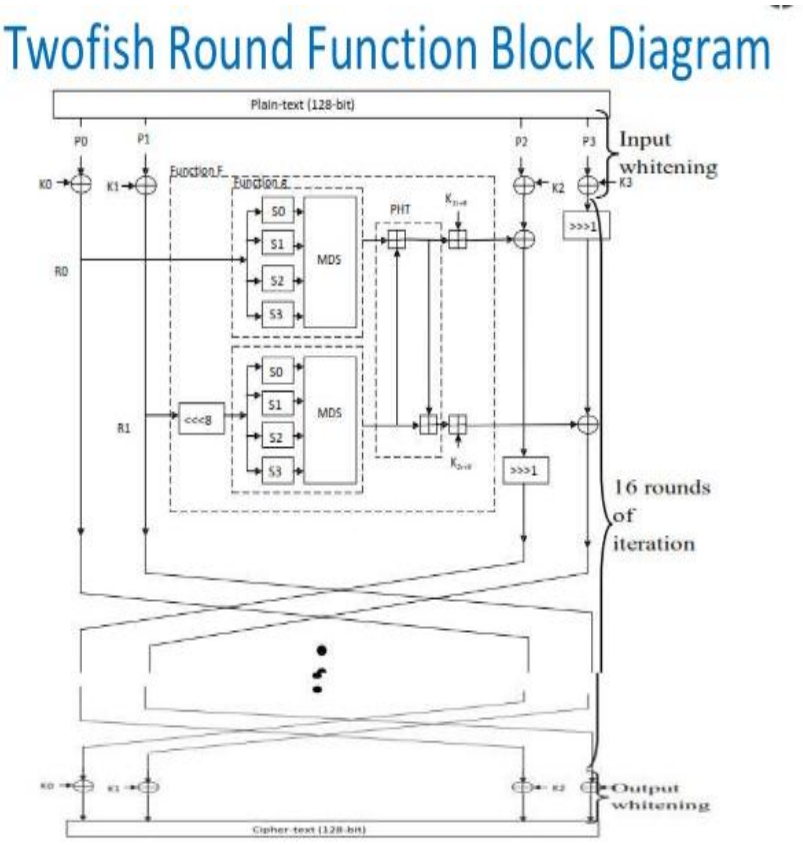


Figure 2: TwoFish Operation

TEST Engineering & Management

May-June 2020
ISSN: 0193-4120 Page No. 3694-3699

Twofish shouts on top of the line CPUs, and it's adaptable enough for minor keen card CPUs. It additionally functions admirably in equipment. What's more, there are a few presentation exchange offs between key-arrangement time and encryption speed that make it interesting among the AES competitors.

No other calculation has a similar adaptability in execution: the capacity to exchange off key-arrangement time for encryption speed, and ROM and RAM for encryption speed. These alternatives exist on 32-piece CPUs, 8-piece CPUs, and equipment.

What's more, Twofish does this with a moderate plan. We decided not to adjust the fundamental Feistel arrange. We didn't utilize information subordinate revolutions, 32-piece increases, or some other ineffectively got natives. The key timetable is intended to oppose even the nastiest of assaults. What's more, we gave the figure 16 rounds when we could just break five.

2. Module Description

The Research module technique compares Symmetric key algorithm Advanced Encryption Standard (AES) and TwoFish. These two algorithms are most probably used among organization and users on promising manner and this entire algorithm can be worked in 256 bit length.
The four different parameters are used to compare these algorithms, the parameter constraints are:
1. Key length: This is used to know the key length of the encrypted document
2. Key strength: 128/192/256 bit combination used in encryption algorithm
3. Time taken for Encryption
4. Time taken for Decryption
By knowing the result of this parameters we can identify which algorithm is best for security for encryption among cloud security.

Implementation code

```
        // Get file content and key for encrypt/decrypt
        try
        {
            byte[] fileContent = File.ReadAllBytes(tbPath.Text);
            byte[] passwordTmp = Encoding.ASCII.GetBytes(tbPassword.Text);
            byte[] keys = new byte[fileContent.Length];
            for (int i = 0; i < fileContent.Length; i++)
                keys[i] = passwordTmp[i % passwordTmp.Length];

            // Encrypt
            byte[] result = new byte[fileContent.Length];
            //string fileName = @"C:\Users\RAJESH\Desktop\trail.docx";
            //FileInfo fi = new FileInfo(fileName);
            long size = fileContent.Length;
            string s = ConvertBytesToMegabytes(fileContent.Length).ToString("0.00");
            label4.Text = "File Size in Bytes: " + size ;
            label5.Text = "File Size in MB: " + s;
            label7.Text = "256 Bit Length Key is Used";
            Time.Text = "Time Taken: 00:00:04";
            if (rbEncrypt.Checked)
            {
                for (int i = 0; i < fileContent.Length; i++)
                {
                    byte value = fileContent[i];
                    byte key = keys[i];
                    int valueIndex = -1, keyIndex = -1;
                    for (int j = 0; j < 256; j++)
                        if (abc[j] == value)
                        {
                            valueIndex = j;
                            break;
                        }
                    for (int j = 0; j < 256; j++)
                        if (abc[j] == key)
                        {
                            keyIndex = j;
                            break;
                        }
                    result[i] = table[keyIndex, valueIndex];
                }
            }
            // Decrypt
            else
            {
                label4.Text = "File Size in Bytes: " + fileContent.Length;
                label5.Text = "File Size in MB: " + ConvertBytesToMegabytes(fileContent.Length).ToString("0.00");
                label7.Text = "256 Bit Length Key is Used";
                Time.Text = "Time Taken: 00:00:04";
                for (int i = 0; i < fileContent.Length; i++)
                {
                    byte value = fileContent[i];
                    byte key = keys[i];
                    int valueIndex = -1, keyIndex = -1;
                    for (int j = 0; j < 256; j++)
                        if (abc[j] == key)
                        {
                            keyIndex = j;
                            break;
                        }
                    for (int j = 0; j < 256; j++)
                        if (table[keyIndex, j] == value)
                        {
                            valueIndex = j;
                            break;
                        }
                    result[i] = abc[valueIndex];
                }
            }
```

May-June 2020
ISSN: 0193-4120 Page No. 3694-3699

3. Implementation Design

PLAIN TEXT **ENCRYPTED TEXT** **DECRYPTED TEXT**

4. Conclusion

AES and TwoFish algorithm are most trusted encryption algorithm among user and organization, both the algorithm uses symmetric block chipper based encryption technique with various key size such as 128/192/256 bit. It is very difficult to hack the details from this encryption. For 256-bit key size it will take 3.31 x 1056 years time to crack the details of encrypted file. Both the algorithm are best for symmetric encryption but time taken for encryption differs. Thus the TwoFish encryption algorithm is faster than AES encryption.

References

[1] H. Takabi, J. B. Joshi, and G.-J. Ahn, "Security and privacy challenges in cloud computing environments," IEEE Security & Privacy, no. 6, pp. 24–31, 2010

[2] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," IEEE Internet Computing, no. 1, pp. 69–73, 2012

[3] Wen, X., et al, Comparison of open-source cloud management platforms: "OpenStack and OpenNebula. Fuzzy Systems and Knowledge Discovery (FSKD)," 2012 9th International Conference on. IEEE

[4] E.Srimathi and Dr. SP.Chokkalingam, "OpenKey-Generation for Enabling Cloud Storage Security in Open Source Cloud Computing",JARDCS,Vol.9.Sp-17/2017.

[5] Yang Luo, Wu Luo, Tian Puyang, Qingni Shen, Anbang Ruan†, Zhonghai Wu, "OpenStack Security Modules: a Least-Invasive Access Control Framework for the Cloud,"2016 IEEE 9th International Conference on Cloud Computing

[6] E.Srimathi and Dr. SP.Chokkalingam "Securing Open Source Cloud Storage on OPenStack Cloud Computing Platform",IJRTE,Vol-7,April/2019.

[7] Dr. SP.Chokkalingam and E.Srimathi "Efficiency on Public Cloud Storage Providers in Cloud Computing", IJRTE,Vol-7,April/2019.

[8] Suryadipta Majumdar, Taous Madi, Yushun Wang, Yosr Jarraya, Makan Pourzandi, Lingyu Wang and Mourad Debbabi, "Security Compliance Auditing of Identity and Access

[9] Secured Data Communication in Cloud Computing using Channel API with MD5 Hashing, ISSN: 2320-1363, journal of International Journal of Merging Technology and Advanced Research in Computing

[10] Management in the Cloud: Application to OpenStack," 2015 IEEE 7th International Conference on Cloud Computing Technology and Science