ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 3, Special Issue 20, April 2016

Decisive Key Management for Cloud Data Security Using Discrete Logarithm

M. Janaki

Research Scholar, Department of Computer Science, Bharathiar University, Coimbatore, Tamilnadu, India.

Assistant Professor, Dr.Umayal Ramanathan College for Women, Karaikudi, Tamilnadu, India.

mjanaki81@gmail.com

***Abstract* - Cloud Data security can be obtained by managing the key used for encipherment properly. When the key is easily found then the sensitive information transformed through the cloud will be leaked. How hard the key finding and the time taken for doing so, decides the strength of the security. Keys play an important role in securing data transferred through cloud networks. It is not possible to keep encipher and decipher algorithms hidden from unauthorized users always. It is also meaningless to safe guard the process. Hence a better choice will be managing the keys used for encipher/decipher effectively. In this paper, discrete logarithm is used for key agreement, hence it is very hard to detect it. Instead of sharing encipherment key, it is automatically computed without sharing it through the cloud. The time taken for generating the key with various lengths are found. The results are tabulated and a chart is drawn to visualize the key generation time for different key lengths easily.**

***Keywords* - Data security, Discrete logarithm, Key management, Key generation.**

I. INTRODUCTION

Keys play an important role in securing data transferred through the cloud. It is not possible to keep the encipher and decipher algorithms hidden from unauthorized users always. It is also meaningless to safe guard the process. Hence a better choice will be managing the keys used for encipher/decipher effectively. Discrete logarithm can be used to manage encipherment key generated by dispatcher and acceptor. Two important phases of key management are key agreement and antiphon generation.

The key exchange methodology based on discrete log problem will provide a better way to share keys between users. The dispatcher and the acceptor will get a common key without sharing it through cloud. Automatically the common key is generated by sharing only a prime number and a primitive root of the number. Discrete logarithms are logarithms defined with regard to multiplicative cyclic groups.

II. REVIEW OF LITERATURE

Shiang-Feng Tzeng, Cheng-Chi Lee, Tzu-Chun Lin presented an improved version of Shen and Chen’s scheme to reduce the computational time required for key generation and derivation [1]. Nabeel, M., Ning Shang, Bertino, E. formalized a new key management scheme, called broadcast group key management (BGKM), and then gave a secure construction of a BGKM scheme called ACV-BGKM. The idea is to give some secrets to users based on the identity attributes they have and later allow them to derive actual symmetric keys based on their secrets and some public information.

A key advantage of the BGKM scheme is that adding users/revoking users or updating acps can be performed efficiently by updating only some public information [2]. Using our BGKM construct, they proposed an efficient approach for fine-grained encryption-based access control for documents stored in an untrusted cloud file storage.

An important problem in public clouds is how to selectively share documents based on fine-grained attribute-based access control policies (acps). An approach is to encrypt documents satisfying different policies with different keys using a public key cryptosystem such as attribute-based encryption, and/or proxy re-encryption [3]. However, such an approach has some weaknesses: it cannot efficiently handle adding/revoking users or identity attributes, and policy changes; it requires to keep multiple encrypted copies of the same documents; it incurs high computational costs.

A direct application of a symmetric key cryptosystem, where users are grouped based on the policies they satisfy and unique keys are assigned to each group, also has similar weaknesses [4]. They observe that, without utilizing public key cryptography and by allowing users to dynamically derive the symmetric keys at the time of decryption, one can address the above weaknesses.

III. DISCRETE LOGARITHM

If *G* is a multiplicative cyclic group and *g* is a generator of *G*, then from the definition of cyclic groups, we know every element *h* in *G* can be written as g^x for some *x*. The discrete logarithm to the base *g* of *h* in the group *G* is defined to be *x* . The discrete logarithm problem is defined as: given a group *G*, a generator *g* of the group and an element *h* of *G*, to find the discrete logarithm to the base *g* of *h* in the group *G*.

Discrete logarithm problem is not always hard. The hardness of finding discrete logarithms depends on the groups. the dispatcher and the acceptor compute the same key for encipher and decipher since it is a symmetric key cryptography. Dispatcher computes her encipher key by getting the acceptor's shared secret which is got from a chosen prime number and its primitive root along with her own shared secret.

The acceptor computes the decipher key by getting dispatcher's shared secret which is got from the chosen prime number and its primitive root along with his own sealed secret. Since discrete algorithm is used for key agreement, it is very hard to detect it. Instead of sharing encipherment key, it is automatically computed without sharing it through the cloud.

IV. KEY MANAGEMENT

Data Security can be obtained by managing encipherment key properly. When the key is easily found then the sensitive information transformed through cloud will be leaked. How hard the key finding and time taken for doing so, decides the strength of the security. There are two algorithms dedicated for key management. The first algorithm is for key contract between acceptor and dispatcher. The second algorithm is for generating the antiphon key which will be used for encipherment and decipherment [5]. The following notations are used in algorithms defined further for Key Contract, Antiphon generation.

A. Key Contract Algorithm

The first algorithm is for key agreement between dispatcher and acceptor based on discrete logarithm. Using this algorithm the prime and its primitive root are agreed between users. Later shared secret is computed from these values and sealed secret. Using shared secret the common key for encipher and decipher is calculated.

Usually in symmetric key cryptography, key used for encipher and decipher will be the same. It will be shared through cloud between acceptor and dispatcher well in advance before starting encipher process [6]. When encipherment key is shared through communication channel it is suspected to be hacked by hackers.

After knowing encipherment key they will be easily converting cipher text into original text and cause unnecessary issues. In order to safe guard the key, better avoid transmitting it into cloud. Usage of discrete logarithm will allow hiding of secret key which is never transmitted through unsecured communication channel.

The dispatcher and acceptor agrees with initial inputs prime 'P' and primitive root 'Pr'. Larger the prime number, time taken for guessing the key will be more. Hence based on sensitivity of data and confidentially required, acceptor and dispatcher should choose large prime numbers. Then a corresponding primitive root also have to be chosen and these values can be shared through network. Though these values are hacked, the hacker cannot find the key because further processing is available to seal the key.

Once these initial values are agreed between acceptor and dispatcher. Next step is, dispatcher chooses his own Secret value 'DS' and calculate his shared secret 'ds' value by using the formula ds = (Pr ^ DS) mod P. Acceptor will choose his own secret values 'AS' and calculates his shared secret 'as' values by using the formula as = (Pr ^ AS) mod P. The sealed secret values 'DS' and 'AS' are never shared through the cloud.

There lies the strength of this technique, without knowing these values the key cannot be found. Since these are not shared through cloud, cannot be hacked by hackers. Shared secret values 'ds' and 'as' are exchanged between acceptor and dispatcher. By knowing these values, hacker cannot guess the Key. This is possible because discrete logarithm methodology is used here.

After shared secrets are exchanged between acceptor and dispatcher, then next step is to compute the common key to be used for both encipher and decipher.

The dispatches computer the key by using the formula EDK = (as ^ DS) mod P. Here the key is calculated by using dispatcher sealed secret 'DS' and acceptor shared secret 'as' along with agreed prime number.

By using this key, dispatcher will encipher sensitive data and send it to acceptor through cloud. At the receiving end, the dispatcher will compute the key by using the formula EDK = (ds ^ AS) mod P. Here key is calculated by using acceptor sealed secret 'AS' and dispatcher shared secret 'ds' along with agreed prime number.

Now acceptor have computed same key used by dispatcher for encipher. This key can be used by acceptor to decipher the enciphered message sent by the dispatcher. Thus acceptor will retain original message safely without sharing common key through cloud. Acceptor and dispatcher are able to calculate the key of their own due to discrete logarithm technique used here.

B. Key Contract Technique

The prime number and its primitive root are taken as initial values and they are agreed between dispatcher and accepter. Let Prime P = 9973 and Primitive root Pr = 9962 as initial values. Dispatcher after agreeing the prime and primitive root, hunts a sealed secret 'DS' which is never revealed to others , since it is not shared through cloud.

The Dispatcher Sealed Secret DS = 4. Then dispatcher computes shared secret 'ds' using modular arithmetic. The Dispatcher Shared Secret ds = 9962 ^ 4 mod 9973 = 9848864207205136 mod 9973 = 4668. Acceptor after agreeing the prime and primitive root, hunts a sealed secret 'AS' which is never revealed to others , since it is not shared through cloud.

The Acceptor Sealed Secret AS = 3. Then acceptor computes shared secret 'as' using modular arithmetic. The Acceptor Shared Secret as = 9962 ^ 3 mod 9973 = 988643265128 mod 9973 = 8642. After shared secrets are exchanged between dispatcher and acceptor, dispatcher calculates Encipher/Decipher key 'EDK' using modular arithmetic.

The shared secret of 8642 is sent to Dispatcher. The Dispatcher's Encipher key EDK = 5577724352378896 mod 9973 = 4410. This EDK key is used by dispatcher for generating the antiphon and later for enciphering the sensitive data to be sent through cloud.

After shared secrets are exchanged between dispatcher and acceptor, acceptor calculates Encipher/Decipher key 'EDK' using modular arithmetic. The shared secret of dispatcher is sent to Acceptor. The Acceptor's Encipher key EDK = 101716765632 mod 9973 = 4410.

This EDK key is used by acceptor for generating the antiphon and later for deciphering the scrambled data received from dispatcher through cloud. The sample data explained above shows that, after computation is performed both acceptor and dispatcher is obtaining same key as a magic, though they have not shared the key through cloud.

V. ANTIPHON GENERATION ALGORITHM

This algorithm is capable of separating exchanged key as into consecutive two digits from left to right and equivalent key from the VIJANA character set is taken as the Antiphon Key [7][8]. From the key contract algorithm described before, a common key is computed which can be used for encipher and decipher.

In VIJANA cryptography, by using the common key calculated by the acceptor and the dispatcher, Antiphon (Key word) should be generated using VIJANA character set. This Antiphon is used for encipher and decipher. The VIJANA character set has 95 characters which includes 26 uppercase alphabet, 26 lower case alphabets, 10 numerals, 6 braces, 7 operators, 19 special characters and a space.

For internal calculation, Ascii characters are taken into account. First step to generate the antiphon is finding the square of the common key computed before. Then separate the number into two consecutive digits from left to right. Take first two digit and check whether the two digit number is greater or lesser than the character set limit 95.

If it is greater than 95 then remainder should be found and taken as the value [10]. Now pick the corresponding character in VIJANA character set by seeing the index as the 2 digit number. Now pick corresponding characters for all 2 digit numbers from the VIJANA character set.

All these characters are concatenated to get the Antiphon. This antiphon is used for enciphering original message by dispatcher and at destination the same key is used by acceptor for deciphering scrambled data.

A. Antiphon Generation

After the encipher/decipher key is computed using key contract algorithm, next phase is to generate the antiphon to be used for encipher and decipher. For generating the antiphon, EDK need to be squared. Then every consecutive two digits of the number is separated from left to right. The corresponding ASCII value is taken for the two digits related to only printable keys [9].

Let Encipher/Decipher Key EDK = 43322723 and the Square S=1876858328134729. Every two consecutive digits of the number is separated from left to right as 18, 76, 85, 83, 28, 13, 47and 29. The equivalent character for 18 is 1, 76 is k, 85 is t, 83 is r, 28 is ;,13 is ,, 47 is N and 29 is <.

The Antiphon generated is **K = 1ktr;,N<**

The length of the key is generated is 64 bits. Apart from this, key length can be 16 bits, 32bits, 64 bits, 128 bits, 256 bits and 512 bits so on. For a sample of data keys with different lengths are generated. The time taken for the following key lengths are shown in the following table.

Table 1. Antiphon Generation Time

Key Length (bits)	Antiphon Generation Time (ms)
16	0.125
32	0.2508
64	0.375
128	0.4621
256	0.5007
512	0.6254

The antiphon generation time tabulated above can be shown in pictorial form in Figure 1. for clear depiction using 2D column chart.

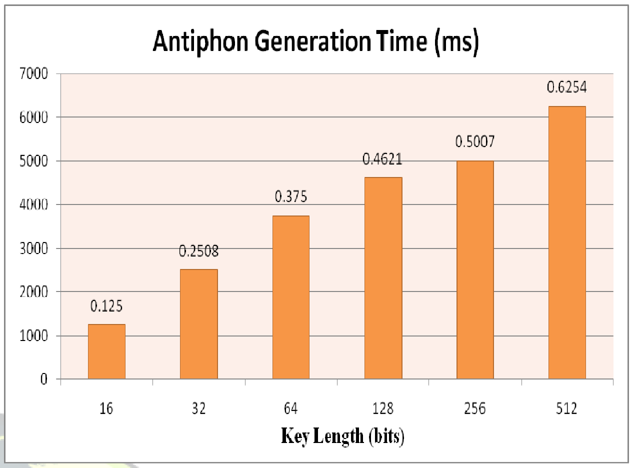

Figure 1. Antiphon Generation Time

The chart is drawn by having the key length measured in bits in x-axis and antiphon generation time in milliseconds in y-axis. The chart shows that key length is directly proportional to the antiphon generation time. As the key length increases so the antiphon generation time also increases.

VI. CONCLUSION

The prime number chosen for computation ensures the security level. If more security is needed then large prime is to be selected. Suppose a prime number of around 20 digits and sealed secrets at least 15 digits each are used for generating keys, then discovering the shared secrets take longer time (even months). After few months, if the key is found using cryptanalysis, it is not useful because the confidential data loses its sensitivity within this time gap. This is the strength of discrete logarithm which computes more secured keys used for enciphering sensitive data shared through cloud. Sending more and more confidential data into cloud cannot be avoided today. Cloud data can be shielded when the key management is done by using the discrete logarithm. Thus key management improved with discrete logarithm ensure safe cloud data storage and transactions.

REFERENCES

[1] Shiang-Feng Tzeng1 , Cheng-Chi Lee2 , Tzu-Chun Lin3," A Novel Key Management Scheme for Dynamic Access Control in a Hierarchy", International Journal of Network Security, Vol.12, No.3, PP.178–180, May 2011.

[2] Shuhua Wu, Kefei Chen, " An Efficient Key-Management Scheme for Hierarchical Access Control in E-Medicine System", Journal of Medical Systems August 2012, Volume 36, Issue 4, pp 2325-2337.
[3] Murali, P, Senthilkumar, G. “Modified Version of Playfair Cipher Using Linear Feedback Shift Register”, Information Management and Engineering, 2009. ICIME '09.
[4] Nabeel, M., Ning Shang,; Bertino, E., "Privacy Preserving Policy-Based Content Sharing in Public Clouds", Knowledge and Data Engineering, IEEE Transactions on (Volume:25 , Issue: 11), September 2012.
[5] Niv Ahituv, Yeheskel Lapid and Seev Neumann, "Processing Encrypted Data", Communications of the ACM, September 1987 Volume 30 Number 9.
[6] Yumnam Kirani Singh, "A SIMPLE, FAST AND SECURE CIPHER", ARPN Journal of Engineering and Applied Sciences, VOL. 6, NO. 10, OCTOBER 2011.
[7] M.Janaki, Dr.M.Ganaga Durga, "A Noval Cryptographic Technique Magnified With Shielded Key Contract" published in "International Journal of Applied Engineering Research", Volume 10, Number 10(2015), PP. 25095-25105, ISSN 0973-4562.
[8] M.Janaki, Dr.M.Ganaga Durga, " Securing Cloud Data Using Cryptography", published in "International Journal of Advance Research in Science and Engineering", Volume 04, Issue S1(01), April 2015, ISSN-2319-8354(E).
[9] M.Janaki, Dr.M.Ganaga Durga, "Enhancing Network Data Security with VIJANA Enchipherment Technique" published in International Journal of Applied Engineering Research, ISSN-0973-4562, Volume 10, No.55 (2015).
[10] M.Janaki, Dr.M.Ganaga Durga, " Preserving sensitive data shared through the network against security threats using cryptography", published in Indian Journal of Education and Information Management, Vol 4 (1), October 2015, ISSN (Online) : 2277 - 5374.