International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 04 | Apr 2020 www.irjet.net p-ISSN: 2395-0072

Data Security Strategy based on Multi-Level File Encryption for Cloud Computing

Vivek Kumar Ratan[1], Dr. Bharathi[2]

[1]Department of CSE, SRM Institute of Science and Technology, Vadapalani, Chennai
[2]Asst. Professor, Department of CSE, SRM Institute of Science and Technology, Vadapalani, Chennai

---

***Abstract* —** In today's modern world where everything is based on the internet, a huge amount of data is getting generated everyday. Cloud computing has been envisioned as the next generation paradigm in computation. Data security and privacy protection are the two main factors of user's concerns about the cloud computing. In the cloud computing environment, both applications and resources are delivered on demand over the Internet as services. Cloud is an environment of the hardware and software resources in the data centers that provide diverse services over the network or the Internet to satisfy user's requirements. The idea is to implementation of an algorithm which can help in protecting the user's data by encryption and decryption. This algorithm is known as Artificial Immune Algorithm. The encryption and decryption keys will be provided to the user. User can authorize others who want to access its file by simply providing the decryption key to other people. This will help in protecting the user's data against hackers, unauthorized access of data.

I. INTRODUCTION

Cloud computing has been envisioned as the next generation paradigm in computation. In the cloud computing environment, both applications and resources are delivered on demand over the Internet as services. The main motive of cloud computing is to provide convenience of uploading and accessing the file over the internet without worrying about the vulnerabilities such as data theft, unauthorized access, etc. Data security and privacy protection issues are relevant to both hardware and software in the cloud architecture.

II. RELATED WORK

In cloud computing, when it comes to cloud data protection methods, no particularly new technique is required. Protecting data in the cloud can be similar to protecting data within a traditional data center. The problem arises in selecting, adopting and implementing the various encryption techniques required to protect the data of user. These techniques helps in determining on how much safe is user's data on the cloud. Some techniques are easier to implement which makes it easy for the developer to integrate while designing the cloud architecture. But it comes at a cost as they are not secure and efficient. Maintaining confidentiality, integrity, and availability for data security is a function of the correct application and configuration of familiar network, system, and application security mechanisms at various levels in the cloud infrastructure. The techniques which are more secure are hard to implement and takes a lot of time. Therefore we came up with an algorithm known as Artificial Immune Algorithm. This algorithm is easy to implement and more efficient than other techniques.

PROPOSED SYSTEM

To overcome these problems with the existing system, we came with an idea of implementation of algorithm which will help in storing data in cloud with the help of artificial immune algorithm to make the system more efficient.

(A) Users:

In our proposed system, there are two types of users:

1. Uploader: The person who has the file and upload that file in the cloud system.

2. Requestor: The person who wants to access the file uploaded by the uploader.

In our system, when the file gets uploaded in the cloud, the requestor can see the file listed in the portal. Requestor can send the request to the uploader for accessing the file. The uploader can provide keys to decrypt the encrypted file.

(B) Files:

- The supported file type in our proposed system is ".txt ".
- Maximum size of file for uploading is 16MB.

(C) File splitting:

The file which is uploaded will be divided into three parts. The division of the file will be based on the length the file. The file stores the data in the form of bits which will determine the length of a file uploaded in the cloud. The system will regain the authenticity of the document to avoid the errors in the uploaded file.

(D) Encryption:

After the file split is done, the next action in queue is encryption where there are three parts of the file now to get encrypted. For the system to not get too predictable, we have come up with some set of options for the uploader to choose from.

- One way would be to opt a standard set of encryption algorithms, i.e., the split parts will sequentially get encrypted through encrypting algorithms in the order of AES(128 bit), DES and SHA-1.

- The other method of encrypting the file would be *randomization*. In this method, the system will randomly deploy the encryption algorithms in such a manner that everytime file would be encrypted using different combinations of algorithms. It will improve the security of the cloud system.

The encryption algorithms used in the system are:

1. AES:

AES or Advanced Encryption Standard is a cipher, i.e., a method for encrypting and decrypting information. Whenever files are transmitted over secure file transfer protocols like HTTPS, FTPS, SFTP, WebDAVS, OFTP, or AS2, there's a good that data will be encrypted by some flavor of AES ciphers - either AES 256, 192, or 128.

AES Functions

AES has four functions: SubBytes, ShiftRows, MixColumns, and AddRoundKey. These functions provide confusion, diffusion, and XOR encryption to the State.

1.1 SubBytes

The SubBytes function provides confusion by substituting the bytes of the State. The bytes are substituted according to a substitution table (also called an S-Box).

To use the table, take the byte of the State to be substituted (assume the byte is the letter "T"). ASCII "T" is hexadecimal byte "53." Look up 5 on the X row and 3 on the Y column, resulting in hexadecimal byte "ed;" this replaces "53" in the State.

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Fig. The AES substitution table directly from FIPS-197, with the byte 53 lookup overlaid on top

1.2 ShiftRows

ShiftRows provides diffusion by shifting rows of the State. It treats each row like a row of blocks, shifting each a different amount:

• Row 0 is unchanged

• Row 1 is shifted 1 to the left

• Row 2 is shifted 2 to the left

• Row 3 is shifted 3 to the left.

1.3 MixColumns

MixColumns also provides diffusion by "mixing" the columns of the State via finite field mathematics

1.4 AddRoundKey

AddRoundKey is the final function applied in each round. It XORs the State with the subkey. The subkey is derived from the key, and is different for each round of AES.

2. *DES:*

The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).

DES is an implementation of a Feistel Cipher. It uses 16 round Feistel structure. The block size is 64-bit. Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only).

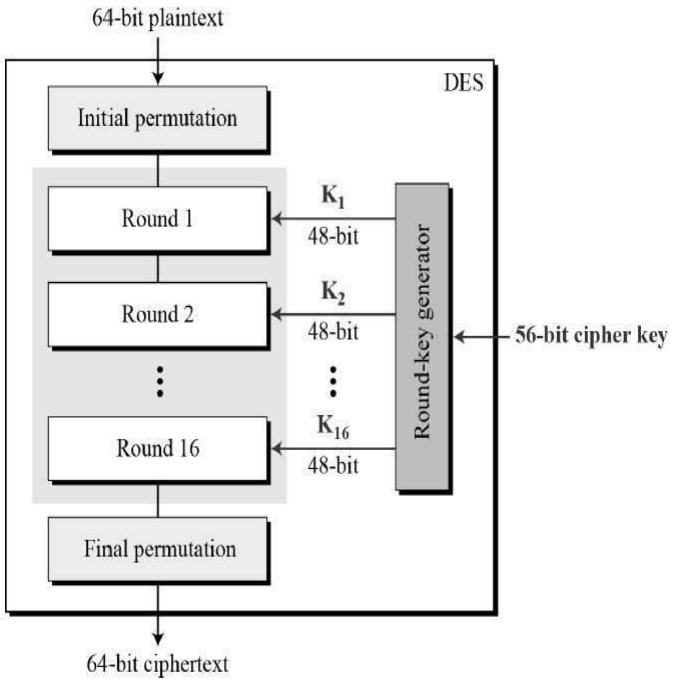
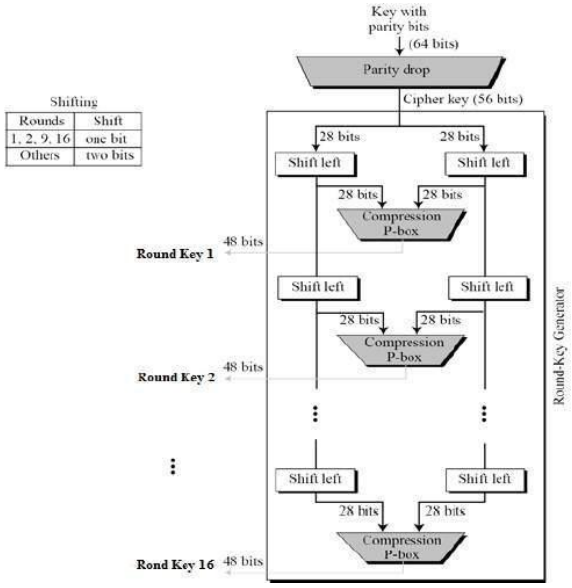


Fig. General Structure of DES

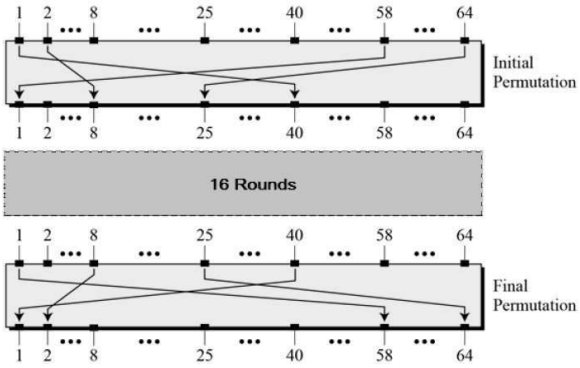
Since DES is based on the Feistel Cipher, all that is required to specify DES is –

- Initial and final permutation
- Round function
- Key Generation



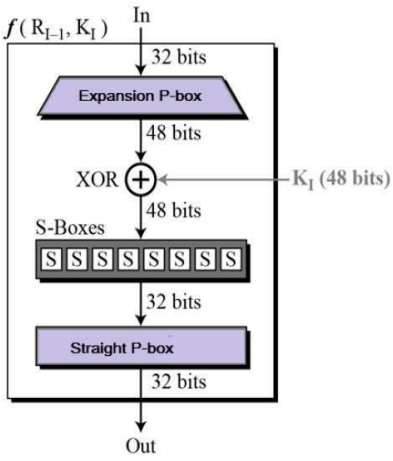
Initial and Final Permutation

The initial and final permutations are straight Permutation boxes (P-boxes) that are inverses of each other. They have no cryptography significance in DES. The initial and final permutations are shown as follows –



Round Function

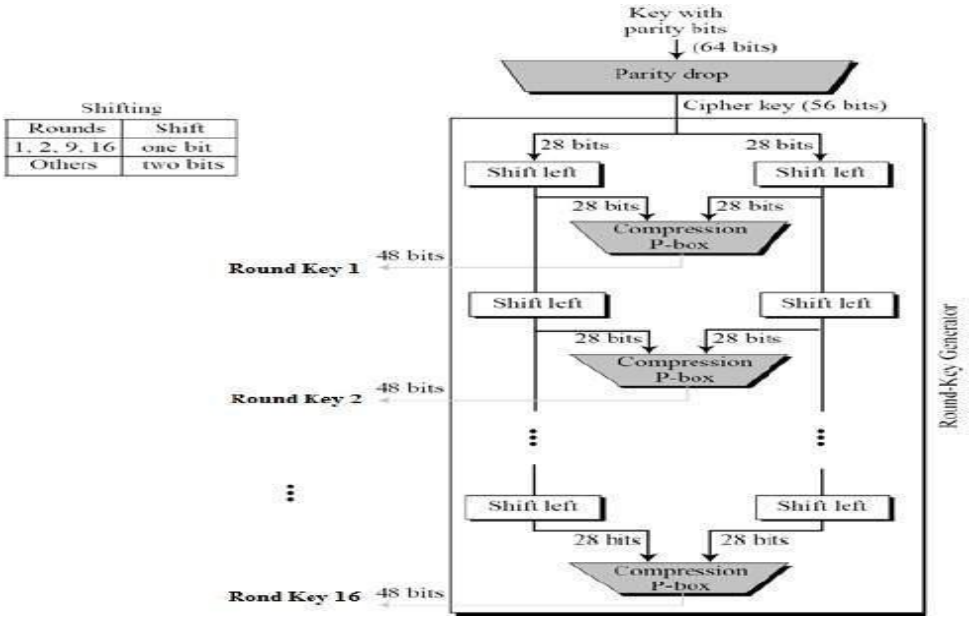
The heart of this cipher is the DES function, f. The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output.



Key Generation

The round-key generator creates sixteen 48-bit keys out of a 56-bit cipher key. The process of key generation is depicted in the following illustration –

DES Analysis



The DES satisfies both the desired properties of block cipher. These two properties make cipher very strong.

- Avalanche effect – A small change in plaintext results in the very great change in the ciphertext.
- Completeness – Each bit of ciphertext depends on many bits of plaintext.

During the last few years, cryptanalysis have found some weaknesses in DES when key selected are weak keys. These keys shall be avoided.

3. BLOWFISH:

Blowfish is a symmetric-key block cipher, designed in 1993 by Bruce Schneier and included in many cipher suites and encryption products. Blowfish provides a good encryption rate in softwares. It was designed as general purpose algorithm. It is unpatented and can be freely used by anyone.

Blowfish is a 16-round Feistel cipher. It's block size is 64-bit and key sizes range from 32 to 448 bits. Encryption with Blowfish has two main stages: sixteen iterations of the round function and an output operation.

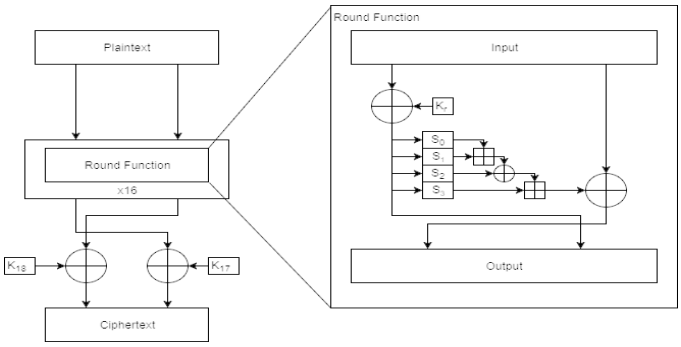


Fig. The Feistel structure of Blowfish Encryption

Blowfish round function:

The round function in Blowfish encryption has four stages (see diagram above):

1. Key whitening of the left side of the input with the rth round key

2. Application of the S-Boxes and combination of their results

3. Exclusive-or of the right side of the input with the output of the F function (key whitening, S-Boxes and combination of S-Box output).

4. Swapping the sides of the output.

In the key-whitening stage, the left side of the input is exclusive-ored with the round key for the given round. The S-Boxes are set as part of the key generation algorithm. The output of an S-Box for an input of n is the nth value in the S-Box.

The outputs of the S-Boxes are combined through a mixture of addition and exclusive-or. The outputs of the first two S-Boxes are added together modulo 232.. More formally, the result, R, of applying this sequence to input, I, is reached through the following equation (where a[0:5] refers to the first 5 bits of a)

A = S0(I[0:8]) + S1(I[8:16]) mod 2**32

B = A ^ S2(I[16:24]))

R = B + S3(I[24:32]) mod 2**32

Like other Feistel functions, the output of this is exclusive-ored with the other side of the input (the right side in this case) and the two sides of the input are swapped before entering the next round.

Blowfish output function

The final stage of the Blowfish cipher involves two steps: reversing the final swap and performing output whitening. In output whitening, the right side of the output (after being swapped) is exclusive-ored with the seventeenth round key and the left side is exclusive-ored with the eighteenth round key. The result of this is the Blowfish ciphertext.

Blowfish Key Schedule (and S-box generation)

Initial values

The Blowfish key schedule relies heavily on the Blowfish encryption algorithm described in the previous section. The key schedule uses a value, P, consisting of eighteen words which contain (in order) the first eighteen words of the binary representation of the fractional part of pi. For example, the hexadecimal representation of pi begins with 3.243F6A8885A308D313198A2E037073, therefore P1=0x243F6A88, P2=0x85A308D3, etc. This value, P, will become the round keys used in encryption.

Next, set the initial values of the S-Boxes in the same manner beginning with the 19th word of the fractional part of pi. The ordering should be that the entire first S-Box is filled in order before moving on to the next and so on.

Since P contains 18 words and the S-Boxes each contain 256 words, a total of 18 + 4*256 = 1042 pi words are used, each 32-bit in size.

Generating Round Keys and S-box:

Generation of the round key is performed in rounds where each round generates two round key values. The process is as follows:

1. Initialize P and S-Boxes as described above
2. Exclusive-or P1 with the first 32 key bits, P2 with the next 32 bits and so on until all of the key has been exclusive-ored (since the key is shorter than P, parts of it will be used multiple times to cover all of P)
3. Set the initial input to zero
4. Encrypt the input using the current version of P as the round keys
5. Set the first two unreplaced values of P to the value of the ciphertext from step 4
6. Set the input to the ciphertext from step 4
7. Repeat steps 4 through 6 until all of P has been replaced
8. Use the resulting value of P as the round keys in encryption
9. Repeat steps 4 through 6, replacing values of the S-Boxes two at a time until all S-Box values have been replaced. Since P contains 18 words and the S-Boxes each contain 256 words, there is a total of 18 + 4*256 = 1042 values to replace, which will take 521 iterations of steps 4 through 6 of the above algorithm to complete.

(E) Decryption:

While encrypting, the keys produced are safely stored in the database mapped with their respective algorithms. The keys differ in size as the algorithms and key size are different for each of them. Once the uploader is ready to share the keys, the requester can get the private keys to decrypt the parts. Once the decryption is completed, the merge functionality will help to merge the split files.

(H) Merging:

Now the secondary user can view the entire file in front of him in three different parts, these parts must be merged in order to form a relevant file.

For this part to be accomplished, a simple file operation must be performed on the pre-existing decrypted parts. The file operation helps merge these parts of text into a single text file and lets the secondary user download this file.

This helps provided a bridge between the primary user and users who wishes to attain the files uploaded by him.

(I) File Integrity:

Hashing algorithms such as SHA-1 could be used to hash the file content before uploading. So that, before providing the access of the encrypted file to the secondary user, the final merged content could be hashed again to compare the values. If the hash matches, then the file can be termed as genuine.

IV. CONCLUSION AND FUTURE SCOPE

The future of cloud computing will most likely represent a combination of cloud-based software products and on premises compute to create a hybrid IT solution that balances the scalability and flexibility associated with cloud and the security and control of a private data center. In the current cloud market the benefits of leveraging the infrastructure of a large cloud provider can be beneficial in many ways. The cost structure works like a utility which provides for an operating expense model with no upfront infrastructure costs.

REFERENCES

1. Shah, M. A., Swaminathan, R., Baker, M. Privacy-preserving audit and extraction of digital contents IACR Cryptology EPrint Archive 2008 186.

2. Xiao, Z., Xiao, Y. Security and privacy in cloud computing IEEE Communications Surveys & Tutorials 2013 15 2 843 859 10.1109/SURV.2012.060912.00182 2-s2.0-84877272118.

3. Jean-Daniel Cryans, Criteria to Compare Cloud Computing with Current Database Technology (2008), 23-26.

4. Jean-Daniel Cryans, Criteria to Compare Cloud Computing with Current Database Technology (2008), 23-26.

5. Dai Yuefa,Wu Bo, Gu Yaqiang, Zhang Quan, Tang Chaojing, “Data Security Model for Cloud Computing”, Proceedings of the 2009 InternationalWorkshop on Information Security and Application (IWISA 2009), 141-144.

6. Mahmood, Z. Data location and security issues in cloud computing Proceedings of the 2nd International Conference on Emerging Intelligent Data and Web Technologies (EIDWT '11) September 2011 IEEE 49 54 10.1109/EIDWT.2011.16 2-s2.0-83055196683