International Journal of Database Theory and Application
Vol.9, No.5 (2016), pp.181-186
http://dx.doi.org/10.14257/ijdta.2016.9.5.18

Character based ASCII Encryption & Decryption on Cloud System

Richa Sharma[1] K. K. Parashar[2] and Jitendra Singh Sengar[3]

[1]ITM University, Gwalior, India
[2]Govt. Modal Science College, Gwalior (M.P.) India
[3]JSSRLPSP, Gwalior
neetu.18sharma@gmail.com, kkparashar1975@gmail.com,
jitendrasinghsengar@gmail.com

Abstract

This paper proposed a algorithm for automatic key generation which generate from an integer value to create more secure communication over cloud architecture. Character based ASCII encryption scheme is used to satisfy the current requirement to establish secure communication. In this work we have a byte level security at each level of data transmission. It helps to create batter security to perform encryption & decryption over cloud infrastructure i.e., from this work we have technique through which we generate private key which help to establish secure communication between two or more node while they are communicating over cloud system.

Keywords*: SAAS Application on Cloud Server, EPKGHED, Public cloud. ASCII Encryption, Private Key Generation*

1. Introduction

Through this work we have an algorithm to establish a secure communication between two or more node while they are communicating. The topic gained momentum after Craig Gentry's first construction of algebraic lattice theory in the year 2009 [2][3]. This work has become a solution, especially for the security or privacy problems of cloud system [7] and related applications theoretically in promising. FHE schemes follow Gentry's blueprint [4] [5] that were found to be inefficient for practical implementation [9] difference causes between the computational complexities to processing the cipher texts and to processing the plaintexts. The contribution of this work is high complexity by large *message expanse and* cipher text refreshing procedure during the bootstrapping. It is estimated that, with the improvements made the Encryption usage and thus encrypted data processing becomes imminent for suitable applications that fall within the multiplicative capacity of the proposed scheme.

2. Known Attacks

The public keys are described, claiming that the high noise cipher texts successfully defend all these attacks.

A. Factoring the exact multiple.

The chosen parameter values, the size of the exact multiple of P i.e., X0 is big enough so that, even the best known integer factoring algorithms General Number Field Sieve [13] will not be factorable X0. Even the factor P is targeted which is smaller than the size of total Q0, algorithms such as Lenstra's elliptic curve factoring [14] takes about exp (O(√*e*

ISSN: 2005-4270 IJDTA

))time to find P plan text. But, the point to be noted that, plan text P will not be recoverable directly as it is not prime and could further decomposed in smaller primes.

B. Over noise Brute-force attack.

Public key integers X0 and X1, the simple brute-force attack choose an R form the interval (-*r*, *r*), subtracting it from X1, which may be the required secret integer P. In a worst case, this may be repeated for all the integers R in the interval. The complexity of this attack will be about 2*r*. Õ(*g*) for *g* bit integers. Another integer more vulnerable to brute-force attack in HESP is the noise factor N used during the encryption. In fact, this integer clearly defines the overall security scheme because, guessing this number simply design the entire scheme, rather than guessing the secret integer P. The attack in the case of this integer will be chosen all the possible even integers N from the interval mentioned, and encrypting 0 with each N and public key. a plaintext bit encrypted using some N, the difference between the corresponding ciphertext and a ciphertext that encrypted 0 using the same N will be only in the significant bit. The complexity of this attack will be exponential in the size of N that is 2*r* and choosing $r = \omega(\lg n)$ foils this attack.

C. Continued fractions and lattice based attacks.

Grave Graham [11] described two methods to solve the two-element's PAGCD problem. In simple waythe continued fraction approach (Algorithm 11, [11]) recovers P if the condition $R < P/Q$ is satisfied. Similarly, lattice based algorithm (Algorithm 12, [11]) recovers P if the condition $R < P2 / (PQ)\varepsilon$ is satisfied for some real number ε. Analyzed in [5] for the case of a two-element PAGCD problem, possible to recover P when *r*/*g* is smaller than (*e* / *g*)2 . Since the parameter setting of HESP does not satisfy these constraints, the concerned methods fail to recover the value of P plan text.

D. The general common divisors attack.

the Theorem 31.2 and its corollaries discussed in. GCD(X0, X1), could be the smallest positive element in the set $\{AX0 + BX1 : A,B \in \mathbb{Z}\}$. This is possible because, A, B can be any integers including negative numbers. Now, if a common divisor exists for both X0, X1, and it will divide all the possible linear combinations of X0, X1. Modular reduction of a ciphertext with such common divisor results in the plaintext, because a ciphertext contains a linear combination of X0, X1. So that taking the pair of integers X0, X1 as co-prime to foils this attack.

3. Proposed Methodology for Encryption

To form new key generated through key change done by character based ASCII Encryption and Decryption on cloud system.

A. KeyGen(*n*):

A pattern integer from the right open interval [i, 2^(i+1)] as the secret key P. For i = 0,1,2,3,4,5,6,7 …..,*text size* , Select a random integer from initial character Qi from index 0, another integer Ri from the open interval random interval (*r*, 2*r*), and compute ch_int + 2 pow (nway+1)upto the size of text until the conditions Xi > Xi+1,Xi+2,Xi+3….., X*t* , go end X0 mod 2 = 1, and (X0 mod P) mod 2 = 0 are satisfied. Output the public key PK = (X0, X1,X2, X3,X4….., X*t*) and the secret key SK = P.

B. Encrypt(PK, M ∈ {0, 1}):

Select an integer B from (r, 2r) to add noise for encryption. Output the ciphertext as C = ch_int + 2 pow (nway+1), nway = 0,1,2,3,4,5,6,.....n give the n different way to encrypt data.

Decrypt (SK, C): Output the cipher text as C = ch_int - 2 pow (nway+1)

4. Proposed Algorithm:

A. Encryption

Take entire data or string text which is to encrypt. Read each character individually through sequence approach .

Step 1: Get ASCII value of each character and assign to"ch_int".
Step 2: Get numeric value from user that could be select n number of way (nway).
Step 3: Apply proposed formula for encryption to evaluate cipher text.
Step 4: Chepher text = ch_int + 2 pow (nway+1) +ascii ch_int upto the size of text till conditions Xi >Xi+1,Xi+2, Xi+3….., Xt , X0 mod 2 = 1, and (X0 mod ASCII int) mod 2 = 0 are satisfied. Step 5: End.

B. Decryption

Step 1: Get ASCII value of each character of chepher text.
Step 2: Insert numeric value (key) that could be select n number of way (nway).
Step 3: Apply step 4 formulas for decryption to get plaintext back from cipher text.
Step 4: Plan text data can get back from data where it is stored in cloud server after applying decryption by P = ch_int - 2 pow (nway+1)-ASCII ch_int.
Step 5: End,

5. Competitive Analysis

Table 5.1. Competitive analysis

ALGORITHM	**GIVEN BY**	**KEY LENGTH**	**BLOCK SIZE**	**SECURITY RATE**	**EXICUTION TIME**
Proposed Algo PKG	***RICHA ITM***	***16 BIT***	***32 BIT***	***Good***	***Fastest***
DES	IBM75	54 BIT	64 BIT	NOT ENOUGH	SLOW
AES	RIJMAN, JOAN	128,192, AND 256	128 BIT	EXELENT	MORE FAST
RSA	RIVEST, SAMIR 78	BASED ON No. of bit in N=P*Q	VARIANT	GOOD	SLOWEST
ECC	NEAL KOBLITZ	135	VARIENT	LESS	FASTEST

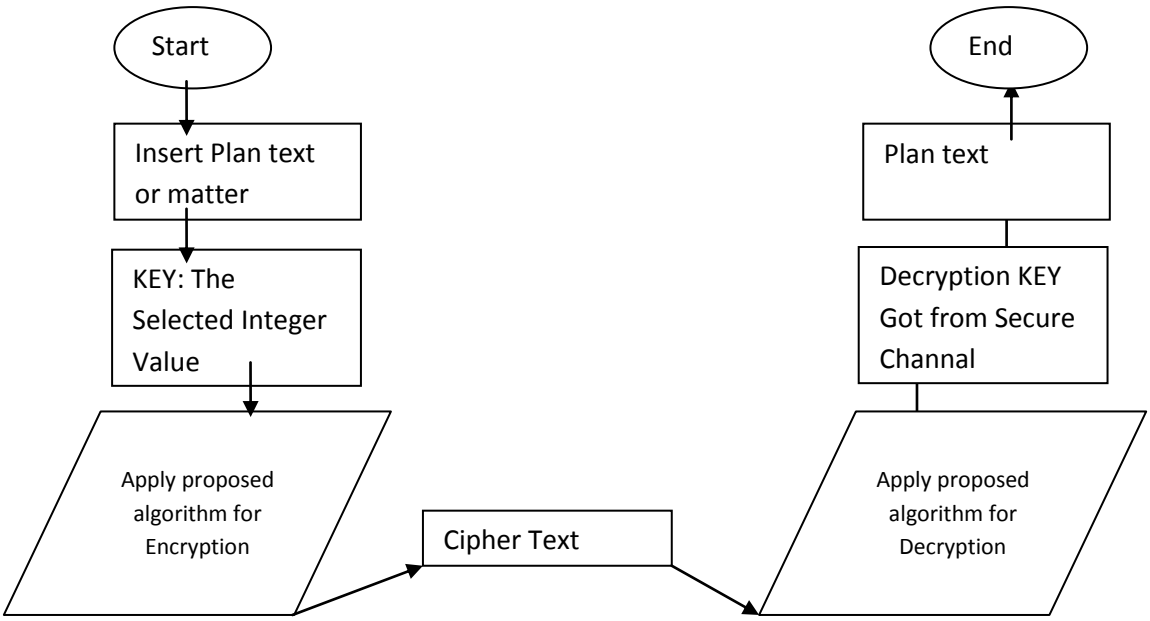
In proposed algorithm an integer assigned to encrypt the plaintext as input and then algorithm generate the number of block of written confidential matter so here working key size is 2 Byte (16 bit) and the generated block would be of at most 32 bit (8 byte) that make encryption faster with less time because of their small block size. the bit length of a fresh cipher text that encrypts single bit is Õ (n), that lead to an expansion ratio of n . The key in scheme PKG consists of only two elements of Õ(n) bits long which makes the complexity of key generation as Õ(n). That's make a considerable improvement over

homomorphic technique of [5] and [8]. encryption of an Õ(*n*) bit plaintext that take place with multiplication of Õ(*n*5). Õ (*n*) and modular reduction of this with Õ(*n*) bit X0 takes Õ(*n*) steps. In same way the bit complexity of decryption is drown Õ(*n*). Therefore, the overall complexity of the proposed system variant PKG is Õ(*n*). Similarly, a single plaintext bit is added in a cipher text of Õ(n) bits make expansion ratio also comparatively less that is *n*. these drastic improvements in bit complexity and ciphertext expansion, this conceptually simple scheme will be suitable for many practical applications that involve simple functions for homomorphic evaluation also.

6. Results Snapshot & Graph

A new approach to secure plan text has got as result of this entire work. This effort is faster, more secure & more reliable than previous task done & could use to establish secure communication with cloud server.

For Plan Text Richa Upto 5 Numeric Value. With Proposed algorithm we may encrypt from any number from the range 1,2,3……..,n and data flow direction is also shown below TA1 and DFD1 respectably, DFD showing it entire tasking first got plan text which is to send then apply any integer value as key and after getting this key value algorithm will activate to convert plane text to cipher text then send cipher text through ordinary channel and key through secure channel then receiver could convert cipher text to plan text with the help of key and decryption algorithm.



DFD 1. Representation of Entire Flow of Procedure

Table 6.1. Encrypted Cipher Text Corresponding to Applied Integer Value as Key

Numeric Value To Encrypt Plan Text Richa	Corresponding Cipher Text
1	Vjeoi
2	Wkfpj
3	Xlgqk
4	Ymhrl
5	Znism

A. Snapshot

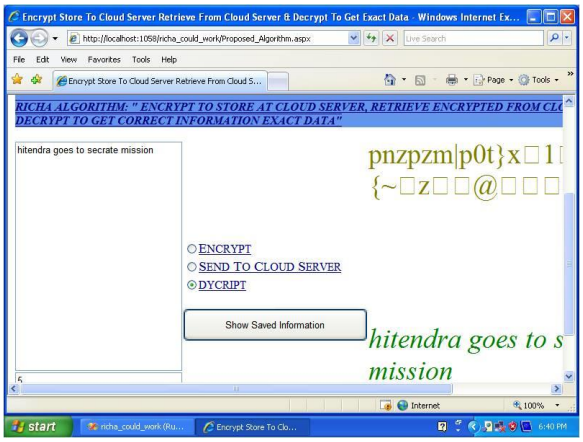


Figure 6.1. Snapshot 1

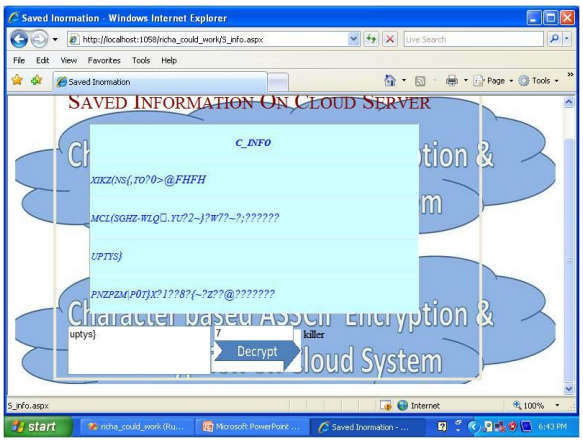


Figure 6.2. Snapshot 2

In above snapshot SS1 & in the SS2 it is clear that we may encrypt from assigned number (That number would be any one from series of integer 1,2,3,4,5,6 …..n) Below mentioned text and the important fact is only that a number can decrypt the text through which it is encrypted.

B. Graphical Representation :

Here Safe cipher value which is drawn with corresponding plane text word situation as shown below by Figure 6.3 and in Table 6.2. Where word position cipher value cipher text all shown in simple and effective way.

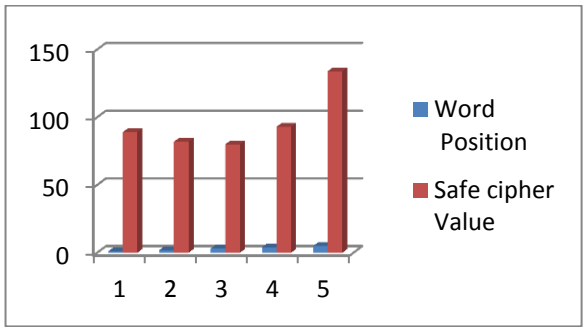


Figure 6.3. Word Position with Safe Cipher Value

Table 6.2. Word Position Cipher Value

Word Position	Word in cipher text.	
	Safe cipher Value	Text over value
1	89	Z
2	82	N
3	80	I
4	93	S
5	134	M

7. Conclusions

The entire implemented program represent the technique through which we are able to establish secure communication between two or more node while they are communication or wish to save their data in encrypted form on cloud system or architecture . To form this task batter and to form more secure environment of communication and secure data saving we are moving onward soft computing with proposed algorithm or technique through which we would be able to form instance change in algorithm to secure data & Communication.

References

[1] Efficient Public Key Generation for Homomorphic Encryption over the Integers CNC 2012, LNICST pp. 262–268, 2012
[2] Rivest, R., Adleman, L., Dertouzos, M.: On data banks and privacy homomorphisms. In:Foundations of Secure Computation, pp.169–180(1978)
[3] Gentry, C.: A Fully homomorphic encryption scheme. Ph.D. thesis, Stanford Univ, 2009
[4] Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC'09, pp. 169-178 ACM (2009)
[5] Smart, N.P., Vercauteren, F.: Fully Homomorphic Encryption with relatively small key and ciphertext sizes. In: Nguyen, P.Q., Pointcheval, D.(Eds), PKC'10, LNCS, vol.6056, pp. 420- 443. Springer (2010)
[6] Jitendra Singh Sengar, "Soft Computing Approach to Evaluate Trust Value of a Node in VNET with HMM through Supervised Learning", International Journal of Communication Technology for Social Networking Services Vol.1, No.1 (2013), pp.11-14
[7] Dijk, M.V., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully Homomorphic Encryption over the integers. In: Gilbert, H.(Ed), EUROCRYPT'10, LNCS, vol.6110, pp. 24-43.Springer (2010)
[8] Gentry, C.: Computing arbitrary functions of encrypted data. In Communications of the ACM, 53(3):97-105 (2010)
[9] GovindaRamaiah, Y., VijayaKumari, G.: State-of-the-art and Critique of Cloud Computing. In: NCNGCIS'11, pp. 50-60. IMS, Noida (2011)
[10] Coron, J.S., Mandal, A., Naccache, D., Tibouchi, M.: Fully Homomorphic Encryption over the Integers with Shorter Public Keys. In: P. Rogaway (Ed.), CRYPTO2011, LNCS, vol. 6841, pp. 487-504. Springer (2011)
[11] Brakerski, Z., Gentry, C., Vaikuntanathan, V.: Fully Homomorphic Encryption without Bootstrapping. In : Electronic Colloquium on Computational Complexity (ECCC) 18: 111 (2011)

