JOURNAL OF MULTIDIMENSIONAL RESEARCH & REVIEW

Website: http://www.jmrr.org **Vol.2, Iss.2 Pages:76-84**

CLOUD SECURITY BY FORMAT PRESERVING ENCRYPTION

[1]Dr S Vidhya, [2]Dr P Ajitha

[1,2] Assistant Profesor, [1]Dept of Information Technology, [2]Department of Software Systems, KG College of Arts and Science, Coimbatore, Tamilnadu, India – 641043

Article Received: April 2021 Published: July 2021

Abstract

In recent years, cloud computing is a widely used technology. Cloud computing provides service to the user over an internet. Cloud computing means storing and accessing programs and data on internet instead of our computer memory. A several number of data types are used in cloud computing such as audio, video, decimal, image, date, Boolean etc. Cloud computing uses both structured and unstructured data. Cryptography is applied for both type of data. While using traditional database encryption algorithms, the length and format of the structured data is changed. The original field on the table and the encrypted field are different in size, format and datatype. Especially the traditional encryption algorithms are not suitable for the field which is used as an index field. It leads to some additional work to handle the encrypted field. Format Preserving Encryption or Datatype Preserving Encryption is special type of encryption in which the original and encrypted field are same in datatype and format. After encryption, the encrypted filed is never changed. The format and data type are same as original filed. In this paper we proposed a special type of encryption that is AES based Format Preserving Encryption for structured data types on cloud.

Keywords: Cloud security, Cryptography, Format Preserving Encryption, AES based FPE, Security for structured data types

1 INTRODUCTION

Cloud computing is an emerging technology which provides IT resources over internet. The resources are server, database, networking, software and analytics. The database is the main resource provided by the cloud. It consists of both structured and unstructured data.

Most of the sensitive information are structured data which should be handled in a secured way. Strong Encryption and Authentication algorithm are required for structured data [1].

In Cryptography, Encryption converts the original information such as plaintext into unreadable secret information such as cipher text. In traditional encryption algorithms, the encrypted value is not same as plaintext in length and format. The encrypted value requires modification to the database and also changes of queries related to the database. To overcome this problem, applying one of the modern cryptography algorithms that is Format Preserving Encryption algorithm can be applied to structured data.

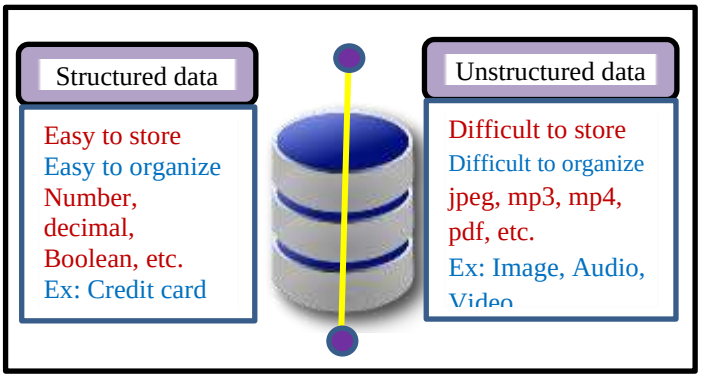

Fig 1 Structured Vs Unstructured data

2 ESSENTIAL OF FORMAT PRESERVING ENCRYPTION

In normal encryption, the length and format of the Plaintext was entirely changed. The database structure was also changed by the new encrypted value. The reconstruction of database is a very difficult process. If randomization was used in the encryption algorithm then referential integrity of the data base was also affected. The index of the column contains encrypted values and then the index is unusable for encrypted data. An encryption algorithm not only alters the structure of the database it also alters the queries passed to the database. Changing of database and queries are complex tasks and also cost prohibitive [2].

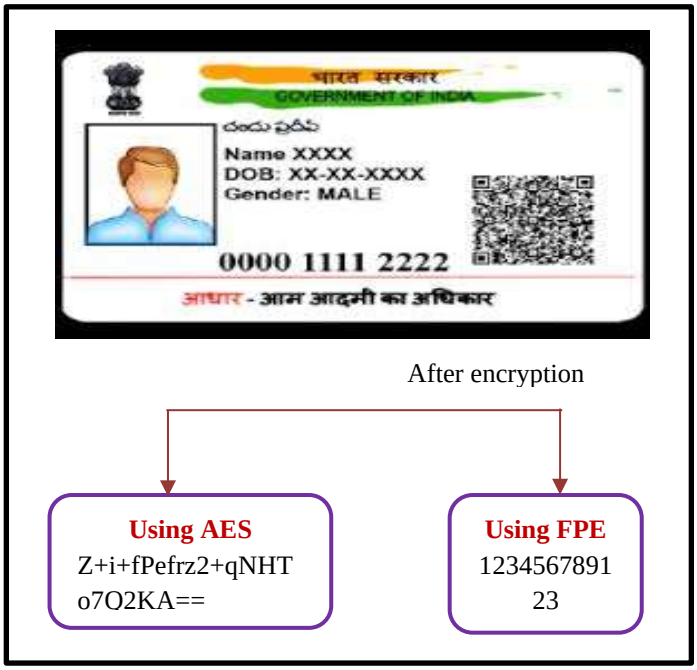

Fig 2 Aadhar number before and after encryption

Most of the block ciphers generate length preserving ciphertext. N bit block is mapped into another N bit block. But the data type of cipher text is not same as plain text. Format preserving encryption is a combination of length and data type conserving encryption [3].

Using Format Preserving Encryption the modification to the database structure, application program and front end is reduced. The cost of updating the database and the application program is also minimized. Most of the encrypted columns are fixed as an index of the table to speed up the searching operation. FPE does not change the index. In Relational Database Management, Referential integrity of the database is also maintained. Use of FPE enables improving database security in a transparent way on many applications[4].

3. FORMAT PRESERVING ENCRYPTION MODES

The National Institute of Standards and Technology (NIST), published the following two modes for FPE. FF1, originally called FFX (Format-preserving Feistel-based Encryption), was proposed by Bellare et al., and FF3 corresponds to the BPS-BC component proposed by Brier et al. Both operation modes are based on a non-binary Feistel structure[5].

FF1 needs extra input tweak to avoid the dictionary attack. It is very complicated to implement. It is more expensive. The operating mode of BPS is simple and efficient. A block cipher with a larger block size is to be preferred with BPS. It does not ensure data integrity.

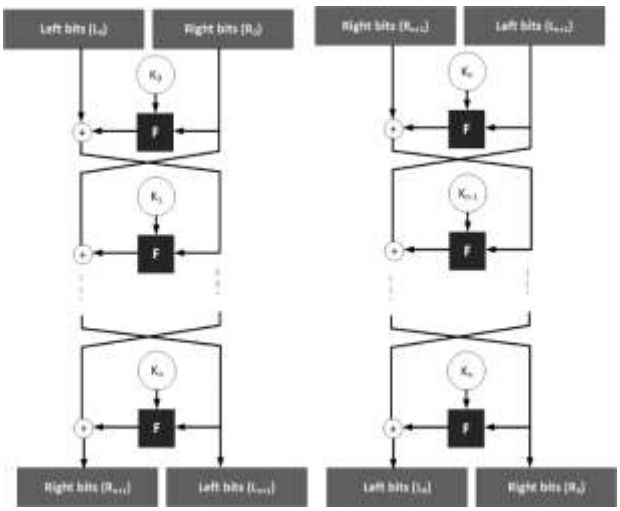

Fig 3 Feistel structure of FFX mode encryption

In Fiestel network the input is divding into two halves and inter relating a nonlinear function only to the right half. The result is added into the left half, and subsequently left and right halves are swapped. Since the nonlinear part requires most of the computation, two rounds of a Feistel cipher require about the same effort as a uniform transformation.

4 STREAM CIPHER BASED FPE

The proposed FPE model is based on AES-CTR mode. It is an AES block cipher mode that changes AES into a stream cipher. The mode is the simplest and most elegant of the confidentiality-only schemes. In the proposed algorithm traditional XOR function is replaced by Modulo Addition in encryption and Modulo Subtraction in decryption. AES-CTR is simple, and AES-CTR can be pipelined and parallelized. AES-CTR also supports pre computation of key stream. The length of the cipher text is same as length of the plaintext.

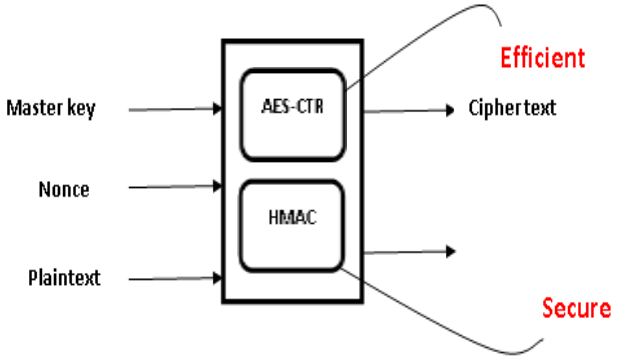

Fig 4 Format preserving Encryption Algorithm

4.1. HKDF – HMAC based Key Derivation Function

It is a simple Hashed Message Authentication Code (HMAC)-based key derivation function (HKDF), which can be used in various applications. A key derivation function (KDF) is a basic and an important material of cryptographic systems. The goal is to take some source of initial keying material and derive from it one or more useful strong secret keys. The nonce is used as salt for hash function. Based on Master key and nonce the HKDF is used to derive encryption key and authentication key. Unique fresh key is important for AES counter mode.

HKDF is an "extract and expand" KDF, where the output keys are derived in two steps:

1. In the extraction phase, the master key and non-secret "salt" are used to derive a "pseudorandom key".

2. In the expansion phase, the pseudorandom key computed in the extraction phase and is used to derive one or more secret keys of variable length.

4.2. Nonce

It is a number which is used only once. The nonce contains 16 octants and the least 8 octants contains the counter value. The counter value is initialized to zero and incremented for each block. CTR mode definitely requires a unique nonce per message and the nonce is generated through unique function.

4.3 AES Encrypt

AES operates on a state that is initialized with a plaintext block, and after encryption it contains the cipher text. The state can be pictured as a rectangular array of bytes. It consists of four rows and four columns. An encryption of a block starts with a transformation AddRoundKey, SubBytes, ShiftRows and MixColumns.

4.3.1. AddRoundKey

AddRoundKey is an XOR between the state and the round key. This transformation is its own inverse.

4.3.2. SubBytes

SubBytes is a substitution of each byte in the block independent of the position in the state. This is an S-box. It is a non-linear transformation. The S-box proved to be optimal with regards to non-linearity. The S-box is based on arithmetic in GF(28).

4.3.3. ShiftRows

ShiftRows is a cyclic shift of the bytes in the rows in the state and is clearly invertible.

4.3.4. MixColumn

Mixing the 4, 6, or 8 columns vertically by taking invertible linear combinations in GF(2^8) of the elements in each column.

4.3. Mapping

The mapping is performed before and after encryption. The proposed algorithm handles integer domain. The string to be encrypted is stored in a character array. During the process of mapping, a string is processed to remove the special characters and encoded to integer using index.

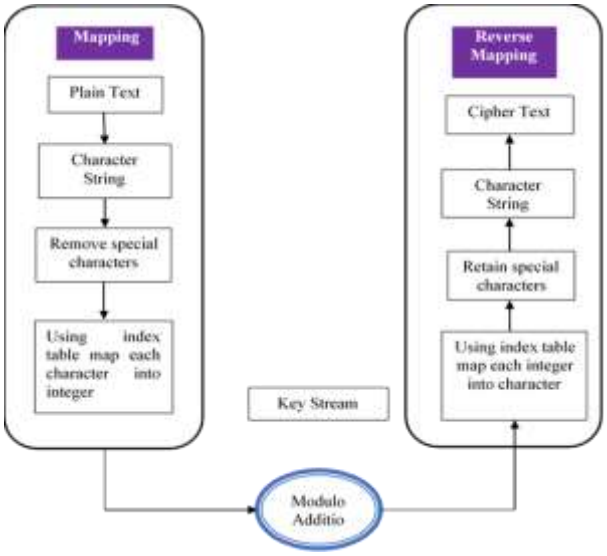

Fig 5 Mapping process

The processed string is used as plaintext for FPE. The output of the FPE is processed to produce a string having an original format. The original format is retained from the character array. While the decryption operation is performed the process is reversed. The given string s is mapped into set of integers p. Numeric values are represented by single digit and alphabets are represented by 2 digits.

4.4. Modulo Addition

In the conventional AE-CTR the plaintext is EX-OR with Key stream. In the proposed algorithm the EX-OR operation is replaced by Modulo Addition. MOD 10 for numeric or MOD 26 for alphabets is used to calculate the cipher text. The length of the key stream is equivalent to the length of the plaintext. The remaining bits are truncated.

4.5. HMAC

Encrypting the data ensures only privacy, however, it doesn't ensure integrity. Using a HMAC provides message integrity. In the proposed algorithm, Encrypt-then-MAC is applied i.e.

Encrypting the plaintext and then calculating a MAC of the cipher text. Encrypt-then-MAC is proven secure. Message authentication codes are used by the sender and the receiver who shares a secret key to validate information transmitted between these parties. Then MAC of the cipher text and nonce are calculated[6].

The HMAC process combines a secret key with the message data, hashes the result with the hash function, combines that hash value with the secret key again and then implements the hash function the second time. The output hash is 160 bits in length. The sender computes the hash value for the Encrypted data and sends both the cipher text and hash value.

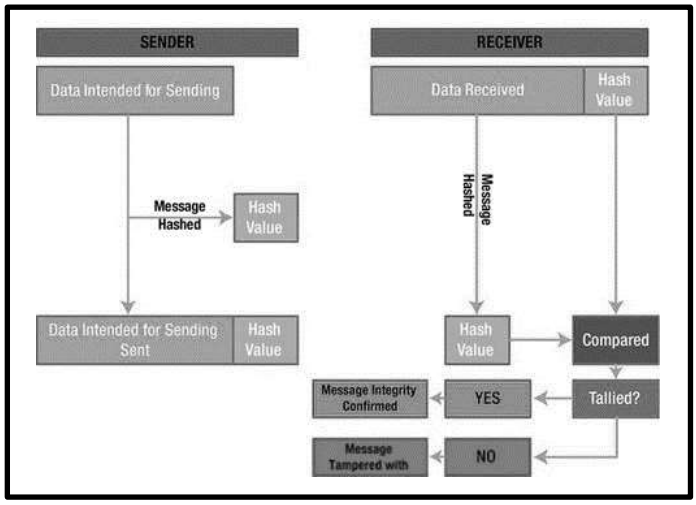

Fig 6 HMAC Authentication

Algorithm FPE.Encrypt(X, K)

Input:

X - Plaintext

K – Master Key

Output

c- Ciphertext

T – HMAC Tag

Prerequisites:

K1 – Encryption Key

K2 – Authentication key

S- Character array to represent X

p – Integer array (P- Integer to represent the plaintext)

L- Length of p

n – Number of bits to represent P

b – Number of Blocks (n bits are divided into blocks; Each contains 128 bits)

N –Nonce(N1,N2….Nb) N∑ ({0, 1}128)+. For each message the counter is initialized to 0. It is incremented for each block.

Ks – Key Stream generated by AES-CTR encryption

C – Integer domain which contains ciphertext

Steps:

1. Derive K1,K2 form HKDF(Master Key)

2. Store the Plaintext into S

3. Mapping. Remove the special characters and spaces from S. Map each character to integer array p. (P – convert p into single integer)

4. For j from 1 to b

i. Ksj= EK1(Nj)

5. Convert Ks to Integer (L digits) (Truncate remaining bits)

6. For I from 1 to number of characters is S

i. If(Si is integer)

Ci =(pi + Ksi) mod 10

Else if(Si is alphabet)

Ci =(pi + Ksi) mod 26 (Take 2 digits from p and Ks)

7. Map C to character array c (Retain the special characters)

8. Calculate T = HMACK2 (N,c)

9. Return c, T

The main drawback in CTR mode is that it is malleable. HMAC must always be added to confirm that the encrypted data has not been tampered with[7]. In some situation it could be

preferable for performance reasons to use a cipher mode such as GCM which combines encryption and authentication.

5 CONCLUSIONS

There is a powerful need for privacy of sensitive fields before data is shared with any cloud provider, semi-trusted vendors, partners etc. Network telemetry data, transaction logs etc. are frequently required to be shared for benefiting from a variety of Software-as-Service applications. Such sensitive data fields are of well-defined data formats. While designing privacy for sensitive fields, it may be desirable to preserve the length of the inputs, in order to avoid any re-constructing of packet formats or database columns of existing systems. The proposed algorithm is a flexible and arbitrary domain cipher. Format preserving encryption would be welcomed for many real-time applications. Especially for cloud environment in which the structured data should be handled in a more secured way.

REFERENCES

1. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta and S. Celma, "A New Method for Format Preserving Encryption in High-Data Rate Communications," in *IEEE Access*, vol. 8, pp. 21003-21016, 2020
2. Haiyun Ma, Zhonglin Zhang, "A New Private Information Encryption Method in Internet of Things under Cloud Computing Environment", Wireless Communications and Mobile computing vol. 2020, Article 8810987, 9 pages, 2020.
3. Pandey, Gyan Prakash, Implementation of DNA Cryptography in Cloud Computing and Using Huffman Algorithm, Socket Programming and New Approach to Secure Cloud Data (August 7, 2019).
4. A. Lenk, P. Marcus and I. Povoa, "GeoFPE: Format preserving encryption of geospatial data for the Internet of Things", *Proc. IEEE Int. Congr. Internet Things (ICIOT)*, pp. 172-175, Jul. 2018.
5. E. Brier, T. Peyrin and J. Stern, BPS: A Format-Preserving Encryption Proposal, Jan. 2020, [online] Available: http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/bps/bps-spec.pdf.
6. A. Rajendran, V. Balasubramanian and T. Mala, "Integrity verification using Identity based Provable Data Possession in multi storage cloud," 2017 International Conference on Computational Intelligence in Data Science(ICCIDS), Chennai, 2017, pp. 1-4, doi: 10.1109/ICCIDS.2017.8272669.
7. Dr.K.Chitra, S.Vidhya, "Format Preserving Encryption for small domain", March 2015, International Conference on Computing on Intelligence Systems.

Cite this article:

Dr S Vidhya, Dr P Ajitha, "Cloud security by format preserving encryption", Journal of Multidimensional Research and Review (JMRR), Vol.2, Iss.2, pp.76-84, 2021.