# A Secure Model on Cloud using a Modified Rivest, Shamir and Adleman Algorithm along with Gray Codes

Matthias D.[1] Osakwe B. P.[2] Anireh, V. I. E.[3]

Department of Computer Science Rivers State University, Port Harcourt, Nigeria.[1,2,3]

E-mails: [1]matthias.daniel@ust.edu.ng, [2]bolouereosakwe@gmail.com, [3]anireh.ike@ust.edu.ng

## *Abstract*

**The widespread presence of insecurity in cloud computing environment is rapidly increasing in most countries, especially among businesses in developing countries due to poor knowledge and evaluation by Firm and Information Technology Experts, this system will help eliminate the security challenges and loop-holes in which the current system is lacking behind with. The system was designed using a modified Rivert Shamir Adleman algorithm as well as server-side scripting language. The server side scripting languages used includes Java Server Pages, Hypertext Preprocessor, and My structured query language. The software system is a web-based application which was developed using JSP and PHP, MYSQL for the database. To enhance the security level of the system, gray codes where used to modify the RSA algorithm. This helped in solving not only the problem of security but also the speed of encryption and decryption of the previous system as gray codes are said to be weightless.**

***Keywords*: RSA Algorithm, Gray codes, Cloud computing, Server side scripting.**

## 1. INTRODUCTION

Data is one of the most valuable things in the world. It can be text, image, audio or video. Millions of data are transmitted in every second. Hence, a secure scheme on cloud of these data is one of the crucial topics in network environment.

A secure scheme on cloud simply means giving access only to an authorized user so as to gain access to data. In other for this to be possibly the use of an asymmetric algorithm (RSA) is needed.

The RSA Public-Key algorithm is widely used. RSA stands for Ron Rivest, Adi Shamir and Len Adleman who first described it in public in 1977. The proposed work will use the RSA algorithm along with gray codes to encrypt the data for security purposes, so that only the user concerned can obtain it. It will not allow unapproved access to that data by securing it.

Lots of algorithms are available for data security. A low complexity, symmetric cryptographic algorithm with circular queue and gray code is developed here. The security algorithms, which are using circular queue, can make decryption of ciphered message more difficult. Gray code is an ordering of numeral binary system such that two successive differ in only one bit. This algorithm uses the dimension of circular queue and starting of the chosen keyword letter as the flexible factors. Using these adjustable features is the forte of this algorithm and makes retrieval of the

plaintext by invaders more problematic. This algorithm can be used for securing text, image, audio, and video files. This algorithm will be less complex compared to the Multiple Access Circular Queue algorithm (MACQ). The circular queue and graycode in this algorithm makes the retrieval of plaintext by attackers more difficult.

## 2. RELATED WORKS

Over the years there has been the issue of storing and securing data, the cloud was a way in which data was stored but the issue of cloud data security persisted. Some of these issues were;

Privacy and Confidentiality: everyone who owned data uploaded in the cloud wanted privacy and confidentiality, and assurance was needed that they will get it.

Clients should be assured, and appropriate standards and privacy policy and procedures should be in order to maintain data protection for cloud consumers. The cloud seeker should be guaranteed confidentiality of data hosted on the cloud [1], [2].

Data integrity: The service company must let the user know precisely what particular details are held on the network, the origins and the data retention measures put in place.

Data availability: Data should be made available to the client at all times. Customer data is usually processed in chunk on multiple servers, /which also exist in separate places or clouds. In this situation, data access is a significant legal problem, because it is increasingly impossible to provide consistent and smooth service accessible. [2].

In order to tackle some of this issues some solutions were given:

Outsourcing to clouds is one of the software solutions to data securement. Outsourcing also known as data hosting is one of the most common methods to safe massive data storage [3], [4] where data owners encrypt their data using cryptographic primitives and storing encrypted data in clouds. To achieve data outsourcing, Fully Homomorphic Encryption (FHE) [5] was adopted in order for the cloud to perform operations over encrypted data, which allows direct addition and multiplication over the ciphertexts while preserving decryptability. Homomorphic encryption has also been used to guarantee data storage confidentiality [6], [7]however it was still an incomplete cryptosystem, which is highly unstable in operation, rendering it scarcely usable in implementations throughout the modern world.

Adequate Access Control is a means of preserving data kept in the cloud. Operating processes or software that limit access to information have historically included access protection, but usually reveals all the details if the device or program is compromised [8]. A safer approach is to secure the data using cryptography that only requires approved agencies to decode it.

Some of the following encryption methods were used;

Attribute Based Encryption (ABE) [9], [10], [11], [12], [13]is one of the most important access management strategies in cloud computing systems. In the past years, quite a few attribute-based access management systems [14], [15], [16], [8] were introduced, in which the data owners specify the access policies based on the attributes needed by the data and encrypt the data based on the access policies. This allows data owners to guarantee that ciphertexts can only be decrypted by those who follow the access policies.

Nevertheless, by implementing this ABE-based method, it becomes impossible to change the rules since the data owners will not retain the data in their local repositories until the data becomes outsourced into the cloud. It is often challenging to check the authenticity of the downloaded data because the data is not trustworthy in cloud housing

Key Policy Attribute Based Encryption (KP-ABE) is a primary policy attribute-based encryption scheme and was introduced in order to allow more general access protection [10]. This is the modified version of ABE's classical pattern.

Key Policy Attribute Based Encryption (KP-ABE) scheme is a shared key encryption strategy developed for single or multiple communications. Data is associated with the attributes within this scheme for which a public key is defined for each. Encrypter, which is the encrypter of the code, is connected with the collection of data or message attributes by encrypting it with a public key. Users are delegated to the data attributes with a tree framework for entry. The nodes of the entry tree are the threshold exits. Attributes are associated with the Leaf nodes. The user's secret key is specified to represent the tree structure for the access. Thus, if and only if the data attributes satisfy the access tree structure, the user can decrypt the data which is a ciphertext. In KP-ABE, encrypted message is associated with a set of attributes, and a monotonic access tree structure is associated with the decryption key of the consumer [17].

Like every other system the KP-ABE had its limitations; the encryptor can't determine who will be able to decrypt the files. It can choose for the data only descriptive attributes, and has no choice but to trust the key issuer. KPABE naturally does not fit for some applications.

Cipher Text Policy Attribute Based Encryption was introduced by[18] which is another modified form of ABE called CP-ABE. In the CP-ABE system, attribute policies are associated with data and attributes are associated with keys and the data can be decrypted only by certain keys where the associated attributes fulfill the data policy. CP-ABE functions in the reverse way of KP-ABE.

CP-ABE functions in the reverse way of KP-ABE. CP-ABE associates the ciphertext with an access tree structure and each user private keys is embedded with a set of attributes. The authority runs the Setup and Key Generation algorithm in ABE like KP-ABE and CP-ABE to produce MK, PK, and secret user keys from the device. Only authorized users (that is, users with planned access structures) may decrypt by calling the decryption algorithm. In CP-ABE, each user is equipped with a collection of attributes. His secret key is created based on his attributes

## 3. SYSTEM DESIGN

High-level design explains the architecture that would be used for developing a software product. The architecture diagram provides an overview of an entire system, identifying the main components that would be developed for the product and their interfaces.

The structural and behavioral details of the system includes the following key components: Internet, RSA technique, database is shown below in Figure 1.

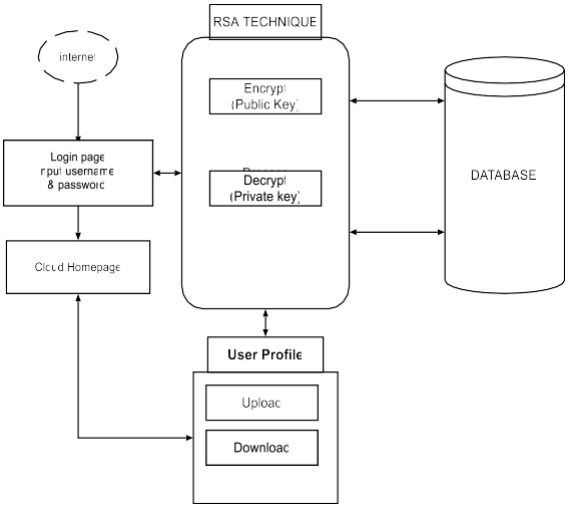


Fig. 1 Architecture of the system

RSA Algorithm

RSA algorithm is asymmetric cryptography algorithm. Asymmetric means that it works on two different keys i.e. Public Key and Private Key. As the name describes that the Public Key is given to everyone and Private key is kept private.

RSA algorithm is named after Ron Rivest, Adi Shamir and Len Adleman, who invented it in 1977 [RIVE78]. The basic technique was first discovered in 1973 by Clifford Cocks [COCK73] of CESG (part of the British GCHQ) but this was a secret until 1997. The patent taken out by RSA Labs has expired. The RSA cryptosystem is the most widely used public key cryptography algorithm in the world. It can be used to encrypt a message without the need to exchange a secret key separately. The RSA algorithm can be used for both public key encryption and digital signatures. Its security is based on the difficulty of factoring large integers.

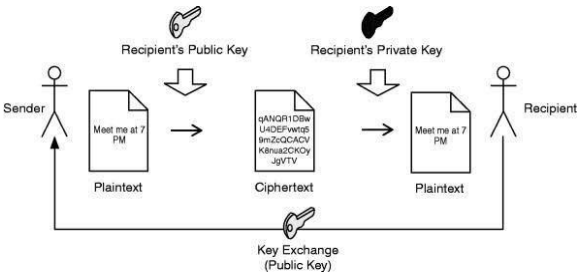


Fig. 2 How asymmetric cryptography works

The RSA algorithm ensures that the keys as shown in figure 1 above are secure as possible.

The following steps is how the RSA encrypts data:

First step towards the RSA encryption is to create the public and private keys. Note that I have secured the keys with a passphrase.

1. $rsa = new Crypt_RSA();

2. $rsa->setPassword('pa$$wrd5');

3. $keys=$rsa->createKey(1024);

4. echo $keys['privatekey'];

5. echo $keys['publickey'];

The code is self-explaining. createKey() method is taking the bit value of the key and output an array which contains the Private and Public keys. setPassword() is optional, you can omit it if you don't want to create the keys with a passphrase. If you want to use 2048 bit key then provide 2048 as an argument in createKey() instead 1024.

Once keys are ready you can start the encryption. I am using the public key. Passphrase is not required when encrypting.

6. $rsa->loadKey($keys['publickey']);

7. $plaintext = 'Text to be transmitted securely !!!';

8. $ciphertext = $rsa->encrypt($plaintext);

9. echo $ciphertext;

Decryption goes as follow (using the private key), note that the passphrase is mandatory:

10. $rsa->loadKey($keys['privatekey']);

11. $rsa->setPassword('pa$$wrd5');

12. $re_plaintText= $rsa->decrypt($ciphertext);

13. echo $re_plaintText;

Example

This is an extremely simple example using numbers you can work out on a pocket calculator (those of you over the age of 35 45 55 can probably even do it by hand).

Select primes p=11, q=3.

n = pq = 11.3 = 33
phi = (p-1)(q-1) = 10.2 = 20

Choose e=3
Check gcd(e, p-1) = gcd(3, 10) = 1 (i.e. 3 and 10 have no common factors except 1), and check gcd(e, q-1) = gcd(3, 2) = 1 therefore gcd(e, phi) = gcd(e, (p-1)(q-1)) = gcd(3, 20) = 1

Compute d such that ed ≡ 1 (mod phi) i.e. compute d = (1/e) mod phi = (1/3) mod 20 i.e. find a value for d such that phi divides (ed-1) i.e. find d such that 20 divides 3d-1. Simple testing (d = 1, 2, ...) gives d = 7 Check: ed-1 = 3.7 - 1 = 20, which is divisible by phi.

Public key = (n, e) = (33, 3) Private key = (n, d) = (33, 7).

This is actually the smallest possible value for the modulus n for which the RSA algorithm works.

Now say we want to encrypt the message m = 7, $c = m^e \bmod n = 7^3 \bmod 33 = 343 \bmod 33 = 13$.

Hence the ciphertext c = 13.

To check decryption, we compute

$m' = c^d \bmod n = 13^7 \bmod 33 = 7$.

Note that we don't have to calculate the full value of 13 to the power 7 here. We can make use of the fact that

$a = bc \bmod n = (b \bmod n)\cdot (c \bmod n) \bmod n$

So we can break down a potentially large number into its components and combine the results of easier, smaller calculations to calculate the final value.

One way of calculating m' is as follows:-

Note that any number can be expressed as a sum of powers of 2.

In particular 7 = 4 + 2 + 1. So first compute values of $13^{2}, 13^{4}, 13^{8}, \ldots$ by repeatedly squaring successive values modulo 33. $13^{2} = 169 \equiv 4, 13^{4} = 4.4 = 16, 13^{8} = 16.16 = 256 \equiv 25$. Then, since 7 = 4 + 2 + 1, we have $m' = 13^{7} = 13^{(4+2+1)} = 13^{4}.13^{2}.13^{1}$ $\equiv 16 \times 4 \times 13 = 832 \equiv 7 \pmod{33}$

Now if we calculate the ciphertext c for all the possible values of m (0 to 32), we get

m 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

c 0 1 8 27 31 26 18 13 17 3 10 11 12 19 5 9 4

m 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32

c 29 24 28 14 21 22 23 30 16 20 15 7 2 6 25 32

Note that all 33 values of m (0 to 32) map to a unique code c in the same range in a sort of random manner. In this case we have nine values of m that map to the same value of c - these are known as unconcealed messages. m = 0, 1 and n-1 will always do this for any $n$, no matter how large. But in practice, these shouldn't be a problem when we use large values for $n$ in the order of several hundred bits.

If we wanted to use this system to keep secrets, we could let A=2, B=3, ..., Z=27. (We specifically avoid 0 and 1 here for the reason given above). Thus, the plaintext message "HELLOWORLD" would be represented by the set of integers $m_{1}, m_{2}, \ldots$

(9,6,13,13,16,24,16,19,13,5)

Using our table above, we obtain ciphertext integers $c_{1}, c_{2}, \ldots$

(3,18,19,19,4,30,4,28,19,26)

Note that this example is no more secure than using a simple Caesar substitution cipher, but it serves to illustrate a simple example of the mechanics of RSA encryption.

Remember that calculating $m^{e} \bmod n$ is easy, but calculating the inverse $c^{-e} \bmod n$ is very difficult, well, for large n's anyway. However, if we can factor n into its prime factors p and q, the solution becomes easy again, even for large n's. Obviously, if we can get hold of the secret exponent d, the solution is easy, too.

## 4. RESULTS AND DISCUSSION

The old RSA algorithm creates its public keys using prime numbers to achieve a one way encryption also known as asymmetric encryption. Since the algorithm relies on random numbers to create its public key by multiplying two random prime numbers

i.e x*y = n

Where n = public key.

Anyone can crack the keys of the bits by simply guessing the random numbers right. So therefore we introduced gray codes into the algorithm to modify it. Gray codes are like binary numbers they convert actual numbers to 0s and 1s. This gray codes are weight less because unlike binary numbers they change only one number when moving from one number to the other.

i.e 00 01 11

000 001 011 etc.

Guessing these codes will be difficult therefore increasing the security level of the algorithm. Gray codes also aids error correction.

The new system developed will also be using server side scripting languages, during the course of the research it was observed that there were so many journal and publication on securing big data in cloud no doubt, somewhere done using RSA algorithm amongst this none was done using server side scripting language which makes the system unique the counter the challenge the previous system developed could not tackle, the new system will satisfy the objectives listed in chapter one of my project.

In development of the application, quite several tools were available but, a choice had to be made among them. The following combination of tools were used to develop the software:

JSP, PHP and MSQL

Java Server Pages (JSP) is a server-side programming technology that enables the creation of dynamic, platform-independent method for building Web-

based applications. JSP have access to the entire family of Java

APIs, including the JDBC API to access enterprise databases.

PHP: Hypertext Preprocessor is a self-referentially open source, server-side, HTML embedded scripting language used to create dynamic Web pages.

The modified RSA algorithm when compared with other cryptographic algorithms as shown on Table 2 below is the most secured and has a very fast speed even with its large key size. This is possible because of the introduction of gray codes to the original RSA algorithm.

Table 2 Comparison of cryptographic algorithm strength

| **Algorithm** | **Key size** | **Speed** | **Speed depends on keys** | **Security** |
|---|---|---|---|---|
| **DES** | 56 bits | Slow | Yes | Insecure |
| **3DES** | 112/168 bits | Very slow | No | Moderately secure |
| **AES** | 128/192/256 bits | Fast | Yes | Secure |
| **Modified RSA** | 1024 bits and above | Very fast | Yes | Most Secure |

It is shown in Table 3 that the higher the key size the higher the level of security DES has a bit size of 56 bits this makes it prone to the brute force attack which makes DES very crackable. RSA having a large bit size has a very high security level.

Figure 3 shows a graphical representation of Table 3 and it is evidence that RSA has the highest security level due to its large bit size.

Table 3 Security level comparison of cryptographic algorithm depending on the length of encryption key

| **Algorithm** | **Bit size** | **Security level in percentage** |
|---|---|---|
| **DES** | 56 bits | 50 |
| **3DES** | 112 bits | 65 |
| **AES** | 128 bits | 70 |
| **RSA** | 2048 bits and above | 99.5 |

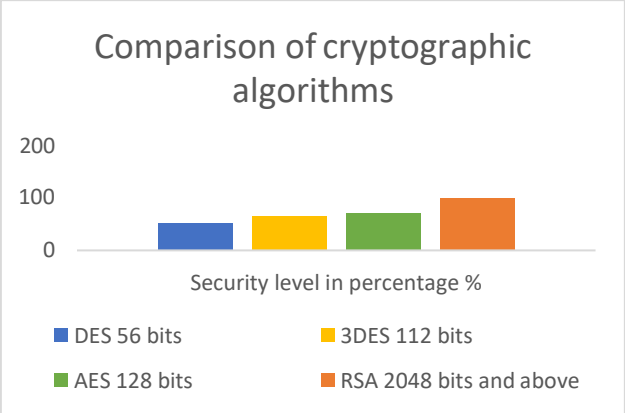


Figure 3 Column graph representing the comparison of cryptographic algorithms

## 5. CONCLUSION

This research was able to successfully incorporate all the requirements specified by the user. Proper care has been taken during database design to maintain data integrity and to avoid data redundancy. The algorithm is embedded in a server side script to help secure data uploaded in the cloud and give access only to an authorized user.

The project is designed and coded in such a way that any further modifications that are needed in the future can be easily implemented without affecting the functionality of the system. The technical documentation provided in this research helps the application developers understand the internal

architecture of the system and thus assists them in enhancing the system.

This project is purely user friendly and platform independent, so user can run this tool in any environment. It is very easy to implement or add many features to this tool.

**References**


[1] N.Benvan, measuring usability as quality of use, software quality J.4 (1995) 115-130.

[2] V.Sandhya, "A Study on Various Security Methods in Cloud Computing", International Journal of Advanced Research in Computer Science, Nov-Dec 2011, Volume 2, No.6.

[3] L.Wei, H. Zhu, Z. Cao, X. Dong, W. Jia, Y. Chen, and A. V. Vasilakos, "Security and privacy for storage and computation in cloud computing," Information Sciences, vol. 258, pp. 371–386, 2014

[4] J.Baek, Q. H. Vu, J. K. Liu, X. Huang, and Y. Xiang,"A secure cloud computing based framework for big data information management of smart grid," Institute of Electricals and Electronic Engineering, 2015.

[5] C.Gentry, "A fully homomorphic encryption scheme," Ph.D. dissertation, Stanford University, 2009.

[6] S. Kamara and K. Lauter, "Cryptographic cloud storage," in International Conference on Financial Cryptography and Data Security. Springer, 2010, pp. 136–149.

[7] A.Hamlin, N. Schear, E. Shen, M. Varia, S. Yakoubov, and A. Yerukhimovich, "Cryptography for big data security," in Big Data: Storage, Sharing, and Security, F. Hu, Ed. CRC Press, 2016, ch. 10, pp. 241–288.

[8] T.Jung, X.-Y. Li, Z. Wan, and M. Wan, "Privacy preserving cloud data access with multi-authorities," in INFOCOM, 2013 Proceedings Institute of Electricals and Electronic Engineering. IEEE, 2013, pp. 2625–2633.

[9] A.Sahai and B. Waters, Fuzzy Identity-Based Encryption, in Proceedings of Advances in Cryptology - EUROCRYPT '05, ser. LNCS, vol. 3494. Springer, 2005, pp. 457–473.

[10] V.Goyal, O. Pandey, A. Sahai, and B. Waters,"Attribute-based encryption for fine-grained access control of encrypted data," in Proceedings of the 13th ACM conference on Computer and communications security. ACM, 2006, pp. 89–98

[11] B.Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," Public Key Cryptography– PKC 2011, pp. 53–70.

[12]A.Lewko and B. Waters, "Decentralizing attribute-based encryption," Advances in Cryptology–EUROCRYPT 2011, pp. 568–588, 2011.

[13] C.Hu, X. Cheng, Z. Tian, J. Yu, K. Akkaya, and L. Sun, "An attributebased signcryption scheme to secure attribute-defined multicast communications," in SecureCommunication, Springer, 2015, pp. 418–435.

[14]K.Yang, X. Jia, K. Ren, B. Zhang, and R. Xie,(2013) "Dac-macs: Effective data access control for multiauthority cloud storage systems," Information Forensics and Security, Institute of Electricals and Electronic Engineering Transactions on, vol. 8, no. 11, pp. 1790– 1801.

[15] K. Yang and X. Jia, "Expressive, efficient, and revocable data access control for multi-authority cloud storage," Parallel and Distributed Systems, Institute of Electricals and Electronic Engineering Transactions on, vol. 25, no. 7, pp. 1735–1744, 2014.

[16] K. Yang and X. Jia, "Attributed-based access control for multi-authority systems in cloud storage," in Distributed Computing Systems (ICDCS), 2012Institute of Electricals andElectronic Engineering 32nd International Conference on. IEEE, 2012, pp. 536– 545.

[17] R.Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with non-monotonic access structures". In Proceedings of CCS'06, New York, NY, 2007.