Anonymous Key Generation Technique with Contributory Broadcast Encryption

Ankush V. Ajmire[1], Falesh M. Shelke[2], Prof. Avinash P. Wadhe[3]
[1]G.H. Raisoni college of Engineering and Management, Amravati (M.S.), India
[2]P. R. Pote College of Engineering & Management. Amravati (M.S.), India
[2]Head of Department of (Computer Science and Engineering), G.H. Raisoni college of Engineering and Management, Amravati (M.S.), India

Abstract— Encryption is used in a communication system to secure information in the transmitted messages from anyone other than the well-intended receiver. To perform the encryption and decryption the transmitter and receiver should have matching encryption and decryption keys. For sending safeguard information to group needed broadcast encryption (BE). BE allows a sender to securely broadcast to any subset of members and require a trusted party to distribute decryption keys. Group key agreement (GKA) protocol allows a number of users to establish a common secret channel via open networks. Observing that a major goal of GKA for most applications is to create a confidential channel among group members, but a sender cannot omit any particular member from decrypting the cipher texts. By bridging BE and GKA notion with a hybrid primitive referred to as contributory broadcast encryption (CBE). With these primitives, a group of members move through a common public encryption key while each member having there decryption key. A sender seeing the public group encryption key can limit the decryption to subset of members of sender's choice. A simple way to generate these keys is to use the public key distribution system invented by Diffie and Hellman. That system, however, pass only one pair of communication stations to share a particular pair of encryption and decryption keys. Key distribution sets are used to generate keys and Elliptic Curve Cryptography (ECC) is used for the encryption and decryption of documents; and this going to provide the security for the documents over group communication.

Keywords— *Contributory Broadcast Encryption, Group Key Agreement, Key Distribution Set, Key Distribution Set, Encryption, Decryption.*

*****

I. INTRODUCTION

With the fast advance and pervasive deployment of communication technologies, there is an increasing demand of versatile cryptographic primitives to protect group communications and computation platforms. These new platforms include instant-messaging tools, collaborative computing, mobile ad hoc networks and social networks. These new applications call for cryptographic primitives allowing a sender to securely encrypt to any subset of the users of the services without relying on a fully trusted dealer. Broadcast Encryption (BE) is a well-studied primitive intended for secure group-oriented communications. It allows sender to securely broadcast to any subset of the group members. Nevertheless, a BE system heavily depends on a fully trusted key server who generates secret decryption keys for the group members and can read all the communications to any members.

As a result of the increased popularity with group-oriented applications and protocols, group communication occurs in many different settings from network layer multicasting to application layer. Regardless of the security services, underlying environment are necessary to provide communication privacy and integrity. While peer-to-peer security is a mature and well developed field, the secure group communication remains relatively unexplored. Contrary to a common initial impression, secure group communication is not a simple extension of secure two-party communication. There are two important differences. First, protocol efficiency is of greater concern due to the number of participants and distances among them. The second difference is due to group dynamics. Communication between two-parties can be viewed as a discrete phenomenon. It starts, lasts for a while, and ends. Group communication is more complicated. It starts and the group members leave and join the group and there might not be a well-defined end.

A group key agreement (GKA) is another well-understood cryptographic primitive to secure group oriented communications. A conventional GKA allows a group of members to form a common secret key via open networks. However, whenever a sender wants to send a message to a group, he must first join the group and run a GKAs protocol to share a secret key with the intended members. More recently, and to overcome this limitation, Wu et al. introduced asymmetric group key agreement, in which only a common group public key is negotiated and each group member holds there different decryption key. However, neither conventional symmetric group key agreement nor the newly introduced asymmetric GKA allow the sender to unilaterally exclude any particular member from reading the plain text. Hence, it is essential to find more flexible cryptographic primitives allowing dynamic broadcasts without a fully trusted dealer. Contributory Broadcast Encryption (CBE) primitive, which is a hybrid of GKA and BE.

The model with the CBE primitive and formalize its security definitions. CBE incorporates the underlying ideas of

GKA and BE. A group of members interact via open networks to negotiate a public encryption key while each member holds a different secret decryption key. Using the public encryption key, anyone from the group can encrypt any message to any subset of the group members and only the intended receivers can decrypt. Unlike GKA, CBE allows the sender to exclude some members from reading the cipher texts. Compared to BE, CBE does not need a fully trusted third party to set up the systems. With formalize collusion resistance by defining an attacker who can fully control all the members outside the intended receivers but cannot extract useful information from the cipher text.

The rest of the paper is organized as follows. In Section II, literature survey about the BE and GKA is given. In Section III, we present key generation technique. In section IV, we examine with result and provide the conclusion in Section V.

II. LITERATURE SURVEY

Ankush V. Ajmire, Prof. Avinash P. Wadhe[1] has given the concept about possible way to bridge the GKA and BE notation in which group member can send the secure document to the other with some member to exclude into it by introducing the CBE. So CBE model efficient and secure in the standard model.

C.K. Wong, M. Gouda and S. Lam[2] proposed to address the scalability problem of group key management, author propose the use of key trees in which they investigated three rekeying strategies, key-oriented , group-oriented, user-oriented, and specified join/leave protocols for them. The rekeying protocols and strategies are implemented in a prototype key server author have built. From the measurement results of a large number of experiments, authors conclude that their group key server using any of the three rekeying strategies is scalable to very large groups with frequent leaves and joins. In particular, the average server processing time per leave/join increases linearly with the logarithm of group size.

J.H. Park, H.J. Kim, M.H. Sung and D.H. Lee[3] has proposed two fully collusion-resistant broadcast encryption schemes for stateless receivers. The important way to construct their basic scheme has been to use the algebraic property of Strong Diffie-Hellman tuples. Next extended the general scheme to obtain the chosen cipher text security by applying the hash-based method. By combining general and basic schemes, authors were able to obtain a PKBE scheme for shorter transmissions while preserving user storage cost. they proposed schemes had a drawback of requiring more computation cost in the decryption algorithm, but if they use set differences, this drawback can be somewhat alleviated.

Z. Yu and Y. Guan propose a key management scheme by using deployment knowledge for the wireless sensor networks. In author’s scheme, neighbor nodes can utilize stored secret information more efficiently to generate pairwise keys. Author studied about network connectivity based on geometric random graph model and show how to compute transmission range for achieving the desired connectivity. Simulation results show that author’s scheme outperforms others in terms of resilience against node capture. Meanwhile, it achieves a higher connectivity with a shorter transmission range and a lower memory requirement.

III. KEY GENERATION TECHNIQUE

Key distribution sets (KDS) are used to generate key in which there are different types of combination of character are taken from end user at run time which is related to the document which user are going to share on the intended group with group key agreement.

Following are the few definitions of the KDS which giver complete idea about who the sets are form and key is generated by using the definitions.

Definition 1 = docid-S|!-docname-ddate-R|3419-username
Definition 2 = ddate-username-R|4444-docname-docid-S|%
Definition 3 = docname-S|$-username-R|7424-docid-ddate

docid :- which is the id of the document which is share among the group.
S|{!,%,$,@,#} :- S indicates the special symbol in which we have taken any symbol from the sets of the five symbol.
docname :- is the name of the document at the time of sending taken given by the user.
ddate :- Date on which the document is going to share to the intended group.
R|3419 :- R indicate the four digit random number generated by the system in which system can generate any number from 0000 to 9999 at randomly.
username :- username of the sender which are going to share the document on a particular group or a single user which store their document on server for security purpose.

Random numbers are extremely useful, for example, in generating moves in a game or as test data for computer programs. If one is asked to "pick a number between one and a hundred", the task seems simple enough. But, if you need truly random numbers (each number is equally probable!), and if you want to generate them from a computer, the task is quite tricky as it turns out. Mathematicians have labored over this problem, and have devised robust techniques to generate random numbers. However, computers are deterministic (all actions are predictable at some level!), and, so, generating numbers that are "truly" random is not possible. However, we can get pretty close. Algorithms that generate random numbers, admittedly, provide "pseudorandom" numbers. But, for most purposes this is good enough.

Key generation working:

KDS generate for the every branch of the registered customer. It will select any one KDS set from the KDS set available using random algorithm. Random algorithms again select the any one algorithm from selected set of algorithm and generate key using that algorithm which generate session secrete key and Store session secrete key in Database in encrypted format.

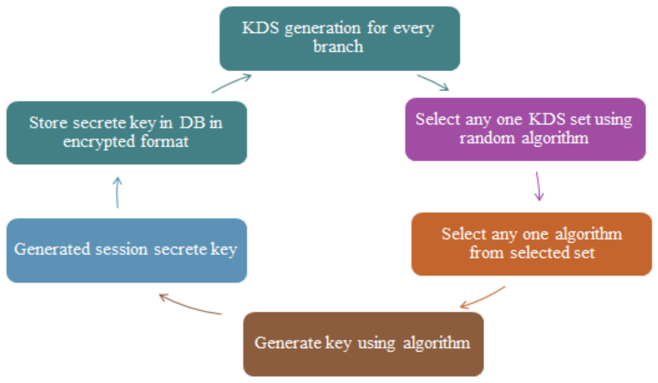

KDS generation for every branch
Select any one KDS set using random algorithm
Select any one algorithm from selected set
Generate key using algorithm
Generated session secrete key
Store secrete key in DB in encrypted format

Figure 1 Key generation working

Elliptic Curve Cryptography (ECC)

Elliptical curve cryptography (ECC) is a public key encryption technique based on elliptic curve theory that can be used to create smaller, faster and more efficient cryptographic keys. ECC generates keys through the properties of the elliptic curve equation instead of the traditional method of generation as the multiplication of very large prime numbers.

The primary benefit promised by ECC is reducing storage, a smaller key size and transmission requirements, i.e. that an elliptic curve group could provide the same level of security Afforded by an RSA-based system with a large modulus and correspondingly larger key. One main advantage of ECC is its small key size. A 160-bit key in ECC is considered to be as secured as 1024-bit key in RSA.

The equation of an elliptic curve is given as,

$$y^2 = x^3 + ax + b$$

Few terms that will be used,

E - Elliptic Curve

P - Point on the curve

n -Maximum limit (This should be a prime number)

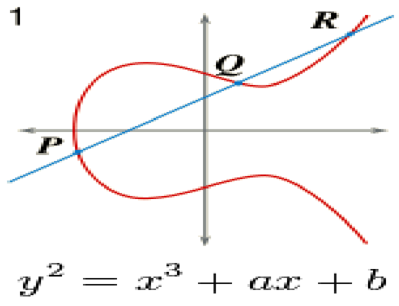


Figure 2 Simple elliptic curve

Key Generation

Key generation is an important part where user has to generate both public key and private key. The sender will be encrypting the message with receiver's public key and the receiver will decrypt its private key.

Now, user have to select a number 'd' within the range of **'**n**'**.

Using the following equation we can generate the public key

$$Q = d * P$$

d = the random number that user have selected within the range of **(**1 to n-1**).**

P is the point on the curve.

Q is the public key.

d is the private key.

Encryption:

Let 'm' be the message that user are sending. user have to represent this message on the curve. This has in-depth implementation details. All the advance research on ECC is done by a company called certicom.

Consider *'m'* has the point *'M'* on the curve *'E'*. Randomly select 'k' from [1 – (n-1)].Two cipher texts will be generated let it be C1 and C2**.**

$$C1 = k*P$$

$$C2 = M + k*Q$$

C1 and C2 will be send.

Decryption:

User has to get back the message 'm' that was send to us,

$$M = C2 – d * C1$$

M is the original message that we have send.

Proof:

How do we get back the message?

$$M = C2 – d * C1$$

'M' can be represented as 'C2 – d * C1'

C2 – d * C1 = (M + k * Q) – d * (k * P)

Where (C2 = M + k * Q and C1 = k * P)

= M + k * d * P – d * k *P

(Canceling out k * d * P)

= M (Original Message)

Upload / Download Working:

In uploading and downloading provides the information about the how the file to be uploaded and already uploaded files how to download it.

Specify the group wise access permission which includes the permission about the group that is GKA which group having permission to upload the files and that access permission store into database so that whenever required then fetches access permission from database. Encrypt the document by using contributory broadcast encryption algorithm document will be share in encrypted format only the user having valid key to decrypt the document can only decrypt that document.

Group will specify and verify the group key and send session secret key to the intended user's mail id after key specification and validation of session secret key user are allow decrypting the document.

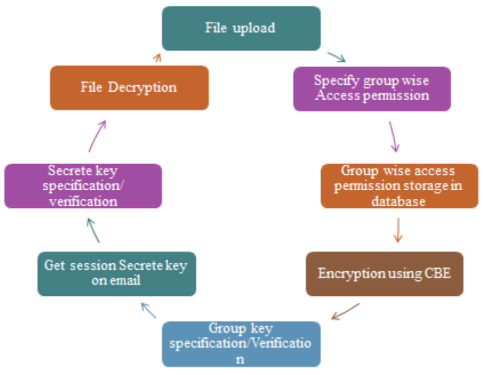
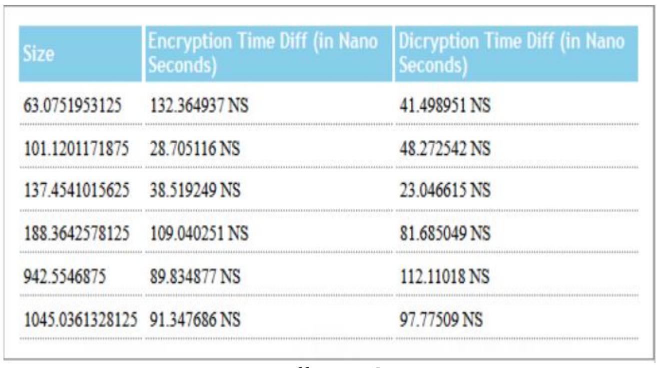

File upload
Specify group wise Access permission
Group wise access permission storage in database
Encryption using CBE
Group key specification/Verificatio n
Get session Secrete key on email
Secrete key specification/ verification
File Decryption

Figure 3 Upload / Download working

IV. RESULT ANALYSIS

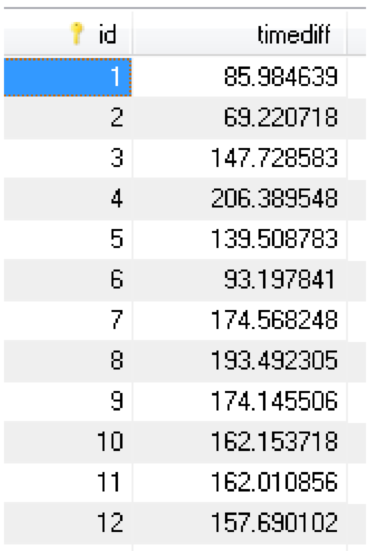
As we have Encrypt and Decrypt the documents on the server we have the experimental analysis about the time duration needed to encrypt/decrypt the document on server.



Size	Encryption Time Diff (in Nano Seconds)	Dicryption Time Diff (in Nano Seconds)
63.0751953125	132.364937 NS	41.498951 NS
101.1201171875	28.705116 NS	48.272542 NS
137.4541015625	38.519249 NS	23.046615 NS
188.3642578125	109.040251 NS	81.685049 NS
942.5546875	89.834877 NS	112.11018 NS
1045.0361328125	91.347686 NS	97.77509 NS

Figure 4 Time Difference Compare Report

Size shows the size of the document file in KB and Encryption and Decryption time in nano seconds.



id	timediff
1	85.984639
2	69.220718
3	147.728583
4	206.389548
5	139.508783
6	93.197841
7	174.568248
8	193.492305
9	174.145506
10	162.153718
11	162.010856
12	157.690102

Figure 5 Group key generation time

As we are generating group key for every group we are needed some time which is shown in table 2 in nano sec.

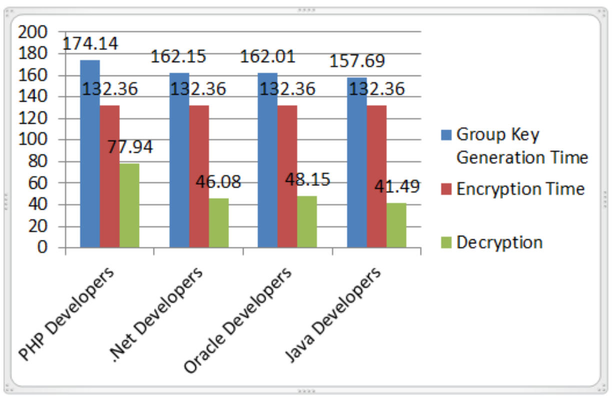


Figure 6 Document of size 63.75 KB in different groups

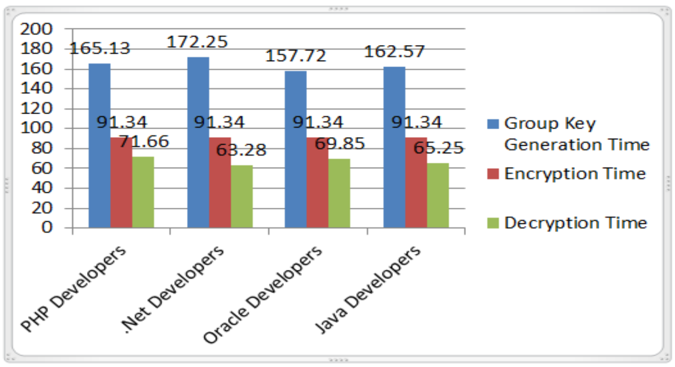


Figure 7 Document of size 1045.03 KB in different groups

In the above Figure 6 and 7 we have plotted graph for the different size of the file in different group like PHP developer, .Net developer, Oracle developer, Java Developer and we examine that at different group, group key generation time and the decryption time varies.

Communication time

For the calculation of the communication time we use following formula.

Group Key Generation Time + Access Time = Communication Time

Encryption Time Decryption Time

Table 1 Communication time

Group Key Generation Time	Access Time		Communication Time
	Encryption Time	Decryption Time	
85.984639	132.364937	41.498951	259.8485
96.220718	28.705116	48.272542	173.1984
147.728583	38.519249	23.046615	209.2944
206.389548	109.040251	81.685049	397.1148
139.508783	2306.524732	195.131093	2641.165
174.145506	209.321878	160.132476	543.5999
193.492305	89.834877	112.11018	395.4374
174.568248	91.347686	97.77509	363.691

Table 1 shows the communication time for the different time to time and by using this we can say that the system requires the communication time varies with varies with group key generation time, access time is the addition of encryption time and decryption time result into communication time.

V. CONCLUSION

The CBE is a primitive which bridges the GKA and BE notions. In CBE, anyone can send secret messages to any subset of the group members, and the system does not require a trusted key server. Neither the change of the sender nor the dynamic choice of the intended receivers requires extra rounds to negotiate group encryption / decryption keys. Following the CBE model, here instantiated an efficient CBE scheme that is secure in the standard model. As a versatile ECC algorithm primitive and KDS, CBE notion opens a new avenue to establish secure broadcast channels and secure numerous emerging distributed computation applications. Our system is going to help to group communication in which there is need to share documents in secure and to intended user.

REFERENCE

[1] Ankush V. Ajmire, Prof. Avinash P. Wadhe,"Review paper on Key Generation Technique With Contributory Broadcast Encryption, " IC-QUEST 2016, 5Th International Conference on Quality Up-gradation in Engineerind, Science & Technology on 12th April 2016.

[2] C.K. Wong, M. Gouda and S. Lam, "Secure Group Communications Using Key Graphs," IEEE/ACM Transactions on Networking, vol. 8, no. 1, pp. 16-30, 2000

[3] J.H. Park, H.J. Kim, M.H. Sung and D.H. Lee, "Public Key Broadcast Encryption Schemes With Shorter Transmissions," IEEE Transactions on Broadcasting, vol. 54, no. 3, pp. 401-411, 2008.

[4] Z. Yu and Y. Guan, "A Key Management Scheme Using Deployment Knowledge for Wireless Sensor Networks," IEEE Transactions Parallel Distributed Systems, vol. 19, no. 10, pp. 1411-1425, 2008.

[5] Q. Wu, B. Qin, L. Zhang, J. Domingo, "Contributory Broadcast Encryption with Efficient Encryption and Short Ciphertexts," IEEE Transactions OnComputer, 2015.

[6] I. Ingemarsson, D.T. Tang and C.K. Wong, "A Conference Key Distribution System," IEEE Transactions on Information Theory, vol. 28, no.5, pp. 714-720, 1982.

[7] Q. Wu, Y. Mu, W. Susilo, B. Qin and J. Domingo-Ferrer, "Asymmetric Group Key Agreement," in Proc. Eurocrypt 2009, 2009, vol. LNCS 5479, Lecture Notes in Computer Science, pp. 153-170.

[8] Q. Wu, B. Qin, L. Zhang, J. Domingo-Ferrer and O. Farras, "Bridging Broadcast Encryption and Group Key Agreement," in Proc. Asiacrypt2011, 2011, vol. LNCS 7073, Lecture Notes in Computer Science, pp. 143-160.

[9] D. H. Phan, D. Pointcheval and M. Strefler, "Decentralized Dynamic Broadcast Encryption," in Proc. SCN 2012, 2011, vol. LNCS 7485, Lecture Notes in Computer Science, pp. 166-183.

[10] M. Steiner, G. Tsudik and M. Waidner, "Key Agreement in Dynamic Peer Groups," IEEE Transactions on Parallel and Distributed Systems, vol. 11, no. 8, pp. 769-780, 2000.

[11] A. Sherman and D. McGrew, "Key Establishment in Large Dynamic Groups Using One-way Function Trees," IEEE Transactions on Software Engineering, vol. 29, no. 5, pp. 444-458, 2003.

[12] Y. Kim, A. Perrig and G. Tsudik, "Tree-Based Group Key Agreement," ACM Transactions on Information System Security, vol. 7, no. 1, pp. 60-96, 2004.

[13] Y. Mao, Y. Sun, M. Wu and K.J.R. Liu, "JET: Dynamic Join-Exit- Tree Amortization and Scheduling for Contributory Key Management," IEEE/ACM Transactions on Networking, vol. 14, no. 5, pp. 1128-1140, 2006.

[14] M. Abdalla, C. Chevalier, M. Manulis and D. Pointcheval, "FlexibleGroup Key Exchange with On-demand Computation of Subgroup Keys," in Proc. Africacrypt 2010, 2010, vol. LNCS 6055, Lecture Notes in Computer Science, pp. 351-368.