A CLOUD SECURITY APPROACH FOR DATA AT REST USING FPE

Nilekh Chaudhari[1]

[1]Cloud Research and Development, Syntel Ltd., Mumbai, India

ABSTRACT

In a cloud scenario, biggest concern is around security of the data. "Both data in transit and at rest must be secure" is a primary goal of any organization. Data in transit can be made secure using TLS level security like SSL certificates. But data at rest is not quite secure, as database servers in public cloud domain are more prone to vulnerabilities. Not all cloud providers give out of box encryption with their offerings. Also implementing traditional encryption techniques will cause lot of changes in application as well as at database level. This paper provides efficient approach to encrypt data using Format Preserving Encryption technique. FPE focuses mainly on encrypting data without changing format so that it's easy to develop and migrate legacy application to cloud. It is capable of performing format preserving encryption on numeric, string and the combination of both. This literature states various features and advantages of same.

KEYWORDS

Cloud Security, FPE, Encryption, Database, Feistal Ciphers

1. INTRODUCTION

Format preserving encryption provides vital solution for encryption problems in cloud scenario. "Format-preserving encryption (FPE) encrypts a plaintext of some specified format into a cipher text of identical format — for example, encrypting a valid credit-card number into a valid credit card number".

Using FPE we can implement the partial encryption which will eventually help us in performing effective searching operations over the set of encrypted credit card numbers.

2. FPE

2.1. WHAT IS FPE?

Encrypting Personally Identifiable Information (PII) in large databases has historically been difficult, because encrypting information typically implies expanding data and changing its format. Previous attempts to encrypt PII data like credit card numbers and Social Security Numbers without changing their format have used questionable cryptographic constructions.

Format-Preserving Encryption (FPE) is a fundamentally new approach to encrypting structured data, such as credit card or Social Security numbers. It uses a published encryption method with an existing, proven algorithm to encrypt data in a way that does not alter the data format.

2.2. How FPE Works?

FPE solution is realised in Microsoft Azure using Type-1 Feistel network. Type-1 Feistel networks use the round function to preserve format. The round function in practice can be build using the block ciphers like AES.

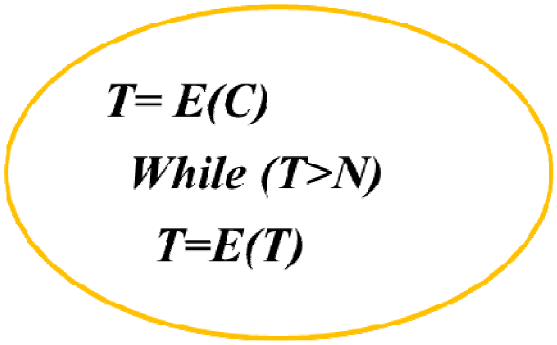
For each round of Feistel network we provide output of AES encryption as a key to that round. Iterating in similar fashion for nth times, we can achieve the format preserving.

2.3. Challenges Faced

Any numbers say 6 digits were encrypted using FPE and represented as an integer between 0 to 999999, which falls under the range from 2^{19} to 2^{20}. What if the final output exceeds this range? There are chances that the output becomes 7 or 6 digit, as $2^{20} = 1048576$ which is too long for 6 digits. So the chance of getting such variance is: $(2^{20} - 10^6) / 2^{20} = 4.6\%$. Hence format preservation is not 100% ensured using FPE. Here where we had to implement Cycle Walking over FPE.

2.3. Cycle Walking

Through Cycle Walking, we can encipher the same value again if we do not get the output as expected i.e. in a particular format. For Example, if we need to encrypt a cypher C in a particular range (N) then:



```
T= E(C)
  While (T>N)
    T=E(T)
```

Figure 1. Encryption Condition

Hence while decrypting; we will follow the same process to validate if the decrypted value is in the expected format, if it is not then again decrypt the same. The cycle-walking technique is then used to insure that the cipher text is in the appropriate range. Hence more accurate and more secured.

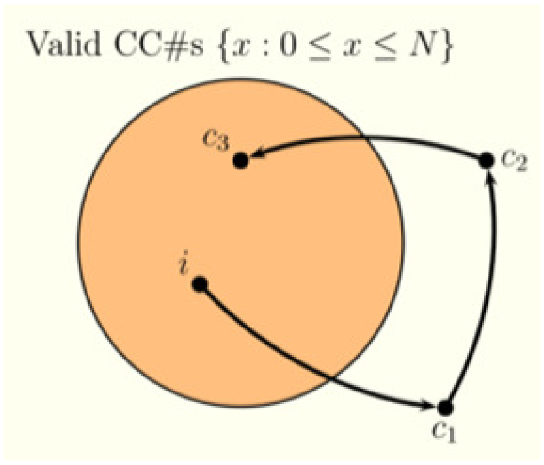


Figure 2. Cycle Walking

The circle is the set of all valid output: say all numbers between 0 and N. The input i when encrypted results in the number c1 which is greater than N and so we repeat. The result c2 =E (k, C1) is still greater than N, so we repeat again. Finally, c 3 ≤N is valid and so we output this as the encrypted value. This process is reversible. So, to decrypt with input (c3 ,k) , we just reverse the procedure, decrypting at each step with D(k,□) and finally getting the Output i.

3. ALGORITHM

A scheme for format-preserving encryption (FPE) is a function $E : K \times N \times T \times X \rightarrow X \cup \{\perp\}$ where the sets K, N, T, and X are called the key space, format space, tweak space, and domain, respectively. All of these sets are nonempty and $\perp \notin X$. We write $E_K^{NT}(X) = E(K, N, T, X)$ for the encryption of X with respect to key K, format N, and tweak T.

3.1. ENCRYPTION ALGORITHM

```
3.1.1  Factorized modulus into 'a' and 'b' in such a way that they are as close together as
       possible.
3.1.2  Copy plain text in X.
3.1.3  Iterate from 0 to r^th round as follows
           for i = 1, . . . , r(N) do
                 Divide the input plain text in left 'L' and right 'R' part
                       $L \leftarrow X / b$
                       $R \leftarrow X \% b$
                 Update the encrypted stream in previous round with the encrypted text in
                 current round.
                       $W \leftarrow (L + FK(N, T, i, R)) \% a$
                 Generate the encrypted text by
                       $X \leftarrow a * R + W$
           End For
3.1.4  Return encrypted text X.
```

3.2. DECRYPTION ALGORITHM

```
3.2.1  Factorized modulus into ‘a’ and ‘b’ in such a way that they are as close together as possible.
3.2.2  Copy encrypted text in Y.
3.2.3  Iterate from r^th round to 0 as follows
           for i = r(N), . . . , 1 do
               Divide the input plain text in intermediate W and right ‘R’ part
                   $W \leftarrow Y \% a$
                   $R \leftarrow Y / a$
               Update the encrypted stream in previous round with the encrypted text in current round.
                   $L \leftarrow (W - FK(N, T, i, R)) \% a$
               Decrypt the text by
                   $Y \leftarrow b * L + R$
           End For
3.2.4  Return plain text Y.
```

4. CONSIDERATIONS

While designing and implementing the solution following aspects are considered:

4.1. IV (INITIALIZATION VECTOR)

An initialization vector (IV) is an arbitrary number that can be used along with a secret key for data encryption (AES in our case). This number, also called a nonce, is employed only one time in any session. We have used 128 bit IV along with the key that is used for AES encryption. We used RNGCryptoServiceProvider to generate the random 128 bit IV.

4.2. KEY

Like IV we have generated the 256 bit key using the RNGCryptoServiceProvider. After generating the key we have pushed it to the azure blobs then application retrieved it to actually perform encryption or decryption.

5. IMPLEMENTATION

In an end to end solution the Key Generation Utility will first generated the random KEY and IV. The IV will be directly used in code as private variable. By doing so it is made more secure for scenarios like reverse engineering using reflection or introspection. The key will be pushed into Azure blobs. Code will access this key from the blob using Shared Access Signature. Once the key is accessed then we encrypt credit card numbers and store it in SQLAzure database. Similarly the decryption is also performed.

The below High-Level Architecture diagram explains the same.

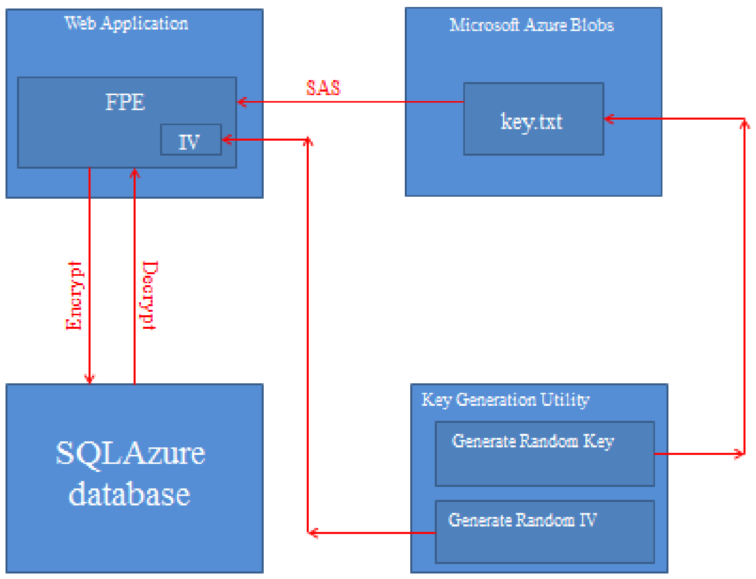


Figure 3. FPE Solution Architecture

6. ADVANTAGES

6.1 FPE allows storing data in same format and hence there is no need to change the structure of database table.

6.2 Using FPE we can implement the partial encryption which will eventually help us in performing effective searching operations over the set of encrypted credit card numbers.

6.3 Every number is encrypted into a unique value; hence it can be used as a primary key in your database table.

7. CONCLUSIONS

Using FPE we can enable a simpler migration path when encryption is added to legacy systems and databases, as required, for example, by the payment-card industry's data security standard (PCI DSS). Use of FPE enables upgrading database security in a way transparent to many applications and minimally invasive to others. Also it helps in performing the search operations over large set of encrypted credit card numbers.

ACKNOWLEDGEMENTS

I would like to thank my guide and mentor Yusuf Rangwala for his constant support. Also want to extend my regards for my employer Syntel Ltd. for offering me an opportunity to research and development on cloud computing.

REFERENCES

[1] Format-preserving_encryption. [Online] Available: http://en.wikipedia.org/wiki/Format-preserving_encryption
[2] Format-preserving_encryption. [Online] Available:https://eprint.iacr.org/2009/251.pdf
[3] A Few Thoughts on Cryptographic Engineering: Format Preserving Encryption [Online] Available: http://blog.cryptographyengineering.com/2011/11/format-preserving-encryption-or-how-to.html
[4] Botan - Format-preserving_encryption. [Online] Available: http://botan.randombit.net/manual/fpe.html
[5] Feistal Cipher. [Online] Available:http://en.wikipedia.org/wiki/Feistel_cipher
[6] On Generalized Feistel Networks. [Online] Available: https://eprint.iacr.org/2010/301.pdf

AUTHORS

Nilekh Chaudhari studies B.E. (Computer Engineering) from Mumbai University, Mumbai, India. I have 3 years of experience in IT industry and research field. I was formerly member of Computer Society of India. Currently working on Cloud Research and Development with Syntel Ltd. My major areaof focus is Microsoft Azure cloud platform.