DEGREE PROJECT IN COMMUNICATION SYSTEMS, SECOND LEVEL
STOCKHOLM, SWEDEN 2014

Efficient Privacy Preserving Key Management for Public Cloud Networks

ANITHA KATHIRVEL

SIDDHARTH MADAN

KTH ROYAL INSTITUTE OF TECHNOLOGY
INFORMATION AND COMMUNICATION TECHNOLOGY

Efficient Privacy Preserving Key Management for Public Cloud Networks

Anitha Kathirvel

anithak@kth.se

Siddharth Madan

smadan@kth.se

2014-07-17

Master's Thesis

Examiner & academic adviser:
Prof. Gerald Q. Maguire Jr.

KTH Royal Institute of Technology
School of Information and Communication Technology (ICT)
Department of Communication Systems
SE-100 44 Stockholm, Sweden

Abstract

Most applications and documents are stored in a public cloud for storage and management purposes in a cloud computing environment. The major advantages of storing applications and documents in public cloud are lower cost through use of shared computing resources and no upfront infrastructure costs. However, in this case the management of data and other services is insecure. Therefore, security is a major problem in a public cloud as the cloud and the network are open to many other users. In order to provide security, it is necessary for data owners to store their data in the public cloud in a secure way and to use an appropriate access control scheme.

Designing a computation and communication efficient key management scheme to selectively share documents based on fine-grained attribute-based access control policies in a public cloud is a challenging task. There are many existing approaches that encrypt documents prior to storage in the public cloud: These approaches use different keys and a public key cryptographic system to implement attribute-based encryption and/or proxy re-encryption. However, these approaches do not efficiently handle users joining and leaving the system when identity attributes and policies change. Moreover, these approaches require keeping multiple encrypted copies of the same documents, which has a high computational cost or incurs unnecessary storage costs. Therefore, this project focused on the design and development of an efficient key management scheme to allow the data owner to store data in a cloud service in a secure way. Additionally, the proposed approach enables cloud users to access the data stored in a cloud in a secure way.

Many researchers have proposed key management schemes for wired and wireless networks. All of these existing key management schemes differ from the key management schemes proposed in this thesis. First, the key management scheme proposed in this thesis increases *access level* security. Second, the proposed key management scheme minimizes the computational complexity of the cloud users by performing only one mathematical operation to find the new group key that was computed earlier by the data owner. In addition, this proposed key management scheme is suitable for a cloud network. Third, the proposed key distribution and key management scheme utilizes privacy preserving methods, thus preserving the privacy of the user. Finally, a batch key updating algorithm (also called batch rekeying) has been proposed to reduce the number of rekeying operations required for performing batch leave or join operations. The key management scheme proposed in this thesis is designed to reduce the computation and communication complexity *in all but a few cases*, while increasing the security and privacy of the data.

Keywords: Cloud storage, Secure Storage, Key Management scheme, access level security

Sammanfattning

De flesta program och dokument lagras i ett offentligt moln för lagring och hantering ändamål i en molnmiljö. De stora fördelarna med att lagra program och dokument i offentliga moln är lägre kostnad genom användning av delade datorresurser och ingen upfront infrastruktur costs.However, i detta fall hanteringen av data och andra tjänster är osäker. Därför är säkerhet ett stort problem i en offentlig moln som molnet och nätverket är öppna för många andra användare. För att ge trygghet, är det nödvändigt för dataägare att lagra sina data i det offentliga molnet på ett säkert sätt och att använda en lämplig åtkomstkontroll schema.

Utforma en beräkning och kommunikation effektiv nyckelhantering system för att selektivt dela dokument som grundar sig på finkorniga attributbaserad åtkomstkontroll politik i en offentlig moln är en utmanande uppgift. Det finns många befintliga metoder som krypterar dokument före lagring i det offentliga molnet: Dessa metoder använder olika tangenter och en publik nyckel kryptografiskt system för att genomföra attributbaserad kryptering och / eller proxy re-kryptering. Dock har dessa metoder inte effektivt hantera användare som ansluter och lämnar systemet när identitetsattribut och politik förändras. Dessutom är dessa metoder kräver att hålla flera krypterade kopior av samma dokument, som har en hög beräkningskostnad eller ådrar sig onödiga lagringskostnader. Därför fokuserade projektet på design och utveckling av en effektiv nyckelhantering system för att möjliggöra dataägaren att lagra data i en molntjänst på ett säkert sätt. Dessutom, den föreslagna metoden gör det möjligt för molnanvändare att få tillgång till uppgifter lagras i ett cloud på ett säkert sätt.

Många forskare har föreslagit viktiga förvaltningssystem för fasta och trådlösa nätverk. Alla dessa befintliga system ke, skiljer sig från de centrala förvaltningssystemen som föreslås i denna avhandling. Först föreslog nyckelhanteringssystemet i denna avhandling ökar Medverkan nivå säkerhet. För det andra, minimerar den föreslagna nyckelhanteringssystemet beräkningskomplexiteten för molnanvändare genom att utföra endast en matematisk operation för att hitta den nya gruppknapp som tidigare beräknades av dataägaren. Dessutom är denna föreslagna nyckelhanteringsschema lämpligt för ett moln nätverk. För det tredje, den föreslagna nyckeldistribution och nyckelhantering systemet utnyttjar integritets bevara metoder och därmed skydda privatlivet för användaren. Slutligen har ett parti viktig uppdatering algoritm (även kallad batch nya nycklar) föreslagits för att minska antalet Ny serieläggning av operationer som krävs för att utföra batch ledighet eller gå med i verksamheten. Nyckelhanteringssystemet som föreslås i denna avhandling är utformad för att minska beräknings-och kommunikations komplexitet i alla utom ett fåtal fall, och samtidigt öka säkerheten och integriteten av uppgifterna.

Nyckelord: Cloud Storage, säker förvaring, nyckelhantering schema, åtkomstnivå säkerhet

Acknowledgments

We thank our parents for their unconditional and unwavering support. We would like to offer our sincere thanks to Prof. Gerald Q. Maguire Jr., whose guidance has made this thesis possible. We would also like to thank Asst. Prof. Vijayakumar Pandi, Anna University whose valuable advice has been immensely helpful. We would like to thank our good friend, Anand Kannan, whose advice and tips have been very helpful. We extend our thanks to friends and our loved ones for their support.

Stockholm, July 2014.
Anitha Kathirvel
Siddharth Madan

Table of contents

List of Figures

List of Tables

List of acronyms and abbreviations

ASAC	Action Status based Access Control
ACV	Access Control Vector
BGKM	Broadcast Group Key Management
CSP	Cloud Service Provider
DVD	Dynamic Virtual Digraph
ECC	Elliptic Curve Cryptography
ELK	Efficient Large-group Key distribution
FEC	Forward Error Correction
GC	Group controller
GF	Galois field
GK	Group Key
GNAVE	Greedy Node capture Approximation using Vulnerability Evaluation
HECC	Hyper Elliptic Curve Cryptography
HWN	Hierarchical Wireless sensor Network
IaaS	Infrastructure as a Service
LAKE	LAyer Key Establishment
LBS	Location Based Service
LFU	Least Frequently Used
LLK	Link Layer Key
MDS	Maximum Distance Separable
NSBHO	Non-Split Balancing High-Order
OBS	On-Bound Selection
OCBE	Oblivious Commitment Based Envelope
P2P	Peer to Peer
PaaS	Platform as a Service
PKI	Public Key Infrastructure
R-CB	Revised Call-Back
RBAC	Role based access control
SaaS	Software as a Service
SB-PER	Server-Based Poll-Each-Read
SGk	Sub Group key
SK	Session Key
SPATE	Small group PKI (Public Key Infrastructure) – less Authenticated Trust Establishment
TLK	Transport Layer Key
TPA	Third Party Auditor
VC	Virtual Circuit
VCR	Video Cassette Recorder
VOD	Video On Demand
WSN	Wireless Sensor Network

1 Introduction

Today cloud computing is used by many large and small organizations, either directly or indirectly. A cloud service enables users to share data in an economical and easy way. Cloud services can be divided into three categories: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). IaaS provides the user with virtual infrastructures, such as servers, routers, switches, and storage. PaaS provides the user with development environments where the user can create and run their own applications. SaaS provides the user with access to existing applications that operate in the cloud. This thesis is concerned with IaaS. In this thesis project, we assume that multicast communication is available between the user and all of the machines in the cloud infrastructure as 75% of the proposed solution is based on group communication and the remaining 25% deals with preserving privacy as discussed in section 4.

Similar to cloud services, cloud computing deployment models can be divided into different types: public cloud, private cloud, hybrid cloud, and community cloud. A public cloud infrastructure is available to the public and is owned by a cloud service provider (CSP). In the public cloud deployment model the CSP provides services and infrastructure to a number of clients. A private cloud infrastructure is operated *exclusively* for a single organization. In a private cloud, the CSP provides specific cloud services only to their client and no other clients are allowed. The hybrid cloud deployment model is a combination of private and public clouds. A hybrid cloud helps businesses to take advantage of secure applications and data, hosted on a private cloud; while still enjoying the cost benefits of storing some of their shared data and running some of their applications in a public cloud. A hybrid cloud can dynamically migrate workloads between public and private hosts without causing any inconvenience to the users. In the community cloud deployment model the cloud infrastructure is shared by several organizations with the same policies, thus reducing costs as compared to a private cloud, as a larger community shares the cloud, while providing the security advantages of a private cloud.

The cloud deployment model considered in this research is a public cloud. In a public cloud, services are available to the public. These services are controlled by the data owner and by the CSP. Google is an example of a public cloud provider. Services can be provided to clients free of charge, pay-per-user, or via a pay per usage model. These different payment models greatly reduce the capital expenditure of a company that wishes to offer a service. In a public cloud, since the public can access the service, many users can access the data located in the CSP's site. However, CSPs cannot be expected to protect the confidentiality of the data placed in their cloud by a data owner. For this reason, data privacy and security issues are major concerns for many organizations that wish to utilize services provided by a public cloud. To provide confidentiality while selectively sharing documents with a group of users in public clouds, an access control mechanism has to be implemented by the service provider. In this thesis, an efficient fine-grained encryption based access control scheme is proposed for storing documents in an untrusted public cloud. Users are allowed to access the documents for which they have received access rights from the data owner. Access is given to cloud users based on their identity attributes. These attributes are submitted at the time of user registration. To preserve the privacy and security of the documents and the users, the identity of the user is also protected. In the proposed attributes based access control mechanism, a user is able to decrypt the documents *if and only if* this user's identity attributes satisfy the data owner's access control policies. Moreover, the data owner and the CSP learn nothing about the user's identity attributes. Thus, hiding the user's identity attributes protects the privacy of the data accessed by each cloud user. In order to implement the proposed access control mechanism, a computationally efficient key management scheme has been developed in this thesis project.

1.1 Access Control Schemes for Security

Access control mechanisms restrict unauthorized users from accessing data. The widespread adoption of Internet standards, protocols, and policies for information exchange is laying a foundation for flexible granularity in information and communication services. There are many previous works concerning access control based security mechanisms for wired and wireless networks. Among them, a Java based system to address the security issues of access control was developed by Bertino, Shang, and Wagstaff [1] based on a policy designed for XML documents. This system supports the specification of policies at various granularities and considers the trust level of users when enforcing access control.

A role based access control (RBAC) model consists of four basic components: a set of users, a set of roles, a set of permissions, and a set of sessions. In a RBAC system, the user can be either a system user or an individual user. An individual working on behalf of an organization is assigned a role. In this role, the user is assigned a set of privileges necessary to carry out their job function within this particular organization. Moreover, the privileges given to the user are a function of the role to be played by this user. When a user logs into a system, he/she establishes a new session using a key. During this session, the user can utilize the privileges of his or her role to perform various data manipulation activities on database tables and objects created for the organization and for the users based on their access rights. Roles have several advantages with regard to access control as roles represent an organizational function. A role based access control model can directly support an organization's security policy. The RBAC model is widely used for access control management, both in closed and open systems (James B.D. Joshi et al. [2]) Where authorizations are specified with respect to roles, rather than with respect to individual users. Each user can have more than one privilege since they can have more than one role at a given time. Based on these roles, privileges are assigned to each of the roles. This is desirable since managing a small number of roles is much more efficient than managing a large number of individual users. RBAC has been investigated by many researchers (such as Bertino, Bonatti, and Ferrari [3], Joshi et al. [2], and Naranjo et al. [4]). Although RBAC has been thoroughly explored, there are still significant application requirements, which are not addressed by current RBAC models. To overcome this, a generalization of the RBAC model, called the Action Status based Access Control (ASAC), was proposed by Barker [5].

A key feature of the ASAC model is that a decision regarding an agent's request to access resources is determined by considering the agent's ascribed status. In such a system, the agent's action status along with additional relevant conditions is considered before processing the agent's access request. An agent's attributed status together with the agent's action status gives a measure of the agent's overall status level. The agent's status level is used as the basis for determining which actions will be authorized and thus the agent's status level is used in rendering a decision about the agent's access request.

Another important criterion for security in distributed systems is location constraints. Therefore, an access control system not only must consider *temporal* constraints, but also considers *spatial* constraints. In order to cope with both temporal and spatial constraints, the conventional RBAC model must be extended to specify temporal and spatial restrictions on permissions assigned to roles.

Group access control can be achieved by encrypting the message (document) using an encryption key with a suitable size. This key is dynamically generated for each communication session. This dynamically generated key uses an effective key management scheme, so that the Session Key (SK) or Group Key (GK) are shared by all legitimate users of a group to access a common set of data stored by the cloud server. This sharing of keys is necessary since group membership in a multicast group is likely to change dynamically whenever a new user joins the group or an existing member leaves the group. Poovendran and Baras [6] have proposed that the encryption keys be updated in order to prevent those leaving or joining from accessing data or messages from future or prior communications. The issues of establishing and updating the GKs have been addressed by various Group Key Management schemes present in the literature (Li, Poovendran, and McGrewb [7], Kim, Perrig, and Tsudik [8], Drira, Seba, and Kheddouci [9],and Naranjo et al. [4]). Compared with all of the existing key management schemes, the key management scheme proposed in this research differs, as a centralized authority or key server in the proposed scheme does not generate the keys. Instead, the data owner generates private keys for each user. Based upon these keys it computes a common public key, which is used as a GK. After generating a common GK, the data owner encrypts the documents using the GK and stores the encrypted documents in the cloud. Moreover, in this research, the privacy of each user is preserved by protecting their identity from both the data owner and CSP.

1.2 Key Management Schemes

The process of generating, distributing, and maintaining keys are taken care by key management schemes. There are many key management schemes in the literature (Bertino, Shang, and Wagstaff [1], Jeong et al. [10], and Kim and Choi [11][12]). There are two types of key management schemes: centralized and distributed key management schemes. Currently these schemes provide security for multicast communication. In the centralized scheme, a trusted third party is used to control the group management activities. These activities include member registration, key generation, key distribution, and group management. Moreover, the trusted third party (called a group controller (GC)) is responsible for interacting with the group members and controlling them via a centralized key management scheme. Alternatively, the keys in a distributed key management scheme are computed and maintained in coordination with the group members. Distributed key management schemes are divided into two types: fully distributed key management and partially distributed key management schemes. In a fully distributed key management scheme, the users themselves contribute to generation and distribution of the key, which helps to maintain secrecy and group membership, while securing the group's communication. In a partially distributed key management scheme, both the users and the GC are responsible for generating and maintaining the keys and group membership. In such a partially distributed scenario, group members get some information from the GC. The group members use this information to maintain secrecy and their group membership. The key management scheme developed in this thesis project is a centralized key management scheme that operates between the data owner and the cloud users.

The provision of an access control facility using a centralized key management system is a challenging task. This is due to the fact that key generation and distribution are more complex when messages are distributed to a group of users from the cloud's servers, as users may dynamically join or leave the multicast group. To support this dynamic and secure group communication, it is necessary to allow members to join or depart from the service at any time. When a new member joins the service, it is the responsibility of the data owner to prevent this new member from having access to prior data in order to provide backward secrecy for the earlier secure group communication. Similarly, when an existing group member leaves any group, such a member should no longer have access to data - as this access should only be available to the current group members in order to achieve forward secrecy. In order to handle the issues of forward and backward secrecy, the keys are updated whenever a member joins or leaves the service. The data owner is responsible for generating a new GK after members join or leave. After a member joins the group or leaves the group, the data owner generates a new GK and securely distributes this GK to the group's members. As a result, when a group membership change, the data owner computes a new GK and the access control vector is updated by changing the public and private information available to the data owner.

After computing the access control vector, the data owner multicasts this access control vector to the current group of cloud users and each cloud user computes the new GK. The old user cannot find the group key since his/her private key is not used when sending the new group key value to the remaining users. Thus, changing the group key securely after a member joins or leaves takes only limited computation (by each user) and has low communication complexity since it can exploit multicast communication. When a user leaves the group they are excluded from all future communications and thus will not be able to compute the new Group Key.

The proposed scheme provides both forward and backward secrecy, hence the former group members cannot receive future communication and a new user cannot access previous group multicasts.

1.2.1 Key Generation

A key generation process is responsible for generating the random private keys assigned to the registered cloud users. This process also generates and computes GKs with respect to these private keys under a common subgroup. An important issue in maintaining the integrity in communication is to propose techniques for generating a GK by the data owner and enabling the group members to independently derive this GK without revealing the identity of the individual members of the group. There are two types of techniques that are used for GK generation. In the first method, users generate their own secret keys from which they compute a common GK, which will act as a public key for a group of members. This method is a distributed key management scheme. In the second method, a trusted third party (in this case the data owner) generates the GK and distributes it to the group members in a secure way. This second method is a centralized key management scheme. In both schemes, several computations are necessary to compute the subgroup and GKs. Moreover, both schemes need to store the public parameters and various key values used for computing the GK. In order to overcome the challenges of computational complexity and minimize memory requirements, a new key management scheme with reduced computational cost and memory requirements is needed. Therefore, this research proposes a new computationally efficient technique that uses simple mathematical functions and an optimal number of multiplications and additions in order to efficiently generate a GK. Operations such as multiplication, division, and exponential operations are expensive; however, most key management schemes primarily use multiplication, multiplicative inverse, and exponential operations and hence they require more computation. The proposed scheme minimizes this complexity by using only multiplication and addition.

1.2.2 Key Distribution

A key distribution scheme for secure group communication is responsible for distributing the private keys and GKs to the registered users in the cloud network. GKs can be distributed either by the data owner to the participating members or the members themselves will distribute the keys generated by them as necessary for computing the GK. In a centralized key management scheme, the GK is distributed by the data owner, whereas in the distributed approach any one of the group members can distribute the GK. This project focuses on centralized key distribution schemes, since the major security challenges lie mostly in the design of a new effective centralized key distribution scheme. Although many designs have been proposed by various researchers (Li et al. [13], Zhang et al [14], Ng et al. [15], and Lihao and Huang [16]), they all incur a lot of communication overhead when members join and leave a group. Moreover, all of the existing key management schemes are unsuitable for providing a group oriented service in a cloud network. Hence, a new and effective centralized key distribution scheme is needed that is suitable for providing a group oriented service in a cloud, while reducing computational complexity.

1.2.3 Key Recovery

In a centralized key management scheme, secure multicast key recovery process is used by group members to construct the original GK computed by the data owner. In contrast, in distributed GK management, the key recovery process is used by group members to individually compute the GK based on values received from other group members. In both of these schemes, the members of the group should perform a minimum number of mathematical operations to recover the newly generated or updated GK. Moreover, the key recovery process should minimize the number of parameters needed for recovering the common GK whenever there is a change in the group's membership.

1.3 Privacy Preserving System

In public cloud networks, the confidentiality of the data and the privacy of the users are not protected. Many privacy preserving techniques exist to preserve the privacy of the user and to protect the identity attributes of the user. However, they are not efficient in protecting the privacy of the user's identity attributes.

In order to protect the privacy of the users' identity attributes two cryptographic techniques, are used in the approach proposed in this thesis: Pedersen commitment and Oblivious Commitment Based Envelope (OCBE) protocols. Using these techniques, the user can decrypt the data sent by the data owner *if and only if* the user satisfies the access control policy. The data owner and the cloud service provider do not know anything about the user's identity attributes. Thus, the privacy of the user's identity attributes is preserved. Using these two techniques both the confidentiality of the data and the user's privacy are protected.

1.4 Scope of the Thesis

Encryption and key management techniques are necessary for ensuring data confidentiality. Since unauthorized users do not possess the GK for a session, they cannot decrypt the information exchanged in the current group communication session. For IP multicast security, several key management schemes have been proposed (Harney and Muckenhirn [17], Harkins and Carrel [18], and Maughan and Schneider [19]). However, all of these static key management schemes do not provide a solution for key change upon membership changes to provide effective security for multicast communication. Moreover, the existing key management schemes available in the literature (Li, Koutsopoulos, and Poovendran [20], Lu [21], and Jeong et al. [10]) only consider access control issues for a *single* multicast session. However, it is necessary to provide a facility for updating keys with respect to dynamic changes in membership and to provide all of the legitimate group members with the necessary level of access privileges. This helps to maintain forward and backward secrecy in the group communication. Moreover, a cloud user's identity should also be preserved in order to protect the privacy of each user's access in a public cloud.

1.5 Objectives and Assumptions

The major objectives of this thesis project are:

- To propose a group key management scheme in order to enhance the security of group communication performed in public clouds.
- To propose a privacy preserving system that hides users' identity from both the CSP and other cloud users.
- To propose a technique that reduces the memory requirements of group members by allowing group members to store less information – this information should be the minimum needed for computing the GK.
- To propose a technique for reducing the communication cost of the key updating process when batch rekeying operations are performed by the data owner.

The major assumptions made in this thesis are:

- The system should support several thousand users.
- The data owner keeps all users' private keys secret and each user keeps their own private keys secret.
- Each user's joining and leaving behaviour is independent of other users.
- There is no network delay (i.e., we have ignored delay in all of our analysis).

2 Background

This section reviews the state of the art regarding security architectures for cloud provisioning models with respect to privacy preserving methods, access control methods, and key management methodologies. Researchers have worked on centralized multicast key management schemes, batch rekeying methods, and access control mechanisms for secure group communication. In addition, there are methods that preserve user's privacy. However, most of these schemes consume considerable computation time and memory. In addition, most existing schemes have high rekeying cost (in terms of communication complexity) for performing batch rekeying operations. In order to address these issues, it is necessary to analyse the merits and limitations of existing key distribution, key management, and privacy preserving schemes.

2.1 Centralized Multicast Key Management

Agee, Wallner, and Harder [22] discussed the difficult problem of key management for multicast communication sessions. Their paper focuses on two main areas of concern with respect to key management: initializing the multicast group with a common key and rekeying the multicast group. Rekeying may be necessary upon the compromise of a user or for other reasons (e.g., periodic rekeying). Although, these authors achieved efficiency in terms of rekeying cost and memory, their solution must be enhanced to provide better performance to avoid delays in packet transmission.

Steiner, Tsudik, and Waidner [23] proposed a new key management protocol based on Diffie-Hellman key exchange. This protocol achieves secure and efficient key agreement in the context of dynamic peer groups that are relatively small and non-hierarchical. However, their protocol is only efficient for small groups, thus it is unsuitable for large groups.

Wong, Gouda, and Lam [24] presented a novel solution to the scalability problem of group or multicast key management. They introduced the concept of key graphs for specifying secure groups. In addition, they presented three strategies for securely distributing rekeying messages after a join or leave and they proposed new protocols for joining and leaving a secure group. The rekeying strategies and join & leave protocols were implemented in a prototype key server that they built. Poovendran and Baras [6]showed that rooted-tree-based secure multicast key distribution schemes can be useful for collision avoidance, while reducing memory requirements.

Li, Poovendran, and Berenstein [25] studied the problem of distributing cryptographic keys to a secure multicast group with a single sender and multiple receivers. They showed that the problem of designing a key distribution model with specific communication overhead could be posed as a constraint optimization problem. Using this formulation, they showed how to minimize the number of keys to store by the GC. An explicit algorithm was designed with a given key update communication budget. The main advantage of their work is that they provide security for one-to-many communications. However, their solution to the constraint optimization problem itself is complex.

Trappe et al. [26] presented two modes of conveyance for transmitting rekeying messages. By embedding the keying information in the multimedia content, the key updating messages associated with secure multicast key management schemes is hidden in the data. They used this approach in conjunction with encryption to protect the data from unauthorized access. The main advantage of these proposals is a reduction in memory requirements to some extent.

Sherman and McGrew [27] presented and analysed a new practical centralized hierarchical algorithm for establishing shared cryptographic keys for large, dynamically changing groups. Their algorithm is a novel application of One-way Function Trees, taking a bottom-up approach with the option of member contributions to the entropy of the common communications key. Unlike previously proposed solutions based on information theory and hybrid approaches, their one-way function algorithm's communication, computation, and storage requirements scale logarithmically with group size, both for adding or evicting operations (i.e., join and leave operations). A design for a storage efficient secure multicast key management scheme based on one-way function trees for a pre-specified key update communication overhead was proposed by Bruhadeshwar, Kulkarni, and Liu [45]. Fei et al. [28] and Calandriello et al. [29] designed a Video-Cassette-Recorder (VCR) friendly broadcast series and proposed an active buffer management technique to implement the functionality of providing interactive services in broadcast Video on Demand (VoD) systems. They each showed that their scheme could implement VCR actions through buffering with a high probability to support a wide range of user interaction levels.

A secure communication-efficient key agreement protocol based on bilinear pairings in *ad hoc* networks was proposed by Shi and He [30]. In their protocol, the dynamics of the network is considered and the one-hop assumption is weakened by employing hierarchical routing techniques to ensure that the logical model of key agreement coincides with the actual topology of the networks.

Wu, Chiu, and Chieu [31] proposed an Elliptic Curve Cryptography (ECC) pairing-wise, user-friendly remote timestamp-based password authentication scheme with smart cards. Its extended version is a nonce-based password authentication scheme. These proposed schemes do not use any password file or verification table to authenticate the users. They also analysed some possible attacks. Their proposed scheme eliminates the drawback of a traditional ID-based scheme of assigning lengthy passwords. The remote distributed hosts do *not* need to know the secret key in order to authenticate the user. This enhances the flexibility of the proposed authentication scheme. In addition, the scheme inherits the merits of ECC, i.e., small key size and high security. This solution is especially well suited to smart card applications, mobile communications, and other remote distributed systems. Since ECC uses a smaller key size while offer high security, its computation complexity lower for a given level of security than other types of cryptography.

Wang and Laih [32] proposed an efficient time-bounded scheme based on a technique called merging. The idea behind merging is to consider primitive keys instead of hierarchies. It is conceptually similar to the compression used in source coding. Through this technique, it is feasible to combine multiple keys into an aggregate key, thus greatly reducing communication and storage requirements. However, the computation time of this approach is high.

Purandare and Guha [33] introduced a novel framework for peer-to-peer (P2P) media streaming, that uses an alliance based peering scheme to solve some of the existing problems in chunk based P2P media streaming. In particular, their main contributions are a reduction in buffering time and scalability.

Je et al. [34] proposed a computation-and-storage-efficient key tree structure, and a key tree management protocol for secure multicast communication. By considering the resource information of each group member's device, this protocol manages the key tree structure to maximize the efficiency of the computation and storage costs, while minimizing the cost of communication.

Lihao and Huang [16] proposed a new multicast key distribution scheme in which the computational complexity is reduced by using Maximum Distance Separable (MDS) codes to dynamically distribute a multicast key.

Ramkumar [35] proposed three techniques for key distribution which not only reduces computational complexity, but also reduce the bandwidth and storage requirements to some extent. Moreover, they have provided a security model to prevent message-injection attacks.

López and Casado [36] presented a new algorithm for key management and described three applications of their algorithm to security and privacy. The first is a method for controlling the disclosure of discrete logarithm-based public keys. This method can be used to privately deliver a public key to a set of recipients with only one multicast. The second method is an authentication technique that can be used in scenarios where a public-key infrastructure is not available. The third application uses the Extended Euclidean algorithm, a zero-knowledge proof. Moreover, it reduces the number of messages exchanged between two nodes in the types of applications mentioned above (i.e., smart card and mobile applications).

The main limitations of these existing schemes are the computational complexity involved in rekeying operations leading to a decrease in performance. In addition, the memory requirements are high in most of these existing schemes. Compared with all of these existing schemes, the key management scheme proposed in this thesis to provide forward and backward secrecy is more efficient in terms of computation, communication, and storage.

2.2 Batch Rekeying

Waldvogel et al. [37] presented the Versa Key middleware framework for secure multicasting. The core of their framework consists of three approaches (each with different properties), but relying on the same basic principle. All these approaches organize the space of keys that will eventually be assigned to group members in a unique way, *without* generating keys before they are needed. Only when new group keys need to be established are the keys generated and then these keys are only distributed to those members of the group affected by a change. Their organization of the key space assures that all operations on groups are executed with a complexity of O(log n) or less, where n is the size of the group and complexity is measured in terms of the number of messages exchanged and the number of cryptographic operations to be performed by any of the participants.

Li et al. [13] presented a new technique for multicast batch rekeying. In order to keep the tree balanced all of the time this technique reallocates the tree nodes. Penrig, Song, and Tygar [38] introduced Efficient Large-group Key distribution (ELK), an efficient, scalable, secure method for distributing group keys. This technique has widespread application, such as access control in streaming multimedia broadcasts. Zhang et al. [14] presented a design and implementation of a new key management protocol that is scalable and reliable with respect to performance. Their protocol uses key trees for secure groups and periodic batch rekeying. At the beginning of each rekey interval, the key server sends a rekey message to all users consisting of encrypted new keys (called “encryptions”) carried in a sequence of packets. They presented a scheme for identifying keys, encryptions, users, and a key assignment algorithm that ensures that all the encryptions needed by a user are in the same packet. Their protocol provides reliable delivery of new keys to all users eventually. It also attempts to deliver new keys to all users with a high probability by the end of the rekey interval. For each rekey message, the protocol runs in two steps: a multicast step followed by a unicast step. A proactive forward error correction (FEC) multicast mechanism reduces delivery latency.

Onen and Molva [39] proposed a new algorithm to separately regroup members into two categories: volatile and permanent members. A threshold value w sets the time at which a volatile member is considered permanent. In order to offer higher reliability to permanent members, the key server adjusts the rekeying intervals Tv and Tp of the two sets after computing their corresponding rekeying cost. The proposed protocol suits applications where there exists a strong requirement for backward and forward secrecy and where clients pay only for the amount of time they were present in the multicast group. Goshi and Ladner [40] proposed a height-balanced 2-3 tree (B-tree of order m=3) and found that it has the best performance among the balancing strategies tested. However, balancing a B-tree as proposed by Goshi and Ladner [40] after a member joins involves splitting oversized tree nodes and results in a worst-case rekeying cost. Haibin Lu [21] developed a Non-Split Balancing High-Order (NSBHO) tree. Unlike the B-tree scheme, Lu’s NSBHO tree does not use node splitting to balance the tree. Experiments confirmed that the NSBHO-tree is superior to the B-tree in terms of the worst-case rekeying performance. In addition, it has better average-case rekeying performance.

Wang et al. [15] presented two merging algorithms that are suitable for batch join events. To handle batch depart requests, they have extended these two merging algorithms into a batch balanced algorithm. All three algorithms try to minimize the difference in height of the key tree *without* adding extra network communication costs. However, all of these algorithms require the GC to update the affected members’ node position by using update messages. By minimizing the differences in height, they minimize the number of key stores and decryptions needed by each member. This is critical for terminals with limited computation and storage (such as low to medium range devices that are not equipped with high power hardware or optimized software). Furthermore, reducing the number of decryptions helps to reduce the energy consumption, which leads to longer battery powered operating time.

Cho, Chen, and Eltoweissy [41] proposed an analytical model to address the issue of how often batch rekeying should be performed. They proposed threshold-based batch rekeying schemes and demonstrated that an optimal rekeying interval exists for each scheme. They compared these schemes to identify the best scheme to minimize the communication cost of rekeying while satisfying application requirements by using a set of parameter values characterizing the operational and environmental conditions of the system. Lee et al. [42] proposed an algorithm that finds a candidate tree structure. As an another approach, in order to determine an optimal tree structure, they proposed a new cost metric that considers member dynamics and the average number of rekeying messages.

A novel hardware or software architecture was proposed by Shoufan and Huss [43], which optimizes the rekeying performance not only by minimizing the number of cryptographic operations, but also reduces the execution time of these operations by performing digital signing with the aid of hardware acceleration. In their system, all help-keys are generated, managed, and stored in hardware, which they claim enhances the system's security. To retain flexibility, control-intensive tasks, such as tree management, are performed as software functions on an embedded processor. Their rekeying processor was designed based on a comprehensive security analysis with the aid of a novel illustration of security threats, requirements, and technical solutions. A performance measurement of a prototype implementation shows that the rekeying processor can join and disjoin members much faster than software solutions, in addition to supporting much larger groups.

Hai-Tao et al. [44] proposed a m-dimensional space geometry sphere rekeying scheme, in which the GC generates a parent's key using its child's key when members join or leave, thus reducing the communication cost of the GC for rekeying. When many of members join or leave at the same time, their batch rekeying scheme improves rekeying performance. Their simulations showed that their scheme decreases communication cost and storage cost, increases the efficiency of new group key distribution, and improves the stability of multicast rekeying.

Bruhadeshwar, Kulkarni, and Liu [45] and Bruhadeshwar and Kulkarni [46] presented a family of algorithms for effective key management that reduces the number of keys that are used by users and the time required for rekeying due to the revocation of multiple users. They showed that their algorithms reduced the cost of rekeying by 43-79 % when compared with previous solutions.

Compared with these existing rekeying algorithms, the batch rekeying algorithm proposed in this thesis is more efficient in terms of rekeying cost and computational complexity.

2.3 Key Distribution in Wireless Networks

Hu and Chen [47] devised an adaptive information dissemination mechanism by exploiting data broadcasting to support the dissemination of static and dynamic information services simultaneously. In their design, both static and dynamic information services are subsumed as service groups, i.e., the building blocks use a uniform representation of structure and the group's popularity; thus, the conventional scenario becomes a special case of their framework. Furthermore, in order to tolerate broadcast traffic dynamics, they designed an online loan based slot allocation and feedback control technique to deal with the adaptation of the service group classification, bandwidth allocation, and broadcast schedule to avoid performance degradation. An experimental study showed that their proposed adaptive information dissemination mechanism associated with the online loan based feedback control was able to achieve a substantial reduction in the amount of message traffic for dynamic information dissemination in wireless networks.

Westhoff, Girao, and Acharya [48] presented an approach that conceals sensed data end-to-end, while still providing efficient and flexible in-network data aggregation. They applied a particular class of encryption transformations and discussed techniques for computing the aggregation functions "average" and "movement detection". They showed that their approach is feasible for the class of "going down" routing protocols. They considered the risk of corrupted sensor nodes by proposing a key predistribution algorithm that limits an attacker's gain and showed how key predistribution and a key-ID sensitive "going down" routing protocol helped to increase the robustness and reliability of the connected backbone.

Sakiyama et al. [49] presented a reconfigurable curve-based cryptographic processor that accelerates scalar multiplication of ECC and Hyper Elliptic Curve Cryptography (HECC) of genus 2 over the Galois field (GF) GF(2^n).

Sultana, Choi, and Huh [50] proposed and analysed a scalable and efficient cluster based group key management protocol by introducing an identity based infrastructure for secure communication in mobile wireless sensor networks. To ensure scalability and dynamic re-configurability, their system takes a cluster based approach by which group members are broken into clusters and leaders of clusters securely communicate with each other in order to agree upon a compromised group key whenever a member joins or leaves. Through analysis, they have shown that their protocol has a high probability to be resilient for secure communication among mobile nodes. Finally, they clarified that their proposed scheme is efficient for secure positioning of nodes in a wireless sensor network.

Chen and Xiao [51] proposed a cache replacement policy, called On-Bound Selection (OBS), that uses both data access and update information. OBS was inspired by an analytical analysis of Server-Based Poll-Each-Read (SB-PER) and Revised Call-Back (R-CB) policies. OBS provides an upper bound for effective hit ratio and a lower bound for communication cost. Their proposed scheme was evaluated and compared with a Least Frequently Used (LFU) replacement policy through extensive simulations.

Zhou and Fang [52] proposed a novel key establishment scheme for sensor networks. They use t-degree tri-variate symmetric polynomial to establish both Transport Layer Keys (TLKs) and Link Layer Keys (LLKs) between sensor nodes. Each node directly calculates TLKs and LLKs with logically neighbouring nodes which in turn negotiate indirect TLKs and LLKs with other nodes. Pairs of nodes can directly negotiate a TLK on demand or utilizing an intermediate node to assist them. They state that their two-LAyer Key Establishment (LAKE) scheme is more secure under a compromised node attack and has much lower memory cost than conventional solutions. Additionally, LAKE is energy efficient as each node has direct LLKs with its neighbours while minimizing the amount of energy expended establishing indirect LLKs with neighbours by using multihop routing.[52]

Xu, He, and Harn [53] improved Chien's [54] scheme for time-bound hierarchical key assignment *without* public key cryptography. Their scheme is resistant to Yi and Ye [55] three-party collusion attack, while being as efficient as Chein's.

Lin and Shieh [56] proposed a new lightweight, pollution-attack resistant authentication scheme for multicast communication that generates evidence on receivers for validating on a fast per-packet basis. This approach is effective for preventing pollution attacks and it provides better performance when compared with other existing works.

Jeong et al. [10] proposed a key aggregation technique for facilitating intermediate nodes aggregating data more safely. This protocol is suitable for providing security to multi-tier network architectures and to establish secure sessions between sensor nodes and gateways.

Tague et al. [57][58] and Tsang et al. [59] proposed and evaluated new metrics for quantifying the availability of network services in the presence of malicious nodes and attackers. They proposed a Greedy Node capture Approximation using Vulnerability Evaluation (GNAVE) algorithm to identify compromised users in the network based on a set of control channels that are jammed. They evaluated the estimation error using the GNAVE algorithm based on false alarm rate as well as miss rates in the identification problem. They discussed various design trade-offs between robustness to control channel jamming and resource expenditure.

Tague et al. [57][58] and Tsang et al. [59] developed two complementary vulnerability definitions using two approaches: a set theoretic and a circuit theoretic approach for analysing the network traffic. They utilized a linear programming model to optimize the network traffic.

Sun, La Porta, and Kermani [60] proposed a key tree rebalancing algorithm for improving rekeying performance algorithms in order to provide an effective scheme for GK management. Furthermore, they presented a practical Location Based Service (LBS) implementation where they used hierarchical location information coding. Therefore, their system offers a facility for flexible location information access. Their load tests show that their system is highly practical with good efficiency and scalability.

Zheng et al. [61] investigated the trade-off between performance and confidentiality of existing signature-based air indexing schemes proposed for wireless data broadcast. They used two new metrics (false drop probability and false guess probability) in their evaluation and comparison.

Lin et al. [62] presented Small group PKI (Public Key Infrastructure) – less Authenticated Trust Establishment (SPATE) which is a primitive that allows users to establish trust via mobile devices and physical interaction. In this technique, when the SPATE protocol reaches its completion, its participant nodes will have authentic data that can be used by their applications to interact securely for one week. The SPATE protocol is suitable for large systems to provide effective, secure, and user-friendly collaboration using e-mail, file-sharing, and text messaging services [63], but it is unsuitable for small and medium systems which constitute our focus. This is one of the reasons why SPATE cannot directly be used without appropriately modifying it for small and medium systems.

Li, Koutsopoulos, and Poovendran [20] studied controllable jamming attacks in wireless sensor networks, which are easy to launch and difficult to detect and confront. The derived solutions to the optimization problems dictate optimal attack and network defence strategies. Of particular interest is their comparison between the case of perfect knowledge and that of a lack of knowledge of the attacker and the network about the strategy of each other. In the latter, the attacker and the network respond optimally to the worst-case strategy of the other. Xuan et al. [64] leveraged several optimization problems to provide a complete trigger-identification service framework for unreliable wireless sensor networks. They provided an improved algorithm with regard to two sophisticated jamming models, in order to enhance its robustness for various network scenarios. Theoretical analysis and simulation results are shown by them to validate the performance of their framework.

Fouad, Mostafa, and Dawood [65] proposed a scheme that uses prior know of the energy level of each node for developing a polynomial pool based key pre-distribution scheme as proposed earlier by Liu et al. [66]. Their work shows that "node energy level observations can be used to control the creation and the selection of polynomial keys held by this node". They evaluated their scheme by applying it to the A3 protocol (a topology control protocol). Their scheme avoids unnecessary key assignment and reduces the number of nodes that need to be active when constructing a new wireless sensor network (WSN) topology [65].

Tague et al. [57] presented Nymble, a system in which servers can "blacklist" misbehaving users, thereby blocking users without compromising their anonymity.

Calandriello et al. [29] analysed the effect of security on a virtual circuit (VC) system's effectiveness, specifically for a efficient safety application. They provide a framework to analyse the performance of secure VC systems, along with schemes that reduce the complexity and the overhead of security. They considered multiple system operational dimensions and identified interdependencies of various factors. They strongly believe that the systematic evaluation of overall performance is critical, especially for pervasive computing systems tightly coupled to their users. As security and privacy are paramount for these systems, but incur significant overhead, designs should be evaluated to show that the secured systems are effective, as envisioned and necessary.

Hur and Noh [67] proposed an access control mechanism that uses cipher text policies which are based on attribute level encryption in order to enforce effective access control policies with efficient attribute and user revocation capability. They provide a fine-grained access control mechanism using a dual encryption technique that takes advantage of existing techniques for attribute-based encryption and selective GK distribution in each attribute group. They demonstrated how to apply the proposed mechanism to securely manage outsourced data. Their results are useful for providing security in data outsourcing systems.

Sarkar and Saha [68] discussed the security characteristics unique to Hierarchical Wireless sensor Networks (HWNs) and showed that how attacks against single or multi-hop wireless networks can be translated into powerful attacks against HWNs. They investigated various types of attacks against HWNs and provided an overview of existing solutions for security protection. They also identified underlying challenges in securing HWNs infrastructure and protecting the transmitted information.

Jen-Ho and Chin-Chen [69] proposed a Dynamic Virtual Digraph (DVD) model for public key distribution. They developed a new key distribution scheme by extending graph theory. They proposed a key distribution scheme based on two-channel cryptography for a packet Delay Tolerant Network (DTN) that enables network services for mobile users *even* when they are not in a reachable area.

Liu and Zhu [70] proposed an ID-based remote mutual authentication with key agreement scheme using ECC. Moreover, they adopted multi-servers to realize the scheme's scalability. In multi-server based applications, many servers are linked in order to give the legitimate mobile users access network service (or to access another resource) more conveniently and efficiently.

Compared with existing works, the key distribution scheme proposed in this thesis is more efficient. The proposed scheme is also more secure and resistant to passive attacks *except for* collusion attacks. In addition, the scheme is computationally efficient and consumes only limited memory.

2.4 Privacy Preserving System

Shao et al. [71] presented a privacy enhanced key management technique for providing security to data centric sensor networks. In their work, they provide multiple levels of privacy based on different cryptographic keys. In addition, they proposed several query optimization techniques based on Euclidean Steiner Tree and Keyed Bloom Filter in order to minimize the query overhead while preserving query privacy.

Fan, Huang, and Ho [72] presented an anonymous multi-receiver identity-based encryption scheme where they adopted Lagrange's interpolating polynomial mechanisms to protect the privacy of the message receivers. They claim that their "scheme makes it more complex for an attacker to derive the identity of the message receiver, thus the privacy of receivers can be guaranteed."[72] As a result every receiver is anonymous to every other receiver [72]. Their claim that their scheme is "quite receiver efficient since each of the receivers merely needs to twice perform a pairing computation to decrypt the received cipher text" [72]. Their scheme proved to be secure against adaptive chosen plaintext or cipher text attacks[72].

Wang et al. [73] proposed a privacy preserving system to support cloud computing. They developed a new privacy preserving algorithm to protect individual's privacy. This privacy is an important issue for cloud computing both in terms of legal compliance and users' trust. They insist that privacy is an important security service that must be provided when designing cloud services for processing or sharing of personal data.

Mishra et al. [74] proposed a privacy preserving repository for securing data across the cloud. Their work enables the data owner to delegate most of the computationally intensive tasks to a cloud server *without* disclosing data contents or user access privilege information. Their proposed scheme supports user accountability, while supporting other cloud objectives, such as lower costs for hardware, maintenance, tuning, and support.

Greveler, Justus, and Loehr [75] presented a privacy preserving system for cloud computing that prevents the local administrator or the cloud administrator from learn about the outsourced database's content. This work uses an information centric approach, which aims to make cloud data self-intelligent. In this approach, the cloud data are encrypted and packaged together with a usage policy. When the data is accessed the process will consult this data's policy, create a virtualization environment, and attempt to access the trustworthiness of the data environment. In this manner, a database containing sensitive information is protected not only against an external administrator and service providers, but also against local administrators. However, the creation of complex machine readable access rights to the decryption keys becomes a challenging problem in their approach.

Liu et al. [76] proposed a cloud enabled privacy preserving collaborative learning mechanism for mobile sensing. A technique called a 'pickle' was used in their approach for privacy preserving in mobile collaborative learning. This technique uses a regression method to learn relationships between training data that are required to maintain classifier accuracy. It is secure against many kinds of attacks, including direct inversion, collusion, reconstruction, and poisoning. This method requires minimal computing resources, but requires more memory on the mobile devices.

Jung et al. [77] proposed a privacy preserving cloud data access with multiple-authorities. They presented an anonymous privilege control scheme 'Anonycontrol' to address both data privacy in the cloud storage and user identity privacy. Using multiple-authorities in the cloud computing system, they achieved both fine-grained privilege control and anonymity while conducting privilege control based on the user's identity information. Their proposed scheme can tolerate compromise of up to $(N-2)$ authorities, which is highly desirable, especially in an internet based cloud computing environment - as there may be network partitions or node failures. The cloud server cannot read the contents of the data unless their private keys satisfy the privilege tree.

Wang et al. [73] proposed a privacy preserving public auditing system to support secure storage of data in a cloud. They use a homomorphic linear authenticator and random masking to guarantee that the third party auditor (TPA) would not learn any knowledge about the data content stored on the cloud server during the efficient auditing process. This process minimizes the burden of cloud user from the tedious and possibly expensive auditing task, but also alleviates the users' fear of their outsourced data leakage. They also extended their privacy-preserving public auditing protocol into a multi-user setting, where the TPA can perform multiple auditing tasks in a batch manner for better efficiency. Their proposed scheme provides greater security and it is a highly efficient approach.

2.5 Literature Gaps

In spite of all the contributions to the literature summarized in the previous sections, many gaps remain with respect to an access control mechanism suitable for use in public clouds. Most of the schemes presented in the literature are computationally expensive. Moreover, the existing schemes have a number of overheads, including both memory and communication overheads. Therefore, it is necessary to propose a new and efficient key distribution and management scheme to provide the desired access control facility, to provide privacy in a public cloud, and to support secure multicast communication.

2.6 Proposed Work

Many researchers have proposed various key management schemes to provide secure group communication for wired and wireless networks. Compared with all the key management schemes that exist in the literature, the key management scheme proposed in this thesis is different in many ways. First, the proposed key management scheme increases security and minimizes the group key computation time of the data owner. In order to do so, the data owner uses a matrix in which each row is assigned to one cloud user. Using this matrix, an Access Control Vector (ACV) is computed to deliver the group key in a secure way from the data owner to each cloud user. Using this ACV, each user can find the group key to access documents from the CSP. When a user leaves the system that user's row will be eliminated from the matrix and the data owner computes a new access control vector. Similarly, when a user joins the service, a new row will be added in the matrix from which a new AVC is computed. This prevents the new user from viewing old documents. In this way, both forward and backward secrecy is provided by the key management scheme proposed in this thesis. In order to minimize the group key recovery (derivation) time by each cloud user, each cloud user only needs to perform one vector multiplication operation to find the new GK computed by the data owner.

Second, the proposed key management scheme is suitable for providing secure group communication in public clouds with low communication complexity. To minimize the communication complexity, the data owner sends only one multicast to distribute the ACV and other *public* parameters, which are subsequently used by each of the cloud users to individually compute the GK.

Third, a batch key updating algorithm (also called batch rekeying) is proposed in this thesis which reduces the number of rekeying operations required for batch leave or join operation. Finally, the proposed key distribution and key management scheme provides privacy preserving methods, whereby it preserves the privacy of the user thus hiding the user’s identity from the data owner and CSP. Therefore, the key management scheme proposed in this thesis reduces the computational complexity in all cases with the exception of a few cases where it is designed to increase security. In summary, this thesis provides an efficient key management algorithm, which is more efficient in terms of computation, communication, and memory requirements, that other existing work.

3 Group Key Management with Privacy Preserving Architecture

The system architecture shown in Figure 3-1 consists of four components:

1. Data Owner,
2. Cloud Service Provider (CSP),
3. Token Generator, and
4. User.

The data owner is the one who places the original documents in the public cloud to be accessed by the cloud users. A CSP operates a single server or a collection of servers used to maintain the data owner's data. The token generator is used to generate a token, which should be given to each cloud user to get a secret key from the data owner. A user is a person (or application acting on behalf of this user) who wants to access the data via the CSP. Initially, each cloud user must send their identity attributes to the token generator to get a token. The token generator receives the identity attributes from the cloud user and generates a token. After generating this token for a cloud user based on their identity attributes, it gives the newly generated identity token to the user and then sends the identity token of each cloud users to the data owner for verification. The verification is performed *after* giving the token to the user, since the token generator has to send the identity token of a cloud user to the data owner *only* if it is correctly delivered to that particular cloud user. After receiving confirmation from only that cloud user, the token generator sends the identity token to the data owner.

The user registers their identity token with the data owner to get a secret key. The data owner gives secret keys to the user based on their identity token. After providing the user with a secret key, the data owner computes a GK for each group of users based on their secret key values. Then, the data owner encrypts the documents and then uploads the encrypted document to the cloud provided by the CSP. Each cloud user can derive the GK using their secret key and thus can use this GK to decrypt the encrypted documents placed in the cloud. When group membership changes, it is the responsibility of the data owner to change the GK. The data owner may also change the GK periodically. For example, when a user leaves a group or joins the group the data owner downloads the corresponding document from the cloud service provider and re-encrypts the document with the new GK. Then, each re-encrypted document is uploaded into the cloud. These processes are grouped into three modules:

1. Privacy preserving module
2. Key management module
3. Document management module

Each of these modules will be describe in one of the following sections of this chapter. The chapter will end with a section that describes how these three modules interact.

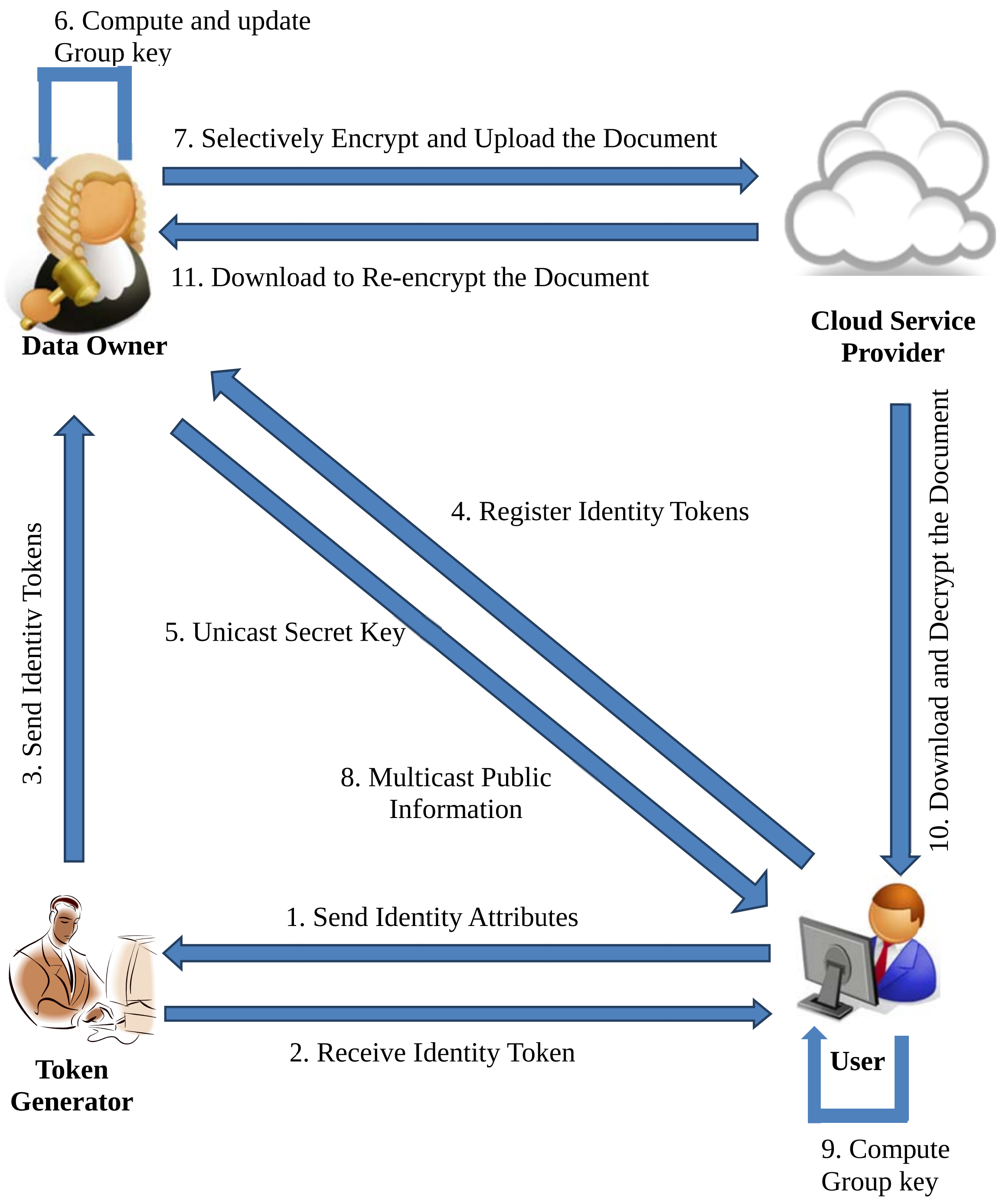

6. Compute and update Group key
7. Selectively Encrypt and Upload the Document
11. Download to Re-encrypt the Document
Data Owner
Cloud Service Provider
4. Register Identity Tokens
5. Unicast Secret Key
3. Send Identity Tokens
10. Download and Decrypt the Document
8. Multicast Public Information
1. Send Identity Attributes
2. Receive Identity Token
Token Generator
User
9. Compute Group key

Figure 3-1: **Privacy Preserving Group Key Management Architecture**

3.1 Privacy Preserving Module

The privacy preserving module is designed to protect the privacy of each cloud user. Each user has his/her own identity attributes The identity attributes of the user should **not** be known to the data owner or other users. . In order to preserve the user's privacy, the identity attributes of the user need to be protected. Moreover, if the identity attributes were to be known to any other persons, then these other persons could access the data as if they were this user – thus lead to a security breach. For this reason it is important to protect the user's identity attributes. Data is protected from access by unauthorized users by protecting the user's identity attributes. The privacy preserving module consists of two phases: Identity token issuance and Identity token registration.

3.1.1 Identity Token Issuance

In the identity token issuance phase, each and every user provides their identity attributes to the token generator. The token generator issues, an identity token for the user after receiving their identity attributes. This identity token encodes a user's identity in a specified electronic format in which identity attributes values are represented in a secure cryptographic commitment. The form used in this thesis for this commitment is known as the Pedersen commitment [78].

Figure 3-2 shows an example of this phase of the token distribution process - as seen from the token generator's side. In this figure, a user who called "Bob" submits his name along with his date of birth (24.12.1990) to get a token from the token generator. The token generator generates a pseudonym (pn), age {derived from the date of birth}, commitment value (6267292101), and the digital signature of all three values (949148425702313975). The cloud user is given resulting token. The cloud user can use this token to conceal their identity from the data owner and CSP.

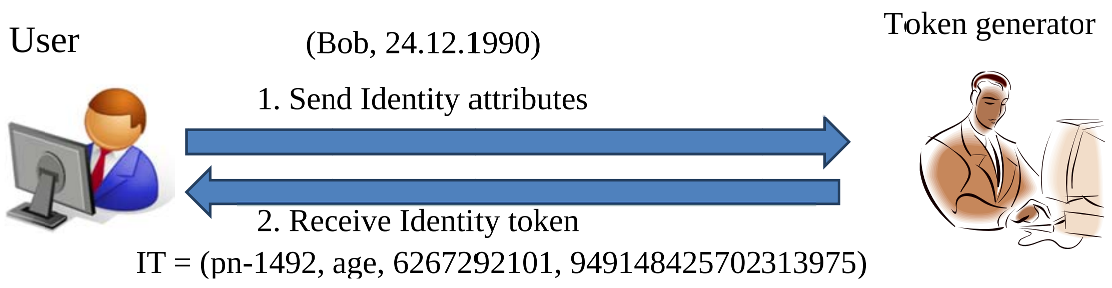

User
(Bob, 24.12.1990)
Token generator
1. Send Identity attributes
2. Receive Identity token
IT = (pn-1492, age, 6267292101, 949148425702313975)

Figure 3-2: Example of identity token issuance

3.1.2 Identity Token Registration

The users register their identity tokens with the data owner in order to access the documents stored by the CSP. During the registration phase, the users present their identity tokens to the data owner and receive a secret key value. Note that the data owner does not have a mapping between the user's identity and the token. Only the token generator has a mapping between the user's identity and the token since it is considered a trusted third party. The data owner cannot map the user's identity and the token, since user identity is not available to the data owner. The data owner has only the pseudonym of each cloud user using which the data owner cannot find the users original identity. This secret value is used by the user to derive the GK to decrypt documents provided by the CSP. The secret value is given to the user using an existing privacy preserving approach known as the OCBE protocol developed by Li and Li [78]. The OCBE protocol allows the user to obtain the secret key value if and only if the users committed identity attribute values satisfies the condition of the access control policy of the data owner. A cloud user can register for receiving an identity token from the token generator for various attributes such as roles, age, year of service etc. In some cases, cloud user may use more than one attribute and in some cases it may register with only one attribute. An identity token is a user's identity in a specified electronic format in which the involved identity attribute value is represented by a semantically secure cryptographic Commitment.

The data owner groups the access control policies (ACPs) into policy configurations. An access control policy (ACP) is a policy that consists of three tuples namely the document (D), subdocument (S) and set of conditions (A). An example ACP is ("Age≥58" ∧ "role = nurse" {physical exam, treatment plan}, "EHR.xml"), where EHR.xml is the main document, physical exam, and treatment plan are the sub documents. An ACP can have various conditions such as $\{<, \leq, \geq, >, =, \land\}$. The above example uses the operators $\geq$, $\land$, and $=$.

In addition to this, the data owner divides the documents into subdocuments based on their policy configurations. After that, the data owner generates a group key based on ACPs in each policy configuration and selectively encrypts the subdocuments and the encrypted subdocuments are uploaded into the cloud. During the user join or leave case, the data owner checks the corresponding user's pseudonym to verify the token and to issue the secret key. Once the secret is issued, the document owner adds a row to the key matrix for this new user and performs a rekey process only for the relevant subdocuments. Similarly, when a user with a pseudonym needs to be removed, the data owner removes the row corresponding to that particular user's pseudonym from the key matrix, and performs the rekeying process. Thus, the data owner does not learn anything about the user's identity attributes value. Moreover, the data owner cannot communicate the user's identity attributes to others since it does not have the user identity attributes in its storage area. This does not affect the data owner's control over the users access rights since they are set based on the commitment generated by the token generator which is explained in section 4.1.Therefore, the privacy of the user is preserved by protecting the user's credentials, by adding/revoking credentials and ACV updates. The credentials are roles, age, year of service, etc., which can be updated for various reasons such as promotions, change of responsibilities, and so on.

This user's credentials have to be updated dynamically from time to time for various reasons such as promotions, change of responsibilities, and so on. If a user updates his/her credentials with the data owner, then the data owner needs to remove the row corresponding to that particular user from the matrix T, and performs a rekey process only for the subdocuments involved.

3.2 Key Management Module

In group key management, a symmetric key encryption scheme is used to encrypt and decrypt the data. Group key management schemes can be divided into two types: static and dynamic. In a dynamic group key management scheme, the GK is periodically updated when the membership of the group changes. This key updating process (also called as rekeying) requires additional computation. A multicast dynamic group key management scheme is used in this thesis. In this scheme forward and backward secrecy is maintained by changing only the public information stored in the cloud *without* affecting the secret values given to the user. In group key management schemes, the GK is not given directly to the group users, but rather a secret key is given to each user. Each user combines this secret key with public information to compute the GK. The advantage of this group key management scheme is that secrecy is maintained by changing in an efficient way *only* the public information. The secret key is not affected by these changes and hence attackers do not learn the secret key. The dynamic group key management scheme proposed in this thesis consists of five phases: setup phase, secret and public key generation phase, key computation phase, key derivation phase, and key updating phase.

Setup Phase	In this phase, the data owner generates an l-bit prime number 'q' to define a finite field F_q and a cryptographic hash function H( ). In addition to this, the data owner computes key space KS=F_q and secret space SS={1,2,3,.......,2^l-1}.
Secret and public key generation phase	In this phase, the data owner randomly generates a secret key s_i for each user$_i$. This secret key is known only to the data owner and the corresponding user for whom the key is issued. Moreover, the data owner also generates a public key value z_i from the secret space SS and informs all the existing users of this public key.
Key computation phase	In this phase, the data owner randomly chooses a GK from the key space (KS) computes the public information using each user's secret value and the public key, and multicasts this public information to all the cloud users.
Key derivation Phase	In this phase, each of the cloud user derives the GK by using their secret key value and the public information received from the data owner. Each user can subsequently use this GK to decrypt a (sub)document stored in the cloud.

Key updating phase	When a new user joins or leaves the group, a new GK has to be generated and distributed securely to the cloud users. The data owner performs the key computation phase and outputs the new public information for the newly generated GK. The active group members can compute the group key by using the key derivation phase.

3.3 Document Management Module

Document management is done *only* by the data owner. The data owner encrypts all documents that are to accessible to the group by using the GK. The owner uploads the encrypted subdocuments into the cloud operated by the CSP. If a user wants to view a document, he or she downloads the encrypted document from the cloud. After the user gets the secret key from the data owner in a secure way, the user derives the GK needed to decrypt the document using the public information and public key values received from the data owner (as this public information includes this user's own copy of the GK encrypted using their private key). The user then decrypts the document using the derived GK. When a new user joins or leaves the group, the data owner uploads each of the documents to be accessible to this new group by encrypting them using the new GK.

The data owner can use policies to determine whether a given user should have access to a document by considering whether to include this user in a given group or not. The data owner might use an access policy such as:

("Age ≥58" ∧ "role = nurse" {physical exam, treatment plan}, "EHR.xml")

The above access policy says that a user of age greater than 58 and having a nurse role can access the subdocuments "physical exam" and "treatment plan" of the document "EHR.xml".

Documents are divided into subdocuments based on the applicable ACP. Different access control policies can apply to the same subdocuments, because such subdocuments may have to be accessible to different categories of cloud users. More than one group of users may have access to a single subdocument and group overlapping con occur. We have proposed that this be handled as a part of the future work mentioned in section 7.2.

3.4 Sequence diagram

Figure 3-3 shows a sequence diagram, showing the sequence and flow of each process described in the previous sections. As shown in the sequence diagram, for getting a token (before the user registers with the data owner), the cloud user initially registers their identity attributes with the token generator for generating the token. Based on these identity attributes, the token generator creates an identity token. This identity token is given to both the user and data owner. The user registers their identity token (provided by the token generator) with the data owner in order to get a secret key that can later be used to compute a GK. This GK can be used to decrypt documents downloaded from the cloud. After receiving the user's identity token, the data owner computes a secret key based on the access control policy relevant to this user – thus classifying this user's rights into those rights assigned to a given group. This secret key is distributed to the cloud users in a secure way. After receiving the secret key, each user can derive the GK based on public information and their secret key. Now the user can download documents from the cloud and decrypt them using the GK.

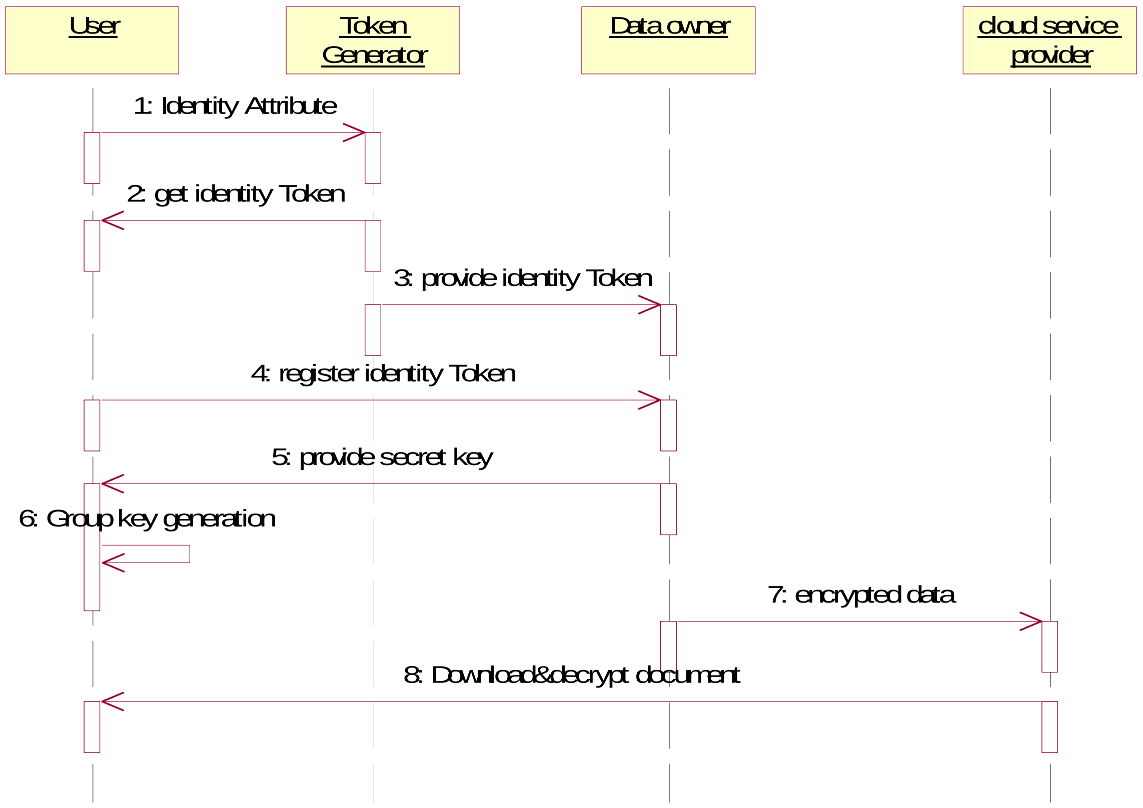


Figure 3-3: Sequence diagram

4 Implementation of Privacy Preservation and Key Management Modules

This chapter explains the privacy preserving system used in this thesis to preserve the privacy of each cloud user. In addition to this, this chapter also explains the group key management scheme used to provide secure group communication in a public cloud.

4.1 Privacy Preserving Module

Privacy preserving is one of the important algorithms used in this thesis work to preserve privacy of the cloud users in the public cloud environment. The privacy preserving algorithm consists of two protocols Pedersen commitment and OCBE protocol.

4.1.1 Pedersen commitment

The Pedersen commitment scheme is used to preserve the privacy of the user. The scheme hides the user's identity attributes value and never reveals these attributes value to any other parties. The Pedersen commitment scheme consists of three phases: Setup, Commit, and Open.

Setup In the setup phase, the token generator defines a finite cyclic group G of the prime number p. The token generator then chooses a generator g and an integer α and computes another generator h such that: $\mathrm{h} = \mathrm{g}^{\alpha} \mathrm{mod}\, p$. Alpha is an integer chosen from the multiplicative prime group such that $\mathrm{g}^{\alpha} \mathrm{mod}\ \mathrm{p} = \mathrm{h}$.

In this case the token generator may or may not store the integer value α.

Commit The token generator randomly chooses a value $\mathrm{r} \in \mathrm{F_p} = \{1, 2\ldots \mathrm{p}\text{-}1\}$, then the token generator commits a value $x \in \mathrm{F_p}$ by computing a commitment value (c) as: $\mathrm{c} = \mathrm{g}^{x} \cdot \mathrm{h}^{\mathrm{r}}$

Open In the open phase, the token generator sends$x$, r, and c to the cloud user and sends the c value to the data owner. The data owner then verifies the c value for the valid cloud user as explained in section 4.1.2.

4.1.2 Oblivious Commitment Based Envelope (OCBE) protocol

In order to preserve the privacy of the user, the OCBE protocol is used as the protocol to securely deliver the secret key value to the valid cloud users. The OCBE protocol consists of three components:

1. The token generator
2. User
3. The data owner

The token generator computes the commitment value using the Pedersen commitment proposed by Nabeel, Shang, and Bertino [79] The token generator then sends the commitment value to the data owner and sends the values $x, r,$ and c to the cloud user.

The interaction between the data owner and the cloud users to deliver the secret value for each user in a secure way is:

- The cloud user sends a data request to the data owner.
- After receiving the request, the data owner sends the predicate $\mathrm{GE}_{\mathrm{x_0}}$ to the cloud user.
- The cloud user receives this predicate and sends the Pedersen commitment value c to the data owner.
- The cloud user computes $d = (x - x_0)\ (mod p)$, where d is the commitment
- The cloud user picks $\mathrm{r_1},\ldots,\mathrm{r_{l-1}} \in \mathrm{F_p}$ and computes $\mathrm{r_0} = -\sum_{\mathrm{i}=1}^{\mathrm{l}-1} 2^{\mathrm{i}}\, \mathrm{r_i}$.
- Let$\mathrm{d_{l-1}},\ldots, \mathrm{d_1}, \mathrm{d_0}$ be d's representation calculated by the cloud user.
- The cloud user then computes ℓ commitments by using$\mathrm{c_i} = \mathrm{g}^{\mathrm{d_i}} \cdot \mathrm{h}^{\mathrm{r_i}}$ for $0 \leq i \leq \ell - 1$.
- The user then sends all the commitments to the data owner.
- The data owner verifies whether $c \cdot \mathrm{g}^{-\mathrm{x_0}} = \prod_{\mathrm{i}=0}^{\mathrm{l}-1} (\mathrm{c_i})^{2^{\mathrm{i}}}$.

- The data owner randomly chooses $k_0, \ldots, k_{l-1}$ and sets $k = H(k_0 \parallel \cdots \parallel k_{\ell-1})$.
- The data owner picks $y \in F_p$ and calculates $\eta = h^y$.
- After that, the data owner encrypts the secret value $C = \varepsilon_k[M]$ where M is the secret value which is to be given to the cloud user.
- The data owner computes $\sigma_i^j = (c_i g^{-i})^y$, $C_i^j = H(\sigma_i^j) \oplus k_i$ for each $j = 0,1$ and $0 \le i \le \ell - 1$.
- The data owner sends the tuple $\{\eta, C_0^0, C_0^1, \ldots, C_{l-1}^0, C_{l-1}^1, C\}$ to the cloud user.

The cloud user receives the tuple $\{\eta, C_0^0, C_0^1, \ldots, C_{l-1}^0, C_{l-1}^1, C\}$ from the data owner. The cloud user then computes $\sigma_i' = \eta^{r_i}$ and calculates $k_i' = H(\sigma_i') \oplus C_i^{d_i}$ for $0 \le i \le \ell - 1$. The data owner derives the key $k' = H(k_0' \parallel k_1' \parallel \ldots \parallel k_{l-1}')$ Using the key k', the cloud user decrypts the secret value M from C.

4.1.3 Privacy Preserving Module Example

The first subsection explains a working example of Pedersen commitment and the second subsection explains the working example of OCBE.

4.1.3.1 Pedersen commitment

Let p=11, g=2, and α=8 then

$$h = g^{\alpha} \bmod p$$
$$= 2^8 \bmod 23$$
$$= 3$$

The commitment value

$$c = g^x . h^r \quad \text{(Where h=3, x=4, g=2, r=4)}$$
$$= 2^4 . 3^4$$
$$= 16 \times 81$$
$$c = 1296$$

4.1.3.2 OCBE protocol

The token generator computes the commitment value using the Pedersen commitment scheme and sends the value to the data owner and the cloud user.

The user computes

$$d = (x - x_0) \bmod p$$
$$= (4-3) \bmod 11$$
$$= 1$$

Let ℓ=2 then

$$\ell - 1 = 2-1 = 1$$
$$r_0 = r - \sum_{i=1}^{\ell-1} 2^i r_i$$
$$= r - 2^1 r_1 \text{ (where, } r_1 = 1)$$
$$= 4-2=2$$

d_0=1

d_1=0 be d's representation.

User computes ℓ - commitments

$$c_i = \sum_{i=0}^{\ell-1} g^{d_i} h^{r_i}$$

$$c_0 = g^{d_0} h^{r_0} = 2^1 . 3^2 = 2 \times 9 = 18$$

$$c_1 = g^{d_1} h^{r_1} = 1^0 3^1 = 3$$

The cloud user then sends the commitments to the data owner.

Data owner verifies:

$$c . g^{-x_0} = \prod_{i=0}^{\ell-1} c_i^{2^i}$$

$$1296 \times 2^{-3} = c_0^{2^0} \times c_1^{2^1}$$

$$1296 \times 0.125 = 18^1 \times 3^2$$

$$162 = 18 \times 9$$

$$162 = 162$$

Data owner picks randomly y=2

Data owner Computes = h^y

$$\eta = 3^2 = 9$$

Data owner chooses $k_0 ... k_{\ell-1}$

$$k_0 = 1$$

$$k_1 = 2$$

Therefore, k= $k_0 \parallel k_1$

$$k=12$$

C =$\mathcal{E}_k$[M] Where **M** is the Secret key

$$= \mathcal{E}_{12}[5]$$

$$= \text{YK7EPQC6w3bq4imushc23Q}$$

Data owner computes:

$$\sigma_i^j = \left(c_i g^{-j}\right)^y$$

$$\sigma_0^0 = (c_0 g^{-0})^2 = (18. 2^{-0})^2 \quad = (18)^2 = 324$$

$$\sigma_0^1 = (c_0 g^{-1})^2 = (18. 2^{-1})^2 \quad = (9)^2 = 81$$

$$\sigma_1^0 = (c_1 g^{-0})^2 = (3. 2^{-0})^2 \quad = (3)^2 = 9$$

$$\sigma_1^1 = (c_1 g^{-1})^2 = (3. 2^{-1})^2 \quad = (1.5)^2 = 2.25$$

Data owner then calculates:

$$c_0^0 = \mathrm{H}(\sigma_0^0) \oplus k_0 = \mathrm{H}(324) \oplus 1 = 104976 \oplus 1 = 104977$$

$$c_0^1 = \mathrm{H}(\sigma_0^1) \oplus k_0 = \mathrm{H}(81) \oplus 1 = 6561 \oplus 1 = 6560$$

$$c_1^0 = \mathrm{H}(\sigma_1^0) \oplus k_1 = \mathrm{H}(9) \oplus 1 = 81 \oplus 2 = 83$$

$$c_1^1 = \mathrm{H}(\sigma_1^1) \oplus k_1 = \mathrm{H}(2.25) \oplus 1 = 2 \oplus 2 = 3$$

The data owner sends the tuple:

(9,104977,6560,83, 3,YK7EPQC6w3bq4imushc23Q) to the user.

The user receives the tuple and the user calculates:

$$\sigma_i' = \eta^{r_i}$$

$$\sigma_0' = \eta^{r_0} = 9^2 = 81$$

$$\sigma_1' = \eta^{r_1} = 9^1 = 9$$

User computes the key using:

$$k_i = \mathrm{H}(\sigma_i^j) \oplus c_i^{d_i}$$

$$k_0 = \mathrm{H}(\sigma_0^1) \oplus c_0^{d_0} = \mathrm{H}(81) \oplus c_0^1 = 1$$

$$k_1 = \mathrm{H}(\sigma_1^1) \oplus c_1^{d_1} = \mathrm{H}(9) \oplus c_1^0 = 2$$

$$k_0 \parallel k_1 = 12 = k$$

User decrypts C using k to obtain the secret key.

4.2 Group Key Management Module

The group key management Broadcast Group Key Management (BGKM) scheme developed in this thesis consists of five phases:

1. Setup phase,
2. Secret key and public key generation phase,
3. Key computation phase,
4. Key derivation phase, and
5. Key updating phase.

4.2.1 Setup phase

In this phase, the data owner generates an l-bit prime number 'q' to define a finite field F_q and a cryptographic hash function H( ). In addition to this, the data owner computes key space KS=F_q and secret space SS={1,2,3,.......,2^l-1}.

4.2.2 Secret key and public key generation phase

In this phase, the data owner randomly generates a secret key s_i for each user$_i$. This secret key is known only to the data owner and the corresponding user for whom the key is issued. Moreover, the data owner also generates a public key value z_i from the secret space SS and informs all the existing users of this value.

4.2.3 Key computation phase

In this phase, the data owner selects a random element K, from the key space KS=F_q as GK. The data owner also creates an $n \times (n+1)$ as shown below:

$$\tau = \begin{pmatrix} 1 & a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ 1 & a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{pmatrix} \tag{1}$$

Where $a_{i,j}$= H (s_i+z_i), $1 \le i \le n, 1 \le j \le n$

s_i= Secret key value

z_i = Public key value

H( )=Hash function.

Data owner then solves for a column vector Y from τ such that

$$\tau \cdot Y = 0$$

After that, the data owner finds the Access Control Vector (ACV), X such that

$$X = (K \cdot e_1^T) + Y \tag{2}$$

Where K is the chosen GK ande_1^T is the transpose of the standard basis vector e_1 ={1,0,0,0,……….,0}.

After computing the access control vector the data owner outputs the public information $PI = \{X,(z_1, z_2 \ldots z_n)\}$.

4.2.4 Key derivation Phase

In this phase, each user derives the GK. In order to derive the GK, each user has to use their secret key value $s_i$and public information PI. User $User_i$ has a secret value s_i and the public information PI. Using s_i and PI each user computes $a_{i,j}$. $User_i$. Thus the user derives the GK as $K'= v_i \times X$, Where v_i is the row vector of $User_i$ and v_i= (1,$a_{i,1}$, $a_{i,2}$……..$a_{i,n}$). Using this $a_{i,j}$ value, each user derives GK.

4.2.5 Key Updating Phase

When a user joins or leaves the group the data owner runs the key computation phase and outputs the new public information PI' and the new GK' to provide backward and forward secrecy.

4.2.6 Key Management Module Example

In this subsection, we give an example of the key distribution and key management module.

4.2.6.1 Setup phase

Let $q = 11$

H (X) = s_i + Z_i

KS = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

SS={1,2,3,4,5,6,7,8,9,10,11,12,13,14,15}

4.2.6.2 Secret key and public key generation phase

Consider the three users A, B, and C (no. of users n=3) who belong to the same group. Hence, consider the secret keys s_1 = 2, s_2 = 4, s_3 = 9 and the public keys z_1 = 5, z_2 = 7, z_3 = 2.

4.2.6.3 Key computation phase

Initially, the data owner computes a 3×(3+1) matrix $\tau = \begin{pmatrix} 1 & 7 & 9 & 4 \\ 1 & 9 & 0 & 6 \\ 1 & 3 & 5 & 0 \end{pmatrix}$.

Where, $a_{1,1}$=2+5=7, $a_{1,2}$=2+7=9;$a_{1,3}$=2+2=4;$a_{2,1}$= 4+5=9; $a_{2,2}$=4+7=11mod11=0; $a_{2,3}$ = 4+2=6, $a_{3,1}$= 9+5=14 mod 11=3;$a_{3,2}$=9+7=16 mod11=5; $a_{3,3}$= 9+2=11 mod 11=0. Then, it solves the matrix τ to find the vector Ysuch that $\tau . Y = 0$.

$$\tau = \begin{pmatrix} 1 & 7 & 9 & 4 \\ 1 & 9 & 0 & 6 \\ 1 & 3 & 5 & 0 \end{pmatrix} \Leftrightarrow \begin{pmatrix} 1 & 7 & 9 & 4 \\ 0 & 2 & -9 & 2 \\ 1 & 3 & 5 & 0 \end{pmatrix} R_2 \longrightarrow R_2 - R_1$$

$$\Leftrightarrow \begin{pmatrix} 1 & 7 & 9 & 4 \\ 0 & 2 & -9 & 2 \\ 0 & -4 & -4 & -4 \end{pmatrix} R_3 \longrightarrow R_3 - R_1$$

$$\Leftrightarrow \begin{pmatrix} 1 & 7 & 9 & 4 \\ 0 & 1 & \frac{-9}{2} & 1 \\ 0 & -4 & -4 & -4 \end{pmatrix} R_2 \longrightarrow \frac{R_2}{2}$$

$$\Leftrightarrow \begin{pmatrix} 1 & 7 & 9 & 4 \\ 0 & 1 & \frac{-9}{2} & 1 \\ 0 & 0 & -22 & 0 \end{pmatrix} R_3 \longrightarrow R_2 \times 4 + R_3$$

$$\Leftrightarrow \begin{pmatrix} 1 & 7 & 9 & 4 \\ 0 & 1 & \frac{-9}{2} & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} R_3 \longrightarrow R_3 \times \frac{-1}{22}$$

$$\Leftrightarrow \begin{pmatrix} 1 & 7 & 9 & 4 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} R_2 \longrightarrow R_3 \times \frac{9}{2} + R_2 \Leftrightarrow \begin{pmatrix} 1 & 7 & 0 & 4 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} R_1 \longrightarrow R_3 \times\text{-}9 + R_1$$

$$\Leftrightarrow \begin{pmatrix} 1 & 0 & 0 & -3 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} R_1 \longrightarrow R_2 \times\text{-}7 + R_1$$

From this, Y_i values are computed where, Y_3= 0; Y_2 = -1 × Y_4; Y_1 = 3 × Y_4; Y_4 = 1 (arbitrary):

$$\text{Y} = \begin{pmatrix} 3 \\ -1 \\ 0 \\ 1 \end{pmatrix}$$

Let e_1 = (1, 0, 0, 0)

$$e_1^T = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

If the group key$K = 3$, then $e_1^T.K = \begin{pmatrix} 3 \\ 0 \\ 0 \\ 0 \end{pmatrix}$

$$\text{X} = (K.e_1^T) + Y$$

$$= \begin{pmatrix} 3 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 3 \\ -1 \\ 0 \\ 1 \end{pmatrix} == \begin{pmatrix} 6 \\ -1 \\ 0 \\ 1 \end{pmatrix};$$

The public information = $\left(\begin{pmatrix} 6 \\ -1 \\ 0 \\ 1 \end{pmatrix}, (5,7,2) \right)$

4.2.6.4 Key derivation Phase

User gets the public information to derive the group key, $K = v_i \cdot \text{X}$

User A:

$$K = v_1 \cdot X = \begin{pmatrix} 1 & 7 & 9 & 4 \end{pmatrix} \begin{pmatrix} 6 \\ -1 \\ 0 \\ 1 \end{pmatrix} = 6-7+0+4 = 3$$

User B:

$$\text{K} = v_2 \cdot X = \begin{pmatrix} 1 & 9 & 0 & 6 \end{pmatrix} \begin{pmatrix} 6 \\ -1 \\ 0 \\ 1 \end{pmatrix} = 6-9+0+6 = 3$$

User C:

$$K = v_3 \cdot X = \begin{pmatrix} 1 & 3 & 5 & 0 \end{pmatrix} \begin{pmatrix} 6 \\ -1 \\ 0 \\ 1 \end{pmatrix} = 6 – 3 + 0 + 0 = 3$$

5 Testing of Privacy Preserving Module

This section describes the experimental test procedure use to test the prototype implementation of the proposed privacy preserving module and key management algorithm.

5.1 Simple experimental cloud test bed

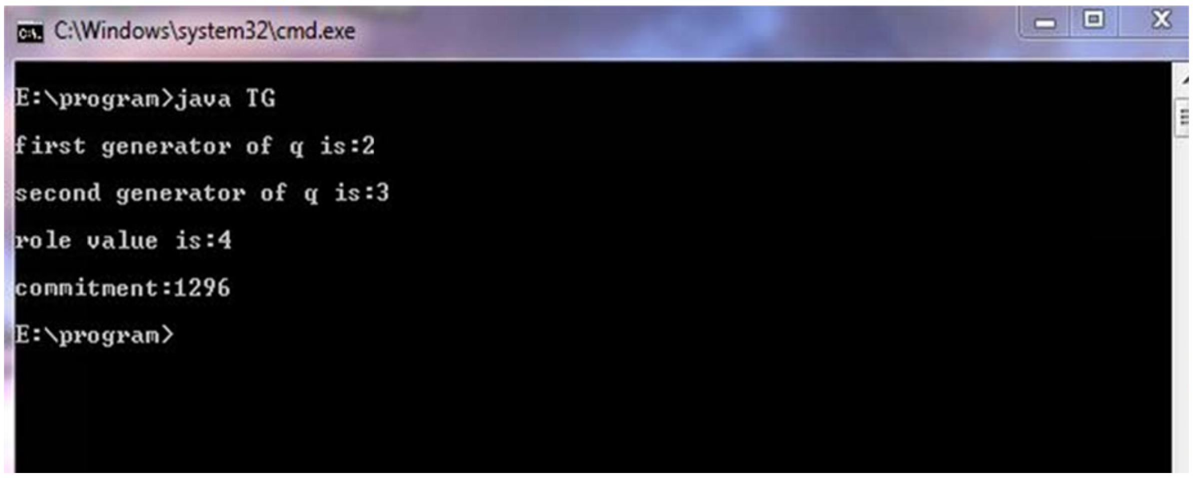
The simple test bed consisted of two computers with similar hardware (see Table 5-1).One computer acts as the token generator, data owner, and CSP. The other computer acts as a cloud user who wishes to access data from the cloud (as realized by the first computer). The cloud user can also be a smart phone whose computation and memory complexity may be lower than a high speed computer. However, the computation complexity and storage complexity remain same regardless of the device. The storage complexity of the cloud user is O(2) since each cloud user stores only one secret key and group key to access the documents from the cloud. Similarly, the storage complexity of the data owner for a particular group is $O(2N) + 1$, where $2N$ represents storage space of all users secret and public keys and 1 represents the memory required to store a group key. If there are N different groups, then the data owner has to store N key values (as required for all the N groups). In this thesis, we concentrated on a single group in which single rekeying and batch rekeying operations are considered. In a real application, there may be many groups where we need to consider group and document overlap, however consideration of these issues are out of the scope of this thesis project.

Initially the data owner uses a fixed bucket size (sample size of 10 users) key matrix to compute the GK. This doesn't limit the number of users to 10, instead all the users are grouped to fit a bucket size of 10 and new buckets can be added, whenever a new user joins. The group key computation time is analysed and compared with existing approaches to perform the key updating operation. The key computation and updating algorithm were implemented in Java™. For the data owner to compute the inverse value of the generated matrix τ, we have used a Gauss-Jordan matrix elimination in our source code The privacy preserving module uses a method Math.pow() to perform the modulo exponential operation with respect to the field size. The document management module uses the Advanced Encryption Standard (AES) algorithm to encrypt the documents and store them in the first computer. The class Multicast Socket is used by the data owner to multicast the AVC value to the cloud users. Each user uses thejoinGroup() method to join an existing group.

Table 5-1: Hardware and software configuration of test computers

Make	Processor	Memory	Disk	Operating system
Intel Core i5-2450M CPU @ 2.50 GHz	Intel i-5	2GB RAM	320GB hard disk	Microsoft's Windows 7 Professional 64-bit operating system

The token generator initially generates the generator values from which commitment values are computed. Figure 5-1 shows a commitment value calculated by the token generator. The token generator sends this commitment value to the data owner and to the user. The cloud user submits their commitment value to the data owner. The data owner verifies this commitment value. If the commitment values match(pn), then the data owner generates a secret key and sends it securely to the cloud user.

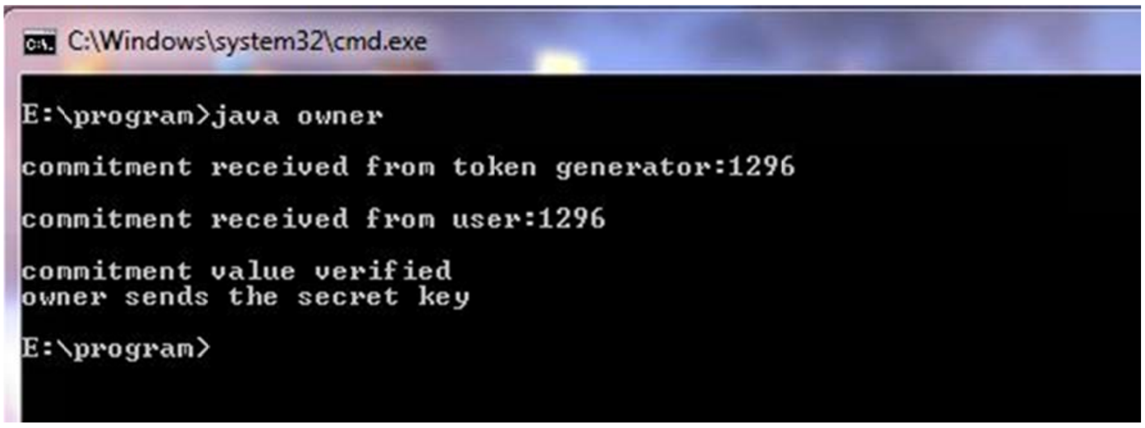


```
E:\program>java TG
first generator of q is:2
second generator of q is:3
role value is:4
commitment:1296
E:\program>
```

Figure 5-1: Pedersen Commitment Value

5.1.1 Verification of commitment value

In this phase, the data owner verifies the commitment value submitted by the cloud user. Figure 5-2 shows that the data owner has verified the commitments sent by the user. After verifying the commitments the data owner sends each user a secret key value.

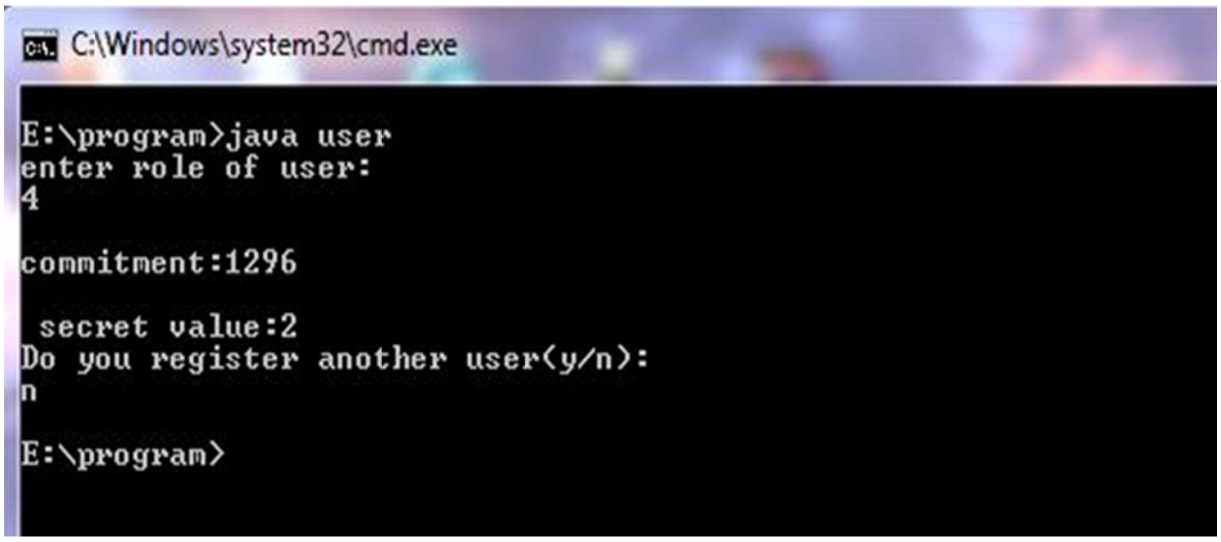


```
E:\program>java owner
commitment received from token generator:1296
commitment received from user:1296
commitment value verified
owner sends the secret key
E:\program>
```

Figure 5-2: Verification of commitments by the data owner

5.1.2 Derivation of secret key value

In this phase, the cloud users receive their secret key value from the data owner in a secure way. Figure 5-3 shows that the user derives the secret key value, which was sent by the data owner from the tuple received from the data owner.



```
C:\Windows\system32\cmd.exe

E:\program>java user
enter role of user:
4

commitment:1296

 secret value:2
Do you register another user(y/n):
n

E:\program>
```

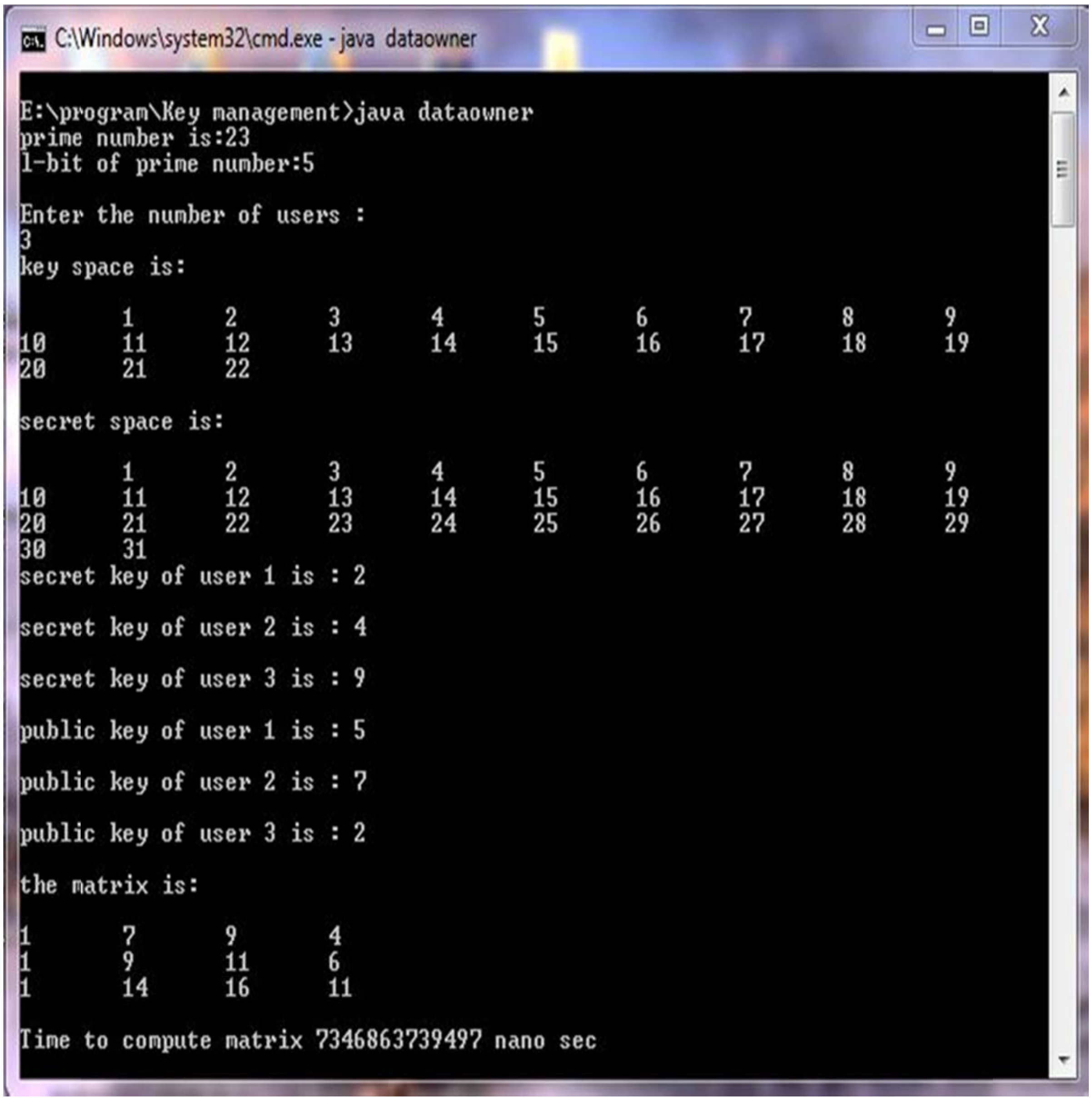
Figure 5-3: Derivation of secret key value by the cloud user

5.2 Key Management Module

This section explains the process of generating, distributing, and recovering the group key values by the data owner and cloud users.

5.2.1 Group key generation

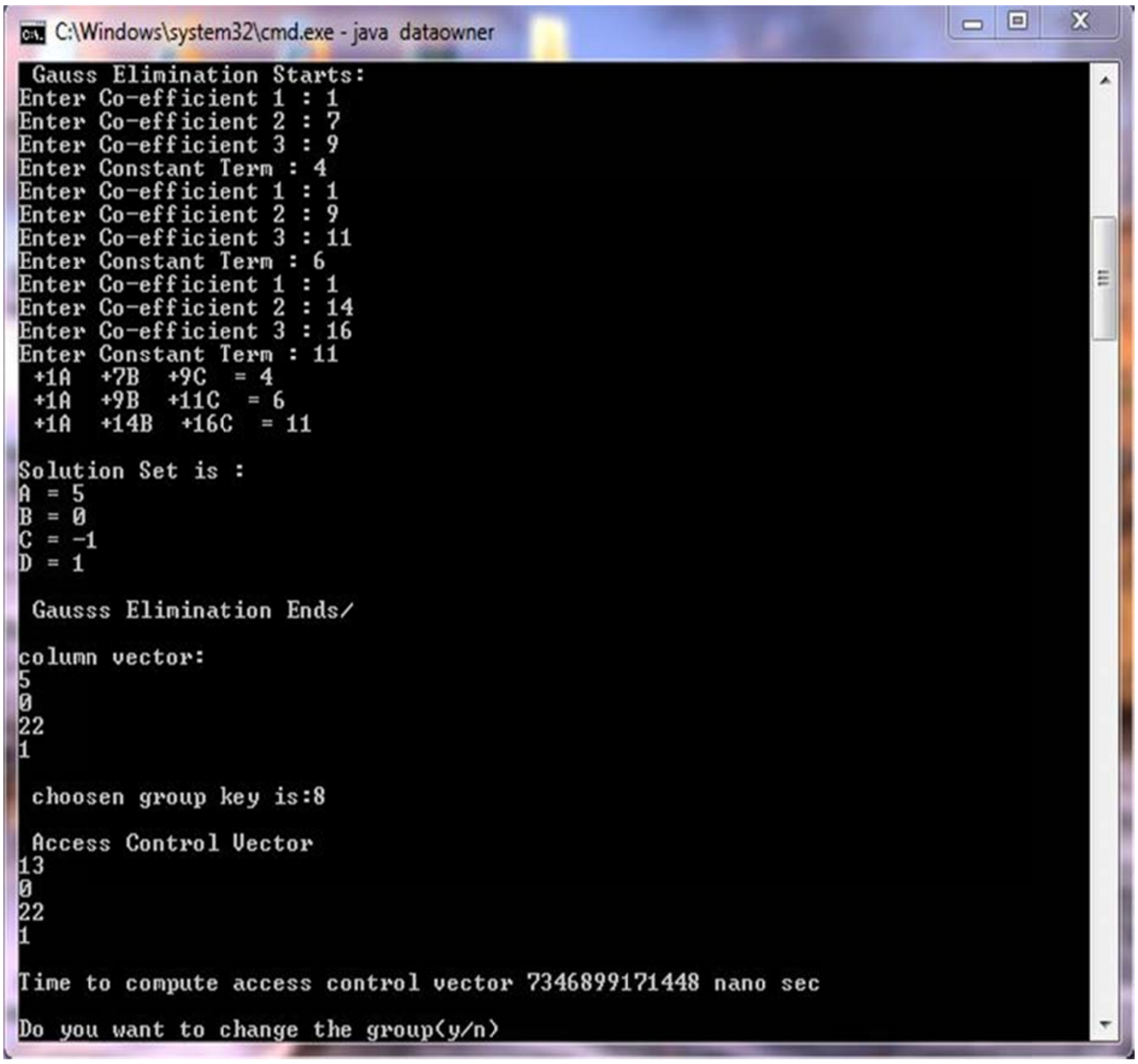
In this phase, the data owner generates the key space and secret space as shown in Figure 5-4. In addition, the data owner also generates a key matrix, which is used to compute the ACV value that can be used to access the documents stored in the cloud. Our sample implementation uses a smaller size key values, however, in real time applications, a minimum of 256 bits or 512 bits key values should be used to avoid brute force attack. Figure shows that the data owner computes a n×(n+1) matrix using the user's secret and public values. The matrix calculation is performed only once with the addition or deletion of a row and column when a user joins or leaves a group. However, in real time, on systems with better configuration, the time taken for the matrix calculation will naturally be reduced.



```
E:\program\Key management>java dataowner
prime number is:23
l-bit of prime number:5

Enter the number of users :
3
key space is:

        1       2       3       4       5       6       7       8       9
10      11      12      13      14      15      16      17      18      19
20      21      22

secret space is:

        1       2       3       4       5       6       7       8       9
10      11      12      13      14      15      16      17      18      19
20      21      22      23      24      25      26      27      28      29
30      31
secret key of user 1 is : 2

secret key of user 2 is : 4

secret key of user 3 is : 9

public key of user 1 is : 5

public key of user 2 is : 7

public key of user 3 is : 2

the matrix is:

1       7       9       4
1       9       11      6
1       14      16      11

Time to compute matrix 7346863739497 nano sec
```

Figure 5-4: Computation of n×(n+1) matrix by the data owner

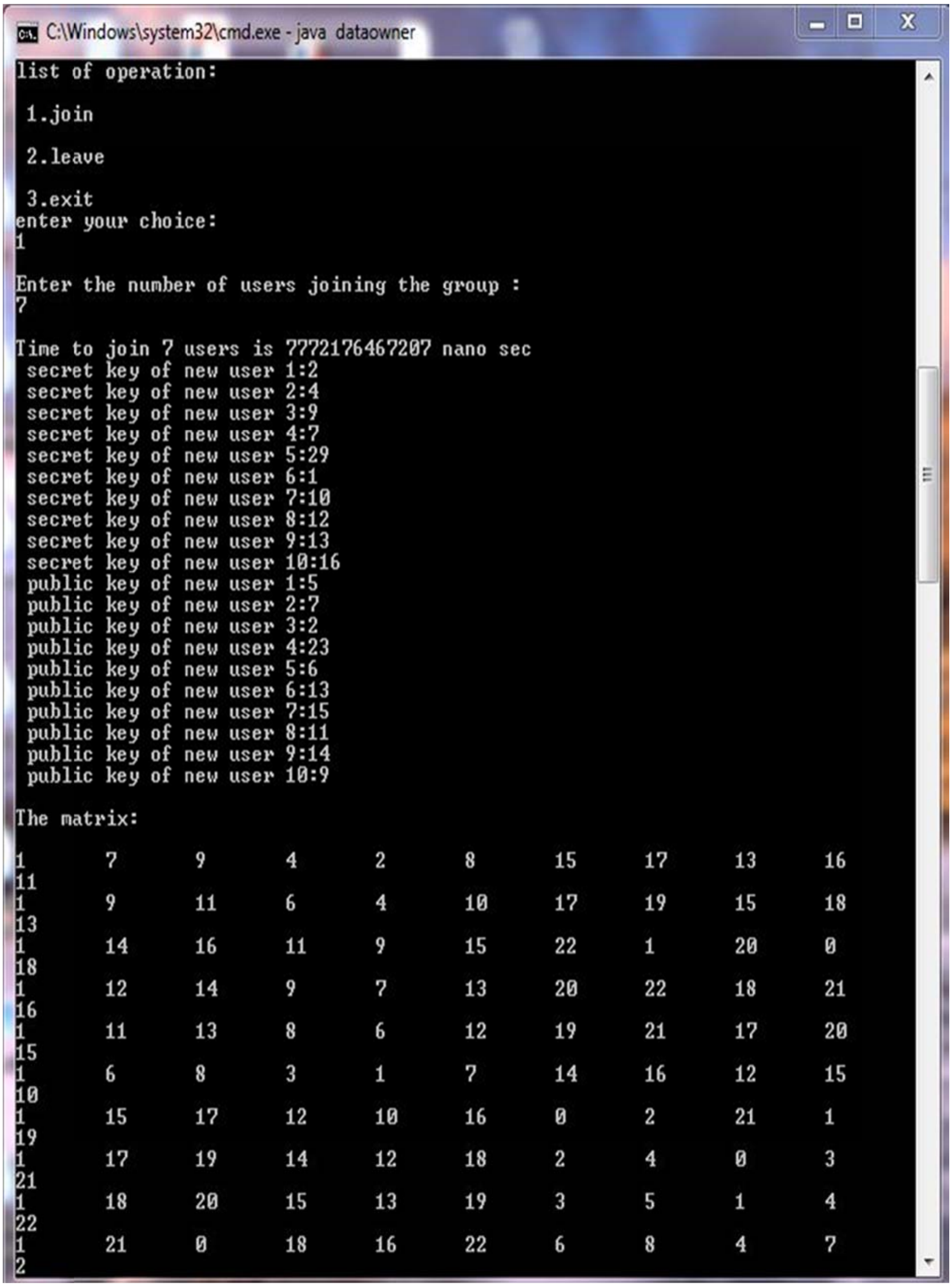
Figure 5-5 shows that the data owner computes the access control vector to be sent to the cloud users from n ×(n+1) matrix. After computing the ACV value, the data owner updates the ACV value whenever the group membership of cloud users changes. This calculation is performed only once with the addition or deletion of a row and column when a user joins or leaves a group. However, in real time, on systems with better configuration, the time taken for the matrix calculation will naturally be reduced.



```
 Gauss Elimination Starts:
Enter Co-efficient 1 : 1
Enter Co-efficient 2 : 7
Enter Co-efficient 3 : 9
Enter Constant Term : 4
Enter Co-efficient 1 : 1
Enter Co-efficient 2 : 9
Enter Co-efficient 3 : 11
Enter Constant Term : 6
Enter Co-efficient 1 : 1
Enter Co-efficient 2 : 14
Enter Co-efficient 3 : 16
Enter Constant Term : 11
 +1A  +7B  +9C  = 4
 +1A  +9B  +11C  = 6
 +1A  +14B  +16C  = 11

Solution Set is :
A = 5
B = 0
C = -1
D = 1

 Gausss Elimination Ends/

column vector:
5
0
22
1

 choosen group key is:8

 Access Control Vector
13
0
22
1

Time to compute access control vector 7346899171448 nano sec

Do you want to change the group(y/n)
```

Figure 5-5: Computation of access control vector by the data owner

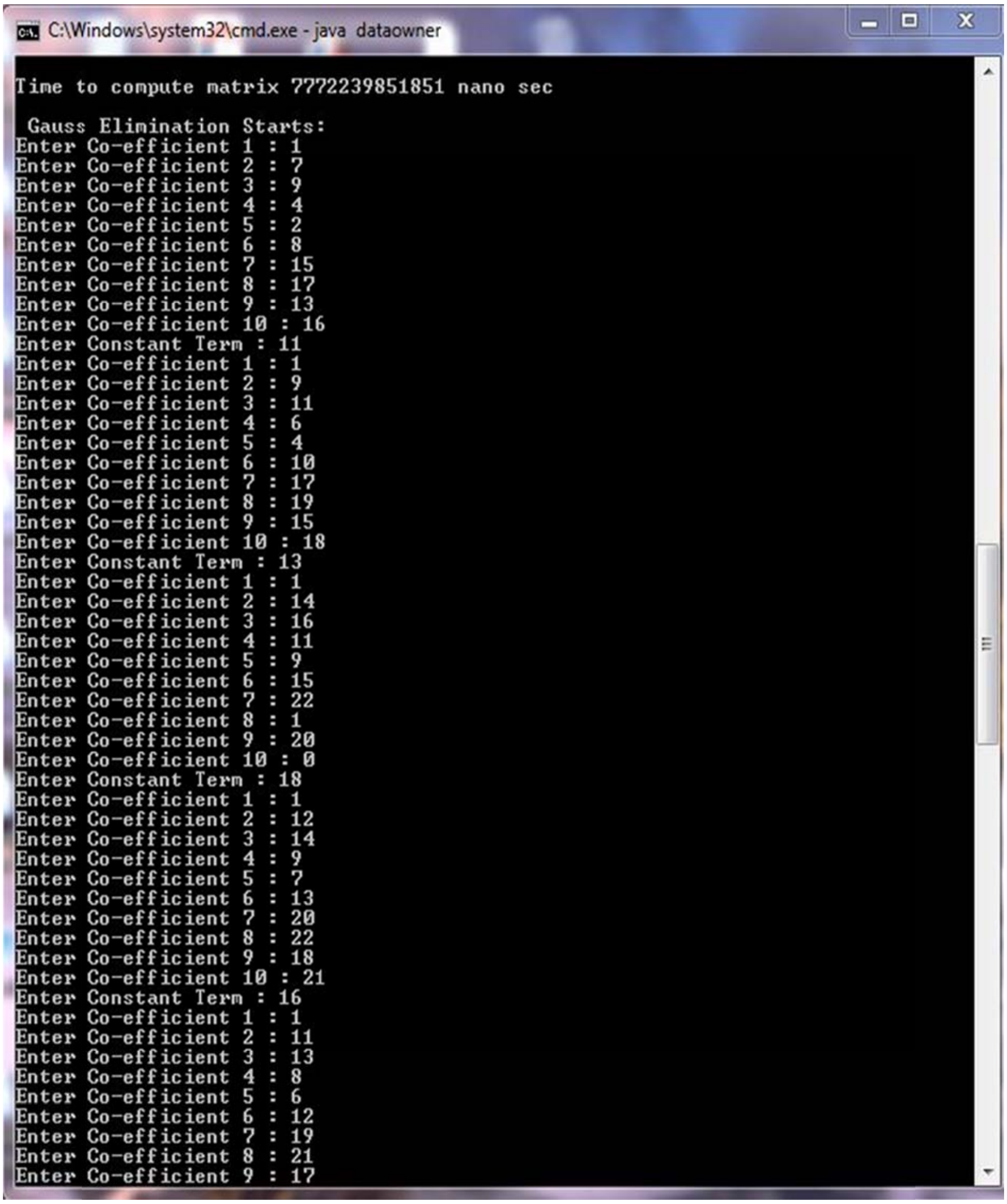
As more users joins in the group, the data owner increases the input key matrix size to produce a modified ACV value to provide backward secrecy. Figure 5-6 shows the matrix after new users joining the group. Similarly, the data owner eliminates a row and a column in the key matrix when a user leaves the group. For example, if a user u_i leaves from a group, then its corresponding row and column vector will be eliminated from the matrix. Therefore, in this case, i^{th} row and j^{th} column vector will be eliminated from the matrix.



```
C:\Windows\system32\cmd.exe - java dataowner
list of operation:

 1.join

 2.leave

 3.exit
enter your choice:
1

Enter the number of users joining the group :
7

Time to join 7 users is 772176467207 nano sec
 secret key of new user 1:2
 secret key of new user 2:4
 secret key of new user 3:9
 secret key of new user 4:7
 secret key of new user 5:29
 secret key of new user 6:1
 secret key of new user 7:10
 secret key of new user 8:12
 secret key of new user 9:13
 secret key of new user 10:16
 public key of new user 1:5
 public key of new user 2:7
 public key of new user 3:2
 public key of new user 4:23
 public key of new user 5:6
 public key of new user 6:13
 public key of new user 7:15
 public key of new user 8:11
 public key of new user 9:14
 public key of new user 10:9

The matrix:

1       7       9       4       2       8       15      17      13      16
11
1       9       11      6       4       10      17      19      15      18
13
1       14      16      11      9       15      22      1       20      0
18
1       12      14      9       7       13      20      22      18      21
16
1       11      13      8       6       12      19      21      17      20
15
1       6       8       3       1       7       14      16      12      15
10
1       15      17      12      10      16      0       2       21      1
19
1       17      19      14      12      18      2       4       0       3
21
1       18      20      15      13      19      3       5       1       4
22
1       21      0       18      16      22      6       8       4       7
2
```

Figure 5-6: New user joining the group

After finalizing the key matrix, the data owner computes the coefficient values to find the inverse matrix using the Gauss elimination method. Figure 5-7 shows the coefficient values produced using the Gauss elimination method. The generated coefficient values are used to find the ACV values.

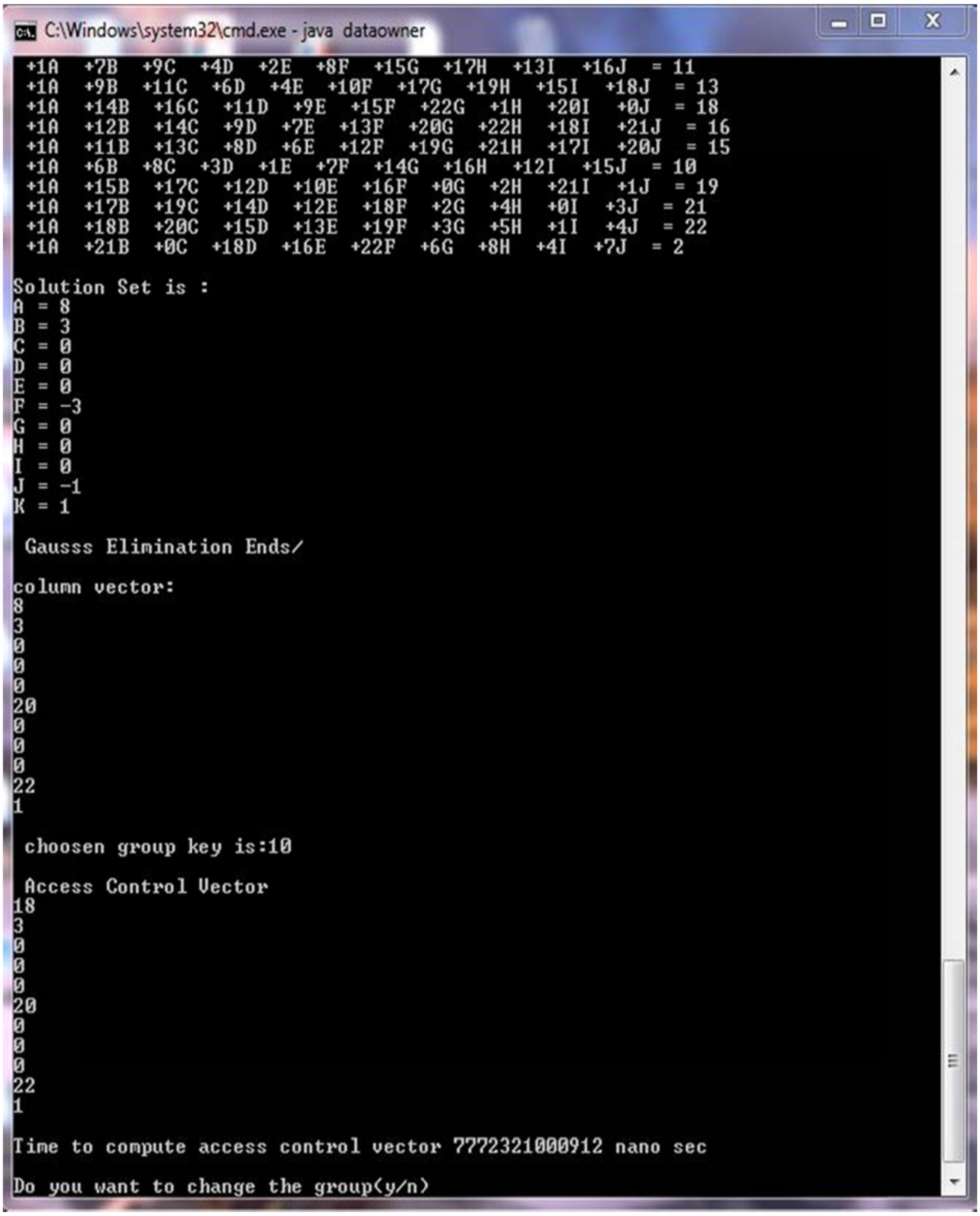


C:\Windows\system32\cmd.exe - java dataowner

```
Time to compute matrix 7772239851851 nano sec

 Gauss Elimination Starts:
Enter Co-efficient 1 : 1
Enter Co-efficient 2 : 7
Enter Co-efficient 3 : 9
Enter Co-efficient 4 : 4
Enter Co-efficient 5 : 2
Enter Co-efficient 6 : 8
Enter Co-efficient 7 : 15
Enter Co-efficient 8 : 17
Enter Co-efficient 9 : 13
Enter Co-efficient 10 : 16
Enter Constant Term : 11
Enter Co-efficient 1 : 1
Enter Co-efficient 2 : 9
Enter Co-efficient 3 : 11
Enter Co-efficient 4 : 6
Enter Co-efficient 5 : 4
Enter Co-efficient 6 : 10
Enter Co-efficient 7 : 17
Enter Co-efficient 8 : 19
Enter Co-efficient 9 : 15
Enter Co-efficient 10 : 18
Enter Constant Term : 13
Enter Co-efficient 1 : 1
Enter Co-efficient 2 : 14
Enter Co-efficient 3 : 16
Enter Co-efficient 4 : 11
Enter Co-efficient 5 : 9
Enter Co-efficient 6 : 15
Enter Co-efficient 7 : 22
Enter Co-efficient 8 : 1
Enter Co-efficient 9 : 20
Enter Co-efficient 10 : 0
Enter Constant Term : 18
Enter Co-efficient 1 : 1
Enter Co-efficient 2 : 12
Enter Co-efficient 3 : 14
Enter Co-efficient 4 : 9
Enter Co-efficient 5 : 7
Enter Co-efficient 6 : 13
Enter Co-efficient 7 : 20
Enter Co-efficient 8 : 22
Enter Co-efficient 9 : 18
Enter Co-efficient 10 : 21
Enter Constant Term : 16
Enter Co-efficient 1 : 1
Enter Co-efficient 2 : 11
Enter Co-efficient 3 : 13
Enter Co-efficient 4 : 8
Enter Co-efficient 5 : 6
Enter Co-efficient 6 : 12
Enter Co-efficient 7 : 19
Enter Co-efficient 8 : 21
Enter Co-efficient 9 : 17
```

Figure 5-7: The Gauss elimination for the new matrix

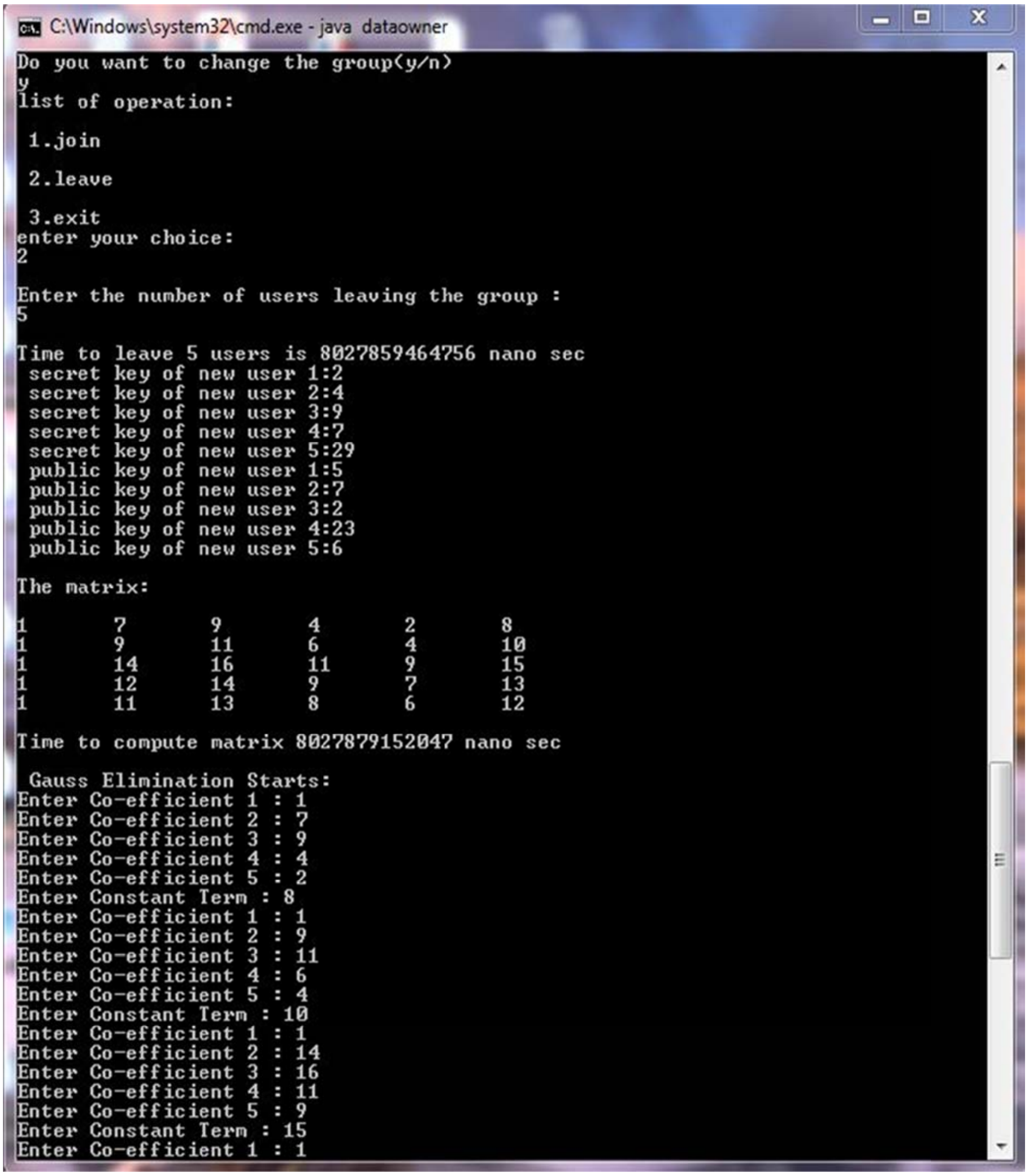
Figure 5-8 shows the access control vector when 10 new users have joined the group. This ACV will be sent to all the existing cloud users so that they can each derive the GK that can be used to decrypt documents accessible by this group.



```
C:\Windows\system32\cmd.exe - java  dataowner
 +1A   +7B   +9C   +4D   +2E   +8F   +15G   +17H   +13I   +16J   = 11
 +1A   +9B   +11C   +6D   +4E   +10F   +17G   +19H   +15I   +18J   = 13
 +1A   +14B   +16C   +11D   +9E   +15F   +22G   +1H   +20I   +0J   = 18
 +1A   +12B   +14C   +9D   +7E   +13F   +20G   +22H   +18I   +21J   = 16
 +1A   +11B   +13C   +8D   +6E   +12F   +19G   +21H   +17I   +20J   = 15
 +1A   +6B   +8C   +3D   +1E   +7F   +14G   +16H   +12I   +15J   = 10
 +1A   +15B   +17C   +12D   +10E   +16F   +0G   +2H   +21I   +1J   = 19
 +1A   +17B   +19C   +14D   +12E   +18F   +2G   +4H   +0I   +3J   = 21
 +1A   +18B   +20C   +15D   +13E   +19F   +3G   +5H   +1I   +4J   = 22
 +1A   +21B   +0C   +18D   +16E   +22F   +6G   +8H   +4I   +7J   = 2

Solution Set is :
A = 8
B = 3
C = 0
D = 0
E = 0
F = -3
G = 0
H = 0
I = 0
J = -1
K = 1

 Gausss Elimination Ends/

column vector:
8
3
0
0
0
20
0
0
0
22
1

 choosen group key is:10

 Access Control Vector
18
3
0
0
0
20
0
0
0
22
1

Time to compute access control vector 7772321000912 nano sec

Do you want to change the group(y/n)
```

Figure 5-8: The access control vector for the new matrix

Similar to the user join operation, users can also leave an existing group, thus the data owner should modify the ACV value in order to prevent the new user from viewing the past documents. This provides backward secrecy. Figure 5-9 shows the new matrix when 5 users leave the existing group.

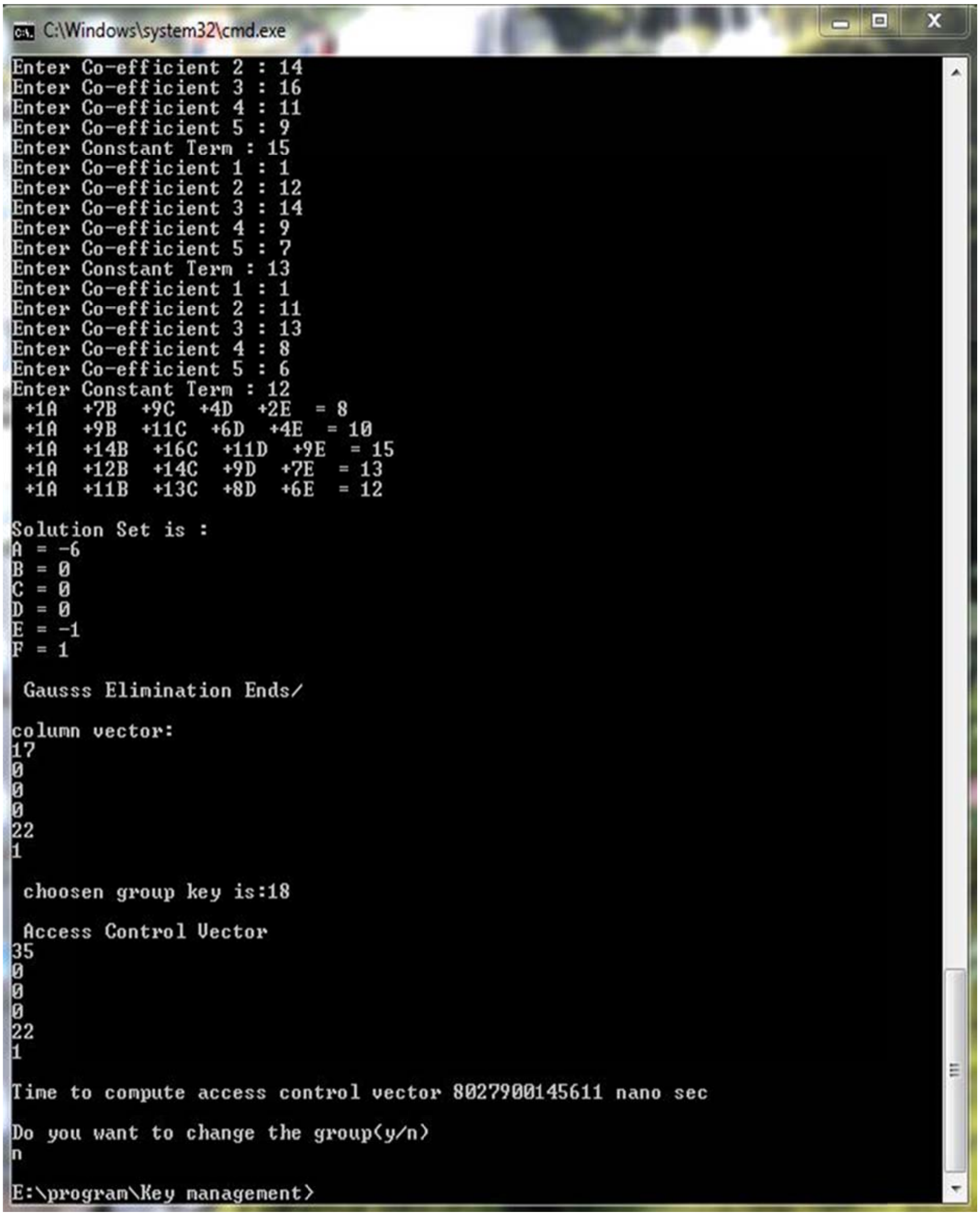


C:\Windows\system32\cmd.exe - java dataowner

```
Do you want to change the group(y/n)
y
list of operation:

 1.join

 2.leave

 3.exit
enter your choice:
2

Enter the number of users leaving the group :
5

Time to leave 5 users is 8027859464756 nano sec
 secret key of new user 1:2
 secret key of new user 2:4
 secret key of new user 3:9
 secret key of new user 4:7
 secret key of new user 5:29
 public key of new user 1:5
 public key of new user 2:7
 public key of new user 3:2
 public key of new user 4:23
 public key of new user 5:6

The matrix:

1       7       9       4       2       8
1       9       11      6       4       10
1       14      16      11      9       15
1       12      14      9       7       13
1       11      13      8       6       12

Time to compute matrix 8027879152047 nano sec

 Gauss Elimination Starts:
Enter Co-efficient 1 : 1
Enter Co-efficient 2 : 7
Enter Co-efficient 3 : 9
Enter Co-efficient 4 : 4
Enter Co-efficient 5 : 2
Enter Constant Term : 8
Enter Co-efficient 1 : 1
Enter Co-efficient 2 : 9
Enter Co-efficient 3 : 11
Enter Co-efficient 4 : 6
Enter Co-efficient 5 : 4
Enter Constant Term : 10
Enter Co-efficient 1 : 1
Enter Co-efficient 2 : 14
Enter Co-efficient 3 : 16
Enter Co-efficient 4 : 11
Enter Co-efficient 5 : 9
Enter Constant Term : 15
Enter Co-efficient 1 : 1
```

Figure 5-9: User leaves from the group

Figure 5-10 shows the newly computed access control vector when 10 users have left the group.

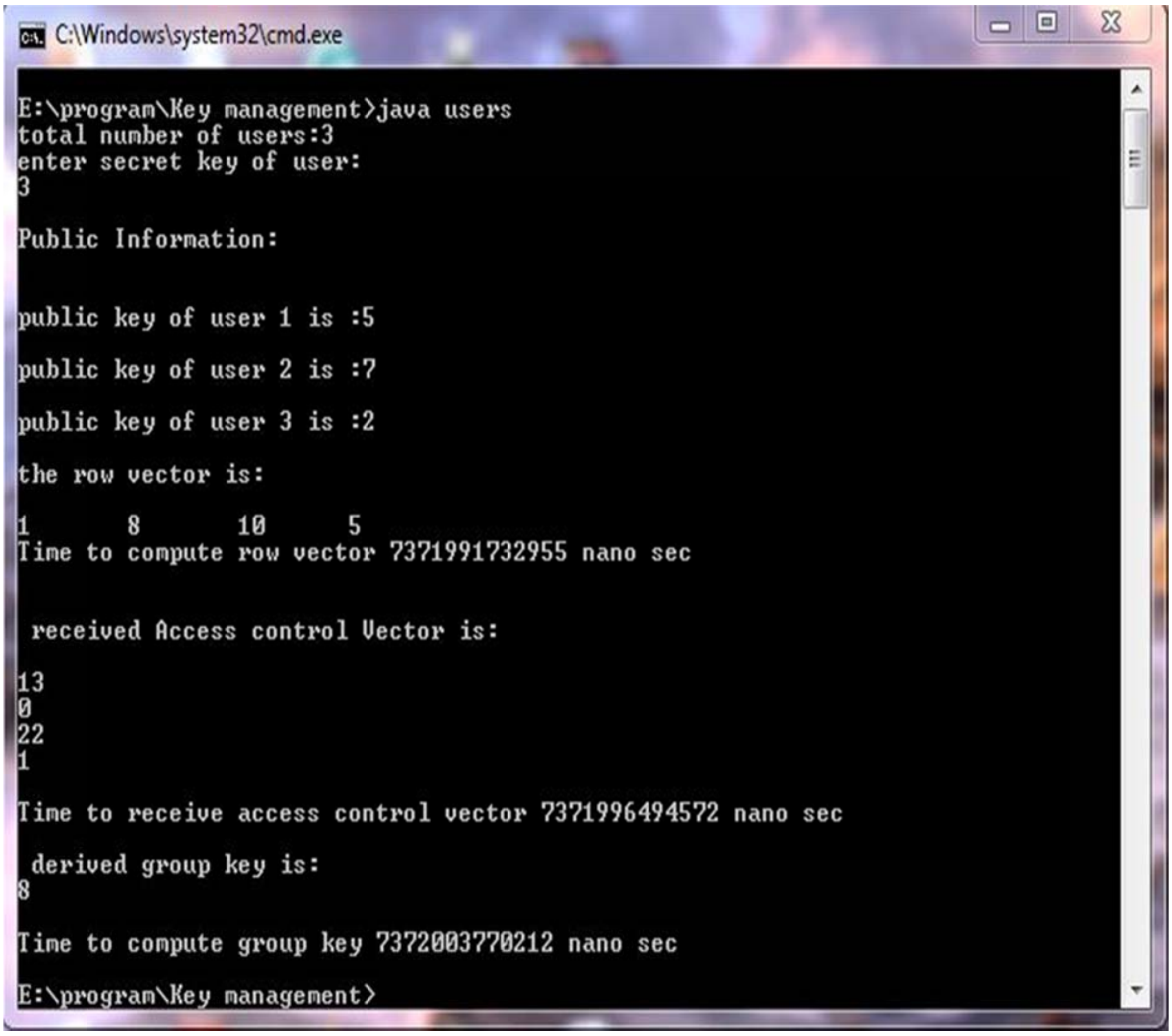


```
C:\Windows\system32\cmd.exe
Enter Co-efficient 2 : 14
Enter Co-efficient 3 : 16
Enter Co-efficient 4 : 11
Enter Co-efficient 5 : 9
Enter Constant Term : 15
Enter Co-efficient 1 : 1
Enter Co-efficient 2 : 12
Enter Co-efficient 3 : 14
Enter Co-efficient 4 : 9
Enter Co-efficient 5 : 7
Enter Constant Term : 13
Enter Co-efficient 1 : 1
Enter Co-efficient 2 : 11
Enter Co-efficient 3 : 13
Enter Co-efficient 4 : 8
Enter Co-efficient 5 : 6
Enter Constant Term : 12
 +1A  +7B  +9C  +4D  +2E  = 8
 +1A  +9B  +11C  +6D  +4E  = 10
 +1A  +14B  +16C  +11D  +9E  = 15
 +1A  +12B  +14C  +9D  +7E  = 13
 +1A  +11B  +13C  +8D  +6E  = 12

Solution Set is :
A = -6
B = 0
C = 0
D = 0
E = -1
F = 1

 Gausss Elimination Ends/

column vector:
17
0
0
0
22
1

 choosen group key is:18

 Access Control Vector
35
0
0
0
22
1

Time to compute access control vector 8027900145611 nano sec

Do you want to change the group(y/n)
n

E:\program\Key management>
```

Figure 5-10: New access control vector

5.2.2 Group key derivation

In this phase, the cloud users are deriving their own group key using their own row vector, which is computed using each users secret and all users public key. Figure 5-11 shows that the user derives the GK using the public information and the user's secret key value.



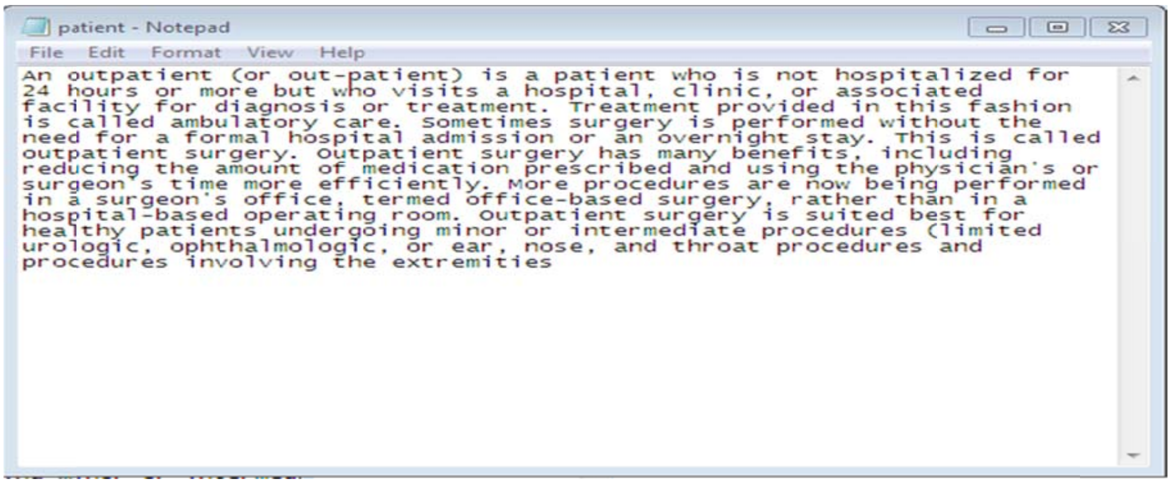
```
E:\program\Key management>java users
total number of users:3
enter secret key of user:
3

Public Information:


public key of user 1 is :5

public key of user 2 is :7

public key of user 3 is :2

the row vector is:

1       8       10      5
Time to compute row vector 7371991732955 nano sec


 received Access control Vector is:

13
0
22
1

Time to receive access control vector 7371996494572 nano sec

 derived group key is:
8

Time to compute group key 7372003770212 nano sec

E:\program\Key management>
```

Figure 5-11: Derivation of Group Key by the user

5.3 Document Management Module

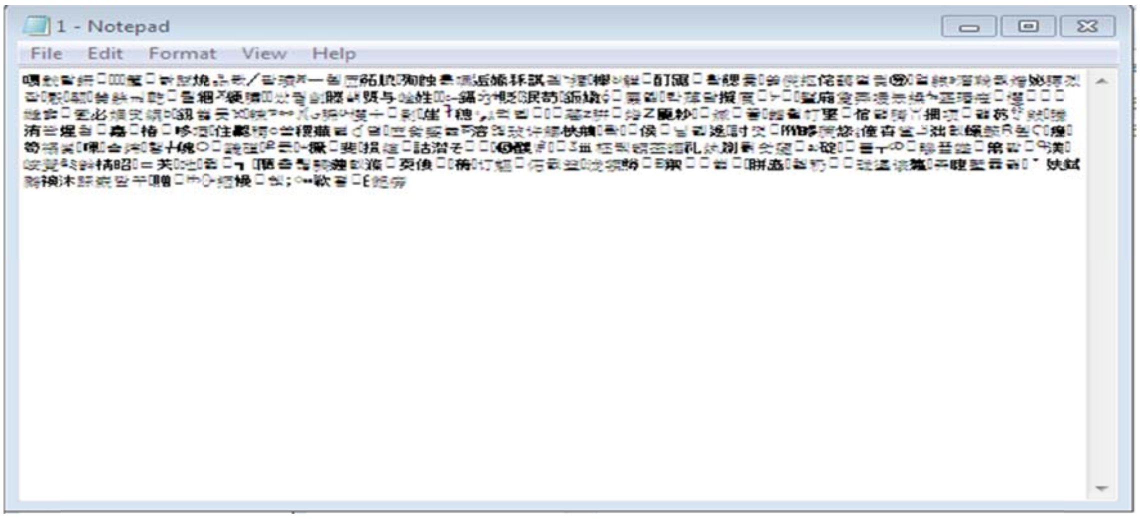
In this module, the data owner encrypts different documents for different groups and places them in the cloud. In this thesis, both the data owner and CSP are located in the same physical computer system. For simplicity, we use a text document containing information about a patient, which should be accessible to anyone in the group “doctor”. This text document is encrypted using the GK computed and derived in the previous module. Therefore, all doctors who belong to this group can view this document by decrypting the document from the cloud using that particular GK. The input to this module are a GK and a document. The output of this module is the encrypted document to be placed in the cloud and shared with the cloud users whom are doctors.

Figure 5-12 shows the original document before encryption. Figure 5-13 shows the encrypted document after encrypting it using DES. Figure 5-14 shows the decrypted document with the valid GK. Figure 5-15 shows the decrypted document when decrypted with an invalid group key.



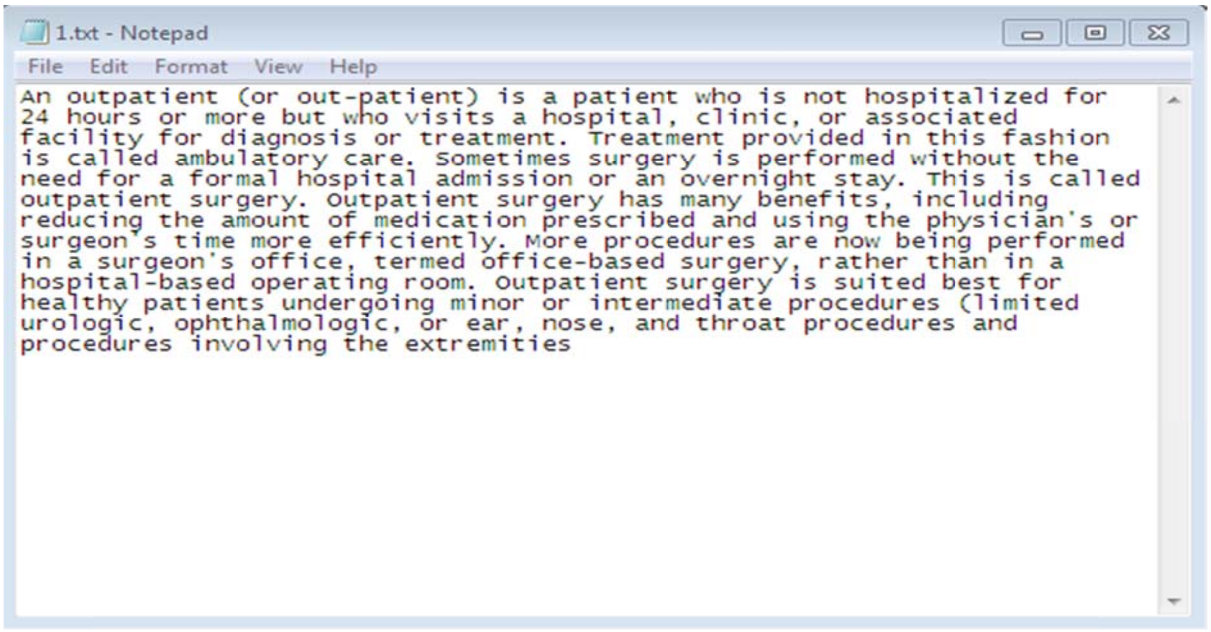
```
An outpatient (or out-patient) is a patient who is not hospitalized for
24 hours or more but who visits a hospital, clinic, or associated
facility for diagnosis or treatment. Treatment provided in this fashion
is called ambulatory care. Sometimes surgery is performed without the
need for a formal hospital admission or an overnight stay. This is called
outpatient surgery. Outpatient surgery has many benefits, including
reducing the amount of medication prescribed and using the physician's or
surgeon's time more efficiently. More procedures are now being performed
in a surgeon's office, termed office-based surgery, rather than in a
hospital-based operating room. Outpatient surgery is suited best for
healthy patients undergoing minor or intermediate procedures (limited
urologic, ophthalmologic, or ear, nose, and throat procedures and
procedures involving the extremities
```

Figure 5-12: Original document



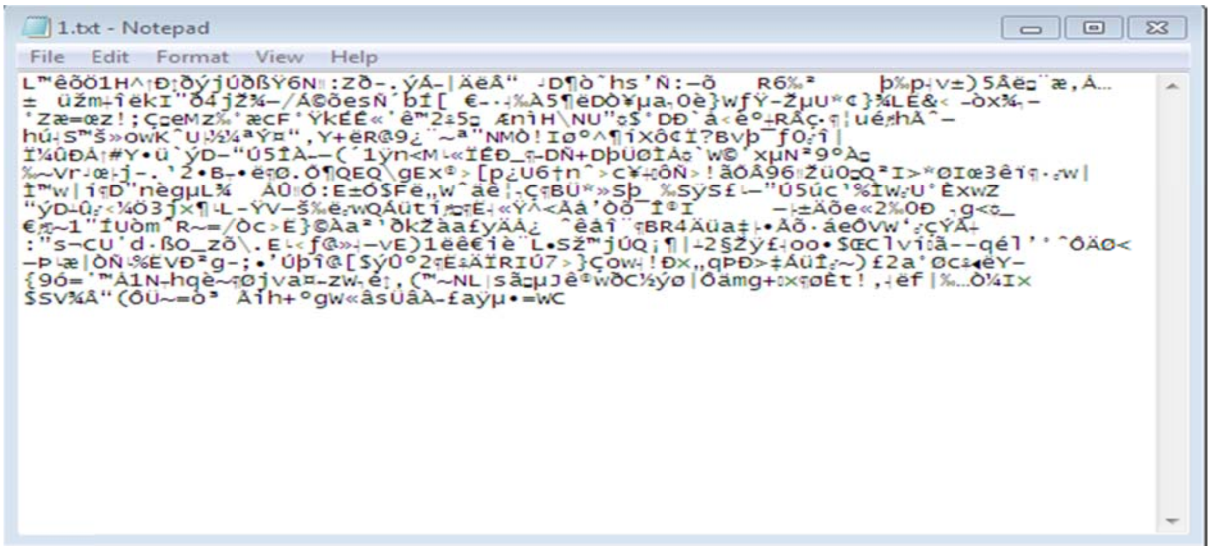
[illegible]

Figure 5-13: Encrypted document



```
An outpatient (or out-patient) is a patient who is not hospitalized for
24 hours or more but who visits a hospital, clinic, or associated
facility for diagnosis or treatment. Treatment provided in this fashion
is called ambulatory care. Sometimes surgery is performed without the
need for a formal hospital admission or an overnight stay. This is called
outpatient surgery. Outpatient surgery has many benefits, including
reducing the amount of medication prescribed and using the physician's or
surgeon's time more efficiently. More procedures are now being performed
in a surgeon's office, termed office-based surgery, rather than in a
hospital-based operating room. Outpatient surgery is suited best for
healthy patients undergoing minor or intermediate procedures (limited
urologic, ophthalmologic, or ear, nose, and throat procedures and
procedures involving the extremities
```

Figure 5-14: Decrypted document with the valid group key



```
L™êõÖ1H^↑Ð↕ðýjÚðßŸ6N‼:Zð-.ýÁ-|ÄëÂ“ ┘D¶ò˜hs’Ñ:–õ   R6‰²      þ‰p┤v±)5Âë▬¨æ,Á…
± üžm┼îëkI"ð4jŽ¾–/Á©õesÑ´bÎ[ €-·┤‰À5¶ëDÒ¥µa┐0è}wƒŸ-ŽµU*¢}¾LÉ&< –òx¾┐–
'Zæ=œz!;Ç▬eMz‰'æcF'ŸkÉÊ«'ê™2±5▬ ÆnìH\NU"☼$'DÐ`á<ê°┼RÂç·¶¦ué♯hÃˆ–
hú┤S™š»owK^U┼½¼ªÝ¤“,Y+ëR@9¿¨~ª"NMÒ!Iø°^¶íXô¢Ï?Bvþ¯ƒ0♯î|
Ï¼ÛÐÂ↑#Y•ü`ýD–“Ú5ÎÀ––(´1ÿn<M└«ÏÊÐ_¶-DÑ+DþÜØÎÂ☼`W©´xµN²9°Â▬
‰~Vr┘œ┼j–.'2•B┬•ë¶Ø.Õ¶QEQ\gEx®>[p¿U6†n^>c¥┼‼ôÑ>!ãÕÅ96‼Žü0▬Q²I>*ØIœ3êï¶·♯w|
Î™w|í¶D"nègµL¾  ÁÛ‼Ó:E±Ó$Fë„w^áè¦┐Ç¶BÜ*»Sþ_‰SÿS£└–"Ú5úc'%ÎW♯U'ÈxwZ
“ýD┴Û♯<¼Ö3ĵ×¶└L–ŸV–š‰ë♯wQÁüti♯☼¶E┤«Ÿ^<Åá’Òõ¯Î®I      –┼±Äõe«2‰0Ð ┐g<☼_
€♯~1"ÍUòm˜R~=/Òc>E}©Àa²'ðkŽàa£yÄÂ¿ ^êáî¨¶BR4Äüa‡┼•Åõ·áeÔvw‘♯çÝÄ┼
:"s¬cU´d·ßO_zõ\.E└<ƒ@»┤–vE)1ëê€ïè¨L•Sž™jÚQ¡¶|┴2§Žÿ£┤oo•$ŒClví‼ã––qél’'^ÔÄØ<
–Þ└æ|ÒÑ└%ÉVÐ²g–;•’Úþî@[$ýÛ°2¶E±ÄÏRIÚ7>}Çow┤!Ðx„qPÐ>‡ÁüÎ♯~)£2a'Øc±◄ëY–
{9ó=´™Å1N┬hqè~¶Øjva¤–zW┐é↕,(™~NL|sã▬µJê®wðC½ýø|Õämg+‼×¶øÈt!,┤ëf|‰…Ò¼I×
$Sv¾Å“(ÔÜ~=ò³ Åîh+°gw«âsÚâÀ–£aÿµ•=wC
```

Figure 5-15: Decrypted document with the invalid group key

6 Results and Discussions

The proposed method has been implemented in Java running on two computers, configured as per Table 5-1, for a maximum of 10 users in each group, when a new user is added a new bucket is formed to accommodate the users joining, yet still the maximum number of users in the bucket will be limited to 10. one sub group can form a functional group for the data owner, which can be handled by generating the same commitments for the groups by the token generator. The proposed broadcast key management can support hundreds of cloud users in each group. However, this broadcast key management takes $o(n^3)$ computation time to solve n linear equations in order to find the inverse matrix using the Gauss-Jordan elimination method. In order to improve this, we have used bucket based approach that divides the number of users into buckets of 10 users when the number of users exceeds more than 100 users in order to improve efficiency. This proved to be necessary since computing the inverse matrix for a 100×100 matrix is a time consuming task. In addition, when bucket based approach is used to improve the efficiency, we use a two-level approach in which the 0^{th} level is the group key, the 1^{st} level is a Sub Group key (SGk) or intermediate key, and the 2^{nd} level is the individual user's secret key. The user's secret key is used to find the sub group key and the subgroup key is used to find the group key by the cloud user.

The tree shown in Figure 6-1 consists of only two levels. The first level (0th level) is the group key and the second level (1st level) contains the sub group keys, SGk_i where $i = 1, 2, \ldots, n$. Each matrix in a bucket is attached to the upper level (1st level) node and in turn with the group key node. When the number of joining users exceeds the bucket size, a new node is created from the root to form the second bucket. The number of buckets formed is based on the matrix size M, which is fixed, by the data owner and the number of joining users. If the bucket based key management consists of N number of users $u_1, u_2, \ldots, u_N$ and each bucket is of size M then there will be $\lceil N/M \rceil$ buckets in the leaf node of the tree. In this bucket based key management scheme, updating is necessary for each rekeying operation used for user leave and user join operations.

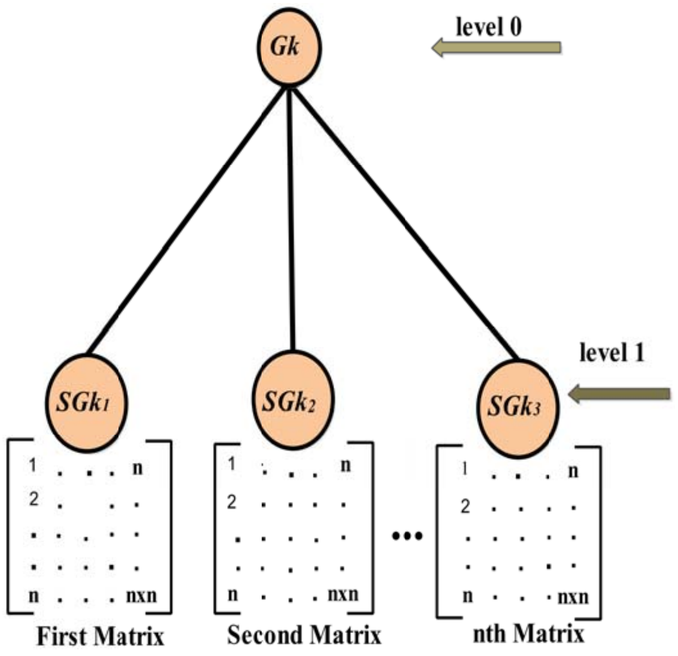


Figure 6-1: Logical View of Bucket based Key Management

For example, if a user u_i from the first bucket in Figure 6-1 leaves the group, then the keys on the path from his/her leaf node to the tree's root node must be changed. Hence, only the keys SGk_i and Gk will become invalid. Therefore, only these two keys must be updated. In order to update these two keys, two approaches are used in the user leave and join operations. In the first approach, updating of the sub group key, SGk_1 for the First Matrix (bucket) is performed as explained in section 4. After updating the SGk_1 successfully, the data owner uses the second approach in order to update the group key Gk using a different procedure as explained in the existing approach by Vijaykumar [80]. The new group key Gk is used to encrypt the documents that are placed in the public cloud. To update Gk, the data owner generates a new group key from z_p^*, with a condition that the new group key $\mathrm{Gk}^1 < \mathrm{SGk}_i$. If this condition is not satisfied then it appends a value 1 in front of SGk_i in order to make SGk_i greater value than Gk^1. So that when a modulo division operation used in the cloud user side, the cloud user can find the group key.

To update the group key, the value of X (the product of all SGk s) is added to the newly generated group key Gk^1 to obtain the group key and the rekeying message is formed by using the equation $\gamma_g = \mathrm{Gk}^1 + \mathrm{X}$. In this way, member leave operations are handled effectively (as we reduce the number of multiplication/divisions). The resultant value γ_g is broadcast to the remaining cloud users. The remaining cloud users of the group can recover the updated group key with the help of their sub group key using the relation, $\gamma_g \bmod (\mathrm{SGk}_i) = \mathrm{Gk}^1$.

If a cloud user with a pseudonym wants to view a particular subdocument S1, then this user should download the encrypted subdocument from the CSP along with the public information. After that, it selects an ACP_k that it satisfies and then derives the group key. In order to derive the group key, the cloud user first derives the intermediate keys from which it can compute the group key. In this example, each cloud user derives only one intermediate key and one group key to perform the decryption operation in order to view the documents.

The group key computation time has been analysed for each of the existing wired group key management approaches to perform the key updating operation by the data owner. Table 6-1 compares the results obtained from the proposed Broadcast Key Management for Public Cloud (BKMPC) with the existing centralized group key management schemes MDS [16], SKDC [24], OFT [27], Binary [26], ELK [38], LKH [81],and Vijayakumar et al.'s KKD [82]. Table 6-1 shows the measured computation time in nanoseconds (ns) taken to compute the access control vector to be given to the cloud users in order to allow them to each compute GK. The operation is performed 10 times and we have taken the average of all the 10 computation times to measure the actual computation time for the cloud users. The average of the 10 runs is calculated and it is included in the table. An average of 10 runs was chosen, since the runs didn't have a significant leap in the computational time.

Table 6-2 shows the measured computation time taken to derive the group key by each user. From

Table 6-2, it is very clear that our proposed key management takes less computation time for the user.

Table 6-1: Group key computation time in ns for various key distribution approaches

Number of users	Binary Tree	ETF	MDS	ELK	LHK	SKDC	OFT	KKD	BKMPC
1	1098458	868635	974632	671940	771940	1171940	506873	506773	129822
2	498030	375151	512546	437103	537103	937103	272036	172036	137934
3	640457	298495	640023	507243	607243	1007243	342176	343176	142236
4	433325	149545	423040	450312	550312	950312	285245	284245	158345
5	419834	296490	417653	349670	449670	849670	184603	183603	165517
6	350018	343275	326578	407737	507737	907737	242670	232670	172293
7	332463	510320	504567	608166	708166	1108166	443099	433099	173693
8	276145	496499	464343	406979	506979	906979	241912	331912	179368
9	245789	487957	458975	389754	478973	879478	234578	298754	186098
10	219875	435789	435278	357898	457898	857894	213457	278954	193317

Table 6-2: Group key derivation time in ns for various key distribution approaches

Number of users	Binary Tree	ETF	MDS	ELK	LHK	SKDC	OFT	KKD	BKMPC
1	298458	468635	474632	371940	771940	1171940	426873	176873	130929
2	288030	455151	512546	397103	637103	937103	372036	180036	138323
3	340457	548495	640023	457243	607243	1007243	342176	290176	142471
4	333325	539545	723040	470312	550312	950312	285245	237245	160766
5	319834	636490	787653	499670	449670	849670	274603	243603	166720
6	250018	643275	826578	507737	407737	907737	273670	252670	170576
7	232463	730320	884567	568166	408166	898166	273099	263099	174795
8	226145	496499	894343	606979	506979	886979	271912	291912	180843
9	215789	487957	908975	689754	478973	879478	254578	238754	187563
10	215875	435789	935278	757898	457898	857894	243457	258954	194359

It is evident from the values that the group key computation time for the proposed broadcast key management is better both for the data owner and the cloud user than the other algorithms used in centralized wired networks. This improvement is achieved because the proposed method only adds the GK value and then performs a matrix inversion as per the equation, $X = (K \cdot e_1^T) + Y$. Therefore, only one addition operation is performed in the group key updating process. The cost of matrix inversion for Gauss-Jordan matrix elimination method is $O(n^3)$. In contrast, in the existing key management schemes, the keys are updated from the leaf to root node when using a tree based approach (MDS, SKDC, OFT, Binary, ELK, LKH, and KKD). Hence, the number of keys to be updated in the existing key management schemes is directly proportional to the height of the tree $\log_d N$, Where N= Number of users and d= degree of the tree. The fanout of the tree in real time would on depend on the number of users and groups for the application.

Table 6-3 shows the computation times, measured in milliseconds (ms), for various mathematical functions. When compared with all other functions, the multiplicative inverse operation takes more computation time. This time is important because it is used in many existing key management schemes. However, this multiplicative inverse operation is not used in our proposed key management scheme and hence the proposed key management scheme takes less computation time. Moreover, each user also performs only one vector multiplication and hence the user's computational effort is also minimized in this approach.

Table 6-3: Computation time complexities of various functions

	16 bit (ms)	32 bit (ms)	64 bit (ms)
Mod	2.8	3.1	3.2
Hash	2.9	3.9	4.6
Multiplication	4.8	5.0	5.4
Inverse	5.3	5.4	6.0

From Figure 6-2, it can be observed that when the group size is 10, the computation time for constructing the τmatrix of size 10×10 was 23.67 ms and the to derive the access control vector from the matrix was 19.33 ms in the proposed approach. If the number of members who are joining and leaving increases, then the computation time increases proportionately.

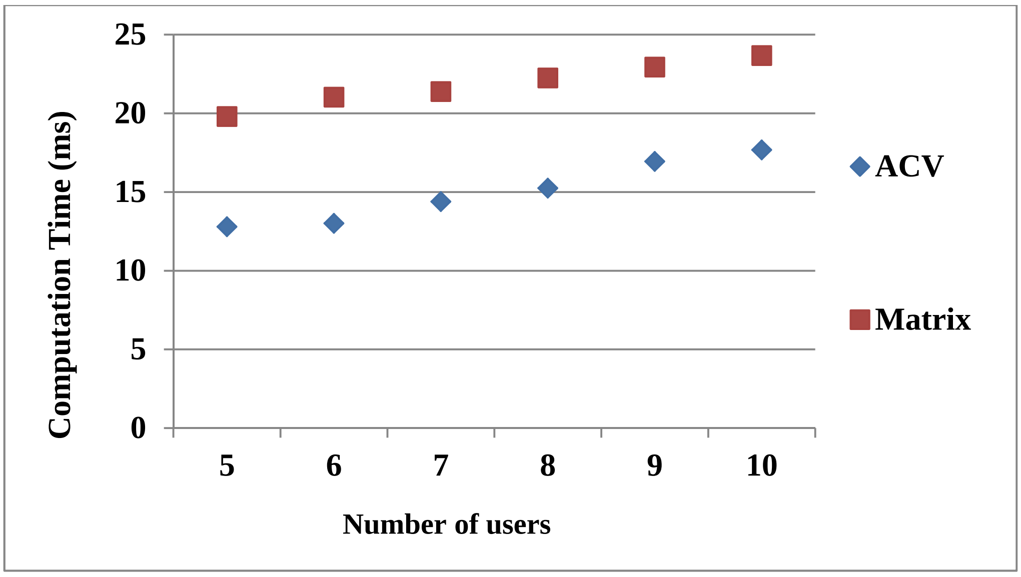


Figure 6-2: Computation time for the construction of n × (n+1) matrix and access control vector

We can observe in Figure 6-3 that when the group size is 10, the computation time for the group key updating by the data owner is 24.49 ms in the proposed approach, which is better than the other existing schemes that it was compared against.

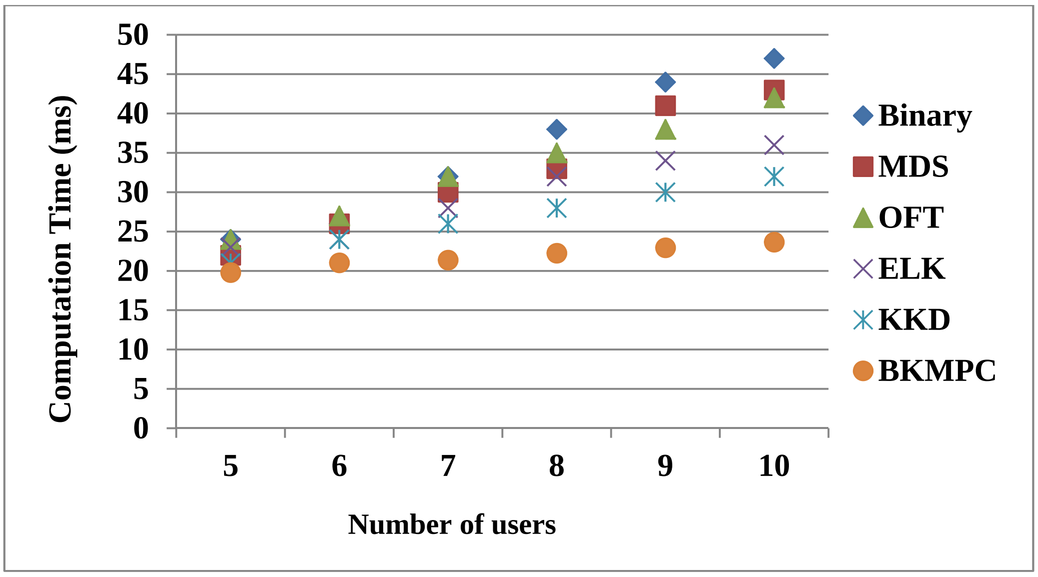


Figure 6-3: Computation time for the computation of group key by the data owner

From Figure 6-4, it can be observed that when the group size is 10, the computation time for the derivation of group key by the user is 22.45 ms in the proposed approach for updating the group key, which is better that for the other existing schemes that it has been compared against.

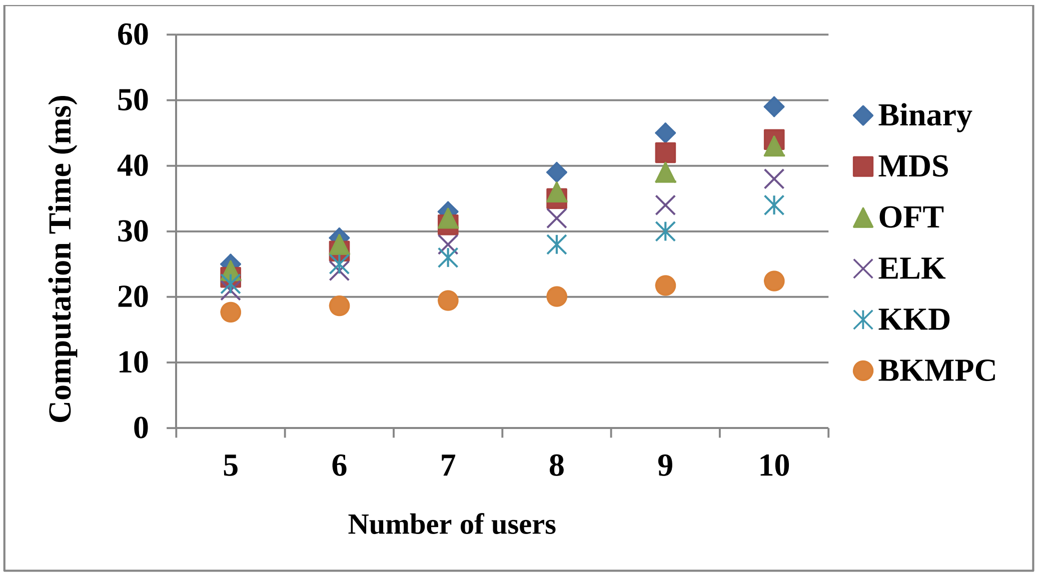


Figure 6-4: Computation time for the derivation of group key by the user

7 Conclusions and Future Work

This chapter concludes the thesis offering a conclusion and suggesting some future research. Section 7.1 states the computation and storage complexity of our proposed broadcast key management approach. Section 7.3 describes some of the social, economic, and ethical issues associated with this thesis project.

7.1 Conclusions

In this thesis project, a privacy preserving algorithm is implemented to improve the privacy preservation of each user who is accessing the data from a public cloud. In addition, a multicast key management scheme is implemented to provide an ACV for all the users who belong to a particular group. This ACV is used to compute a common group key to perform decryption of the document on the user's side. The proposed algorithm is computation, and communication efficient for the cloud user. The proposed key management algorithm is computationally efficient, since the data owner only needs to eliminate one row and one column when a single user leaves the group. After updating the full matrix the data owner needs to recomputed the inverse value to find the ACV and then distributed this ACV to all the users in this group.

In addition, each user performs only one row vector multiplication to recover the group key. The proposed scheme may also be used to perform batch-rekeying operations when a group of users joins or leaves the group at a time. The communication complexity of this proposed key management work is a single multicast of the public information (the full matrix). With this single multicast the data owner informs the group of users of the new AVC. The storage complexity of the proposed algorithm is O(2) since each user stores only one secret key and group key to access the documents from the cloud. However, the data owner needs to fetch, decrypt, and re-encrypt all of the documents when any change is to be made to the GK associated with this group of documents. Therefore, the data owner has to store all users' tokens, secret keys, and group key and hence the storage complexity of the data owner is $O(N)$. In addition to this, it also stores a matrix to compute the ACV.

7.2 Future Work

The multicast key management used in this research work distributes public information consisting of all the group's users' public key and ACV so that the individual group members can compute the GK. This would decrease the security level of group communication performed in the public cloud networks, because both internal and external attackers can get the ACV value from which they can try to find the group key value which is added in the first row of the ACV as explained in the section 4.2.6. Therefore, to improve the security, the ACV value should be encrypted such that the attacker should not able to derive the group key from the ACV. Moreover, if the ACV is lost (or) corrupted during the transmission, then all the group's users might not be able to find the GK. This is a challenging issue in implementing a multicast key management scheme. In order to avoid this, a reliable delivery mechanism should be integrated into the proposed key management algorithm.

The overlapping of documents and user groups is an area, which needs to be improved upon in the future.

7.3 Reflections

We have worked on this thesis with the aim of improving the privacy preservation and security of public cloud based storage solutions. We hope to have contributed a step forward towards the goal of achieving privacy and security for users, data owners, data and service providers on the public cloud.

References

[1] E. Bertino, N. Shang, and S. S. Wagstaff, “An Efficient Time-Bound Hierarchical Key Management Scheme for Secure Broadcasting,” *IEEE Trans. Dependable Secure Comput.*, vol. 5, no. 2, pp. 65–70, Apr. 2008.

[2] J. B. D. Joshi, R. Bhatti, E. Bertino, and A. Ghafoor, “Access-Control Language for Multidomain Environments,” *IEEE Internet Comput.*, vol. 8, no. 6, pp. 40–50, Nov. 2004.

[3] E. Bertino, P. A. Bonatti, and E. Ferrari, “TRBAC: A Temporal Role-based Access Control Model,” *ACM Trans Inf Syst Secur*, vol. 4, no. 3, pp. 191–233, Aug. 2001.

[4] J. A. M. Naranjo, N. Antequera, L. G. Casado, and J. A. López-Ramos, “A Suite of Algorithms for Key Distribution and Authentication in Centralized Secure Multicast Environments,” *J Comput Appl Math*, vol. 236, no. 12, pp. 3042–3051, Jun. 2012.

[5] S. Barker, “Action-status Access Control,” in *Proceedings of the 12th ACM Symposium on Access Control Models and Technologies*, New York, NY, USA, 2007, pp. 195–204.

[6] R. Poovendran and J. S. Baras, “An Information-theoretic Approach for Design and Analysis of Rooted-tree-based Multicast Key Management Schemes,” *IEEE Trans Inf Theor*, vol. 47, no. 7, pp. 2824–2834, Sep. 2006.

[7] M. Li, R. Poovendran, and D. . McGrewb, “Minimizing Center Key Storage in Hybrid One-Way Function Based Group Key Management with Communication Constraints,” *Elsevier*, vol. 93, no. 4, pp. 191–198, 2004.

[8] Y. Kim, A. Perrig, and G. Tsudik, “Group Key Agreement Efficient in Communication,” *IEEE Trans Comput*, vol. 53, no. 7, pp. 905–921, Jul. 2004.

[9] K. Drira, H. Seba, and H. Kheddouci, “ECGK: An Efficient Clustering Scheme for Group Key Management in MANETs,” *Comput Commun*, vol. 33, no. 9, pp. 1094–1107, Jun. 2010.

[10] Y.-S. Jeong, K.-S. Kim, Y. Kim, G. Park, and S.-H. Lee, “A Key Management Protocol for Securing Stability of an Intermediate Node in Wireless Sensor Networks,” in *IEEE 8th International Conference on Computer and Information Technology Workshops, 2008. CIT Workshops 2008*, 2008, pp. 471–476.

[11] J.-Y. Kim and H.-K. Choi, “Improvements on Sun ’s Conditional Access System in Pay-TV Broadcasting Systems,” *IEEE Trans. Multimed.*, vol. 12, no. 4, pp. 337–340, Jun. 2010.

[12] J.-Y. Kim and H.-K. Choi, “An efficient and versatile key management protocol for secure smart grid communications,” in *2012 IEEE Wireless Communications and Networking Conference (WCNC)*, 2012, pp. 1823–1828.

[13] X. S. Li, Y. R. Yang, M. G. Gouda, and S. S. Lam, “Batch Rekeying for Secure Group Communications,” in *Proceedings of the 10th International Conference on World Wide Web*, New York, NY, USA, 2001, pp. 525–534.

[14] X. B. Zhang, S. S. Lam, D.-Y. Lee, and Y. R. Yang, “Protocol design for scalable and reliable group rekeying,” *IEEEACM Trans. Netw.*, vol. 11, no. 6, pp. 908–922, Dec. 2003.

[15] W. H. D. Ng, M. Howarth, Z. Sun, and H. Cruickshank, “Dynamic Balanced Key Tree Management for Secure Multicast Communications,” *IEEE Trans. Comput.*, vol. 56, no. 5, pp. 590–605, May 2007.

[16] X. Lihao and C. Huang, “Computation-Efficient Multicast Key Distribution,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 5, pp. 1–10, 2008.

[17] H. Harney and C. Muckenhirn, “Group Key Management Protocol (GKMP) Architecture,” *Internet Req. Comments*, vol. RFC 2094 (Experimental), Jul. 1997.

[18] D. Harkins and D. Carrel, “The Internet Key Exchange (IKE),” Nov-1998. [Online]. Available: https://www.ietf.org/rfc/rfc2409.txt. [Accessed: 19-Mar-2014].

[19] D. Maughan and M. Schneider, “Internet Security Association and Key Management Protocol (ISAKMP).” [Online]. Available: http://tools.ietf.org/html/rfc2408. [Accessed: 19-Mar-2014].

[20] M. Li, I. Koutsopoulos, and R. Poovendran, “Optimal Jamming Attack Strategies and Network Defense Policies in Wireless Sensor Networks,” *IEEE Trans. Mob. Comput.*, vol. 9, no. 8, pp. 1119–1133, Aug. 2010.
[21] H. Lu, “A novel high-order tree for secure multicast key management,” *IEEE Trans. Comput.*, vol. 54, no. 2, pp. 214–224, Feb. 2005.
[22] R. Agee, D. Wallner, and E. Harder, “Key Management for Multicast: Issues and Architectures.” [Online]. Available: http://tools.ietf.org/html/draft-wallner-key-arch-01. [Accessed: 20-Mar-2014].
[23] M. Steiner, G. Tsudik, and M. Waidner, “Key agreement in dynamic peer groups,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 11, no. 8, pp. 769–780, Aug. 2000.
[24] C. K. Wong, M. Gouda, and S. S. Lam, “Secure Group Communications Using Key Graphs,” *IEEEACM Trans Netw*, vol. 8, no. 1, pp. 16–30, Feb. 2000.
[25] M. Li, R. Poovendran, and C. Berenstein, “Design of secure multicast key management schemes with communication budget constraint,” *IEEE Commun. Lett.*, vol. 6, no. 3, pp. 108–110, Mar. 2002.
[26] W. Trappe, J. Song, R. Poovendran, and K. J. R. Liu, “Key management and distribution for secure multimedia multicast,” *IEEE Trans. Multimed.*, vol. 5, no. 4, pp. 544–557, Dec. 2003.
[27] A. T. Sherman and D. A. McGrew, “Key establishment in large dynamic groups using one-way function trees,” *IEEE Trans. Softw. Eng.*, vol. 29, no. 5, pp. 444–458, May 2003.
[28] Z. Fei, M. H. Ammar, I. Kamel, and S. Mukherjee, “An active buffer management technique for providing interactive functions in broadcast video-on-demand systems,” *IEEE Trans. Multimed.*, vol. 7, no. 5, pp. 942–950, Oct. 2005.
[29] G. Calandriello, P. Papadimitratos, J.-P. Hubaux, and A. Lioy, “On the Performance of Secure Vehicular Communication Systems,” *IEEE Trans. Dependable Secure Comput.*, vol. 8, no. 6, pp. 898–912, Nov. 2011.
[30] H. Shi and M. He, “A communication-efficient key agreement protocol in ad hoc networks,” in *2005 International Conference on Wireless Networks, Communications and Mobile Computing*, 2005, vol. 1, pp. 285–291 vol.1.
[31] S.-T. Wu, J.-H. Chiu, and B.-C. Chieu, “ID-based remote authentication with smart cards on open distributed system from elliptic curve cryptography,” in *2005 IEEE International Conference on Electro Information Technology*, 2005, p. 5 pp.–5.
[32] S.-Y. Wang and C.-S. Laih, “Merging: an efficient solution for a time-bound hierarchical key assignment scheme,” *IEEE Trans. Dependable Secure Comput.*, vol. 3, no. 1, pp. 91–100, Jan. 2006.
[33] D. Purandare and R. Guha, “An Alliance Based Peering Scheme for P2P Live Media Streaming,” *IEEE Trans. Multimed.*, vol. 9, no. 8, pp. 1633–1644, Dec. 2007.
[34] D.-H. Je, J.-S. Lee, Y. Park, and S.-W. Seo, “Computation-and-storage-efficient Key Tree Management Protocol for Secure Multicast Communications,” *Comput Commun*, vol. 33, no. 2, pp. 136–148, Feb. 2010.
[35] M. Ramkumar, “The Subset Keys and Identity Tickets (SKIT) Key Distribution Scheme,” *IEEE Trans. Inf. Forensics Secur.*, vol. 5, no. 1, pp. 39–51, Mar. 2010.
[36] J. A. M. Naranjo, R. Lopez, and L. G. Casado, “Applications of the Extended Euclidean Algorithm to Privacy and Secure Communications,” *CMMSE*, no. Proceedings of the 10th International Conference on Computational and Mathematical Methods in Science and Engineering, pp. 702–713, 2010.
[37] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner, “The VersaKey framework: versatile group key management,” *IEEE J. Sel. Areas Commun.*, vol. 17, no. 9, pp. 1614–1631, Sep. 1999.
[38] A. Penrig, D. Song, and J. D. Tygar, “ELK, a new protocol for efficient large-group key distribution,” in *2001 IEEE Symposium on Security and Privacy, 2001. S P 2001. Proceedings*, 2001, pp. 247–262.
[39] M. Onen and R. Molva, “Reliable Group Rekeying with a Customer Perspective,” *IEEE*, vol. 4, no. IEEE Global Telecommunication Conference, pp. 2072–2076, 2004.

[40] J. Goshi and R. E. Ladner, “Algorithms for Dynamic Multicast Key Distribution Trees,” in *Proceedings of the Twenty-second Annual Symposium on Principles of Distributed Computing*, New York, NY, USA, 2003, pp. 243–251.

[41] J.-H. Cho, I.-R. Chen, and M. Eltoweissy, “On Optimal Batch Rekeying for Secure Group Communications in Wireless Networks,” *Wirel Netw*, vol. 14, no. 6, pp. 915–927, Dec. 2008.

[42] J. S. Lee, J. H. Son, Y.-H. Park, and S.-W. Seo, “Optimal level-homogeneous tree structure for logical key hierarchy,” in *3rd International Conference on Communication Systems Software and Middleware and Workshops, 2008. COMSWARE 2008*, 2008, pp. 677–681.

[43] A. Shoufan and S. A. Huss, “High-Performance Rekeying Processor Architecture for Group Key Management,” *IEEE Trans. Comput.*, vol. 58, no. 10, pp. 1421–1434, Oct. 2009.

[44] X. Hai-tao, Y. Zong-Kai, W. Yu-ming, and C. Wen-Qing, “A M-dimensional Sphere Multicast Rekeying Scheme,” in *WRI International Conference on Communications and Mobile Computing, 2009. CMC ’09*, 2009, vol. 3, pp. 418–422.

[45] B. Bruhadeshwar, S. S. Kulkarni, and A. X. Liu, “Symmetric Key Approaches to Securing BGP #x02014;A Little Bit Trust Is Enough,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 9, pp. 1536–1549, Sep. 2011.

[46] B. Bruhadeshwar and S. S. Kulkarni, “Balancing Revocation and Storage Trade-Offs in Secure Group Communication,” *IEEE Trans. Dependable Secure Comput.*, vol. 8, no. 1, pp. 58–73, Jan. 2011.

[47] C.-L. Hu and M.-S. Chen, “Adaptive information dissemination: an extended wireless data broadcasting scheme with loan-based feedback control,” *IEEE Trans. Mob. Comput.*, vol. 2, no. 4, pp. 322–336, Oct. 2003.

[48] D. Westhoff, J. Girao, and M. Acharya, “Concealed Data Aggregation for Reverse Multicast Traffic in Sensor Networks: Encryption, Key Distribution, and Routing Adaptation,” *IEEE Trans. Mob. Comput.*, vol. 5, no. 10, pp. 1417–1431, Oct. 2006.

[49] K. Sakiyama, L. Batina, B. Preneel, and I. Verbauwhede, “Multicore Curve-Based Cryptoprocessor with Reconfigurable Modular Arithmetic Logic Units over GF(2^n),” *IEEE Trans. Comput.*, vol. 56, no. 9, pp. 1269–1282, Sep. 2007.

[50] N. Sultana, K. M. Choi, and E.-N. Huh, “Application Driven Cluster Based Group Key Management with Identifier in Mobile Wireless Sensor Network,” in *Future Generation Communication and Networking (FGCN 2007)*, 2007, vol. 1, pp. 362–367.

[51] H. Chen and Y. Xiao, “On-Bound Selection Cache Replacement Policy for Wireless Data Access,” *IEEE Trans. Comput.*, vol. 56, no. 12, pp. 1597–1611, Dec. 2007.

[52] Y. Zhou and Y. Fang, “A Two-Layer Key Establishment Scheme for Wireless Sensor Networks,” *IEEE Trans. Mob. Comput.*, vol. 6, no. 9, pp. 1009–1020, Sep. 2007.

[53] Q. Xu, M. He, and L. Harn, “An Improved Time-Bound Hierarchical Key Assignment Scheme,” in *IEEE Asia-Pacific Services Computing Conference, 2008. APSCC ’08*, 2008, pp. 1489–1494.

[54] H.-Y. Chen, “Efficient time-bound hierarchical key assignment scheme,” *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 10, pp. 1301–1304, Oct. 2004.

[55] X. Yi and Y. Ye, “Security of Tzeng’s time-bound key assignment scheme for access control in a hierarchy,” *IEEE Trans. Knowl. Data Eng.*, vol. 15, no. 4, pp. 1054–1055, Jul. 2003.

[56] Y. Lin and S. Shieh, “Lightweight, Pollution-Attack Resistant Multicast Authentication Scheme.” ACM, 2006.

[57] P. Tague, D. Slater, J. Rogers, and R. Poovendran, “Evaluating the Vulnerability of Network Traffic Using Joint Security and Routing Analysis,” *IEEE Trans. Dependable Secure Comput.*, vol. 6, no. 2, pp. 111–123, Apr. 2009.

[58] P. Tague, M. Li, and R. Poovendran, “Mitigation of Control Channel Jamming under Node Capture Attacks,” *IEEE Trans. Mob. Comput.*, vol. 8, no. 9, pp. 1221–1234, Sep. 2009.

[59] P. P. Tsang, A. Kapadia, C. Cornelius, and S. W. Smith, “Nymble: Blocking Misbehaving Users in Anonymizing Networks,” *IEEE Trans. Dependable Secure Comput.*, vol. 8, no. 2, pp. 256–269, Mar. 2011.
[60] Y. Sun, T. F. La Porta, and P. Kermani, “A Flexible Privacy-Enhanced Location-Based Services System Framework and Practice,” *IEEE Trans. Mob. Comput.*, vol. 8, no. 3, pp. 304–321, Mar. 2009.
[61] B. Zheng, W.-C. Lee, P. Liu, D. L. Lee, and X. Ding, “Tuning On-Air Signatures for Balancing Performance and Confidentiality,” *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 12, pp. 1783–1797, Dec. 2009.
[62] Y.-H. Lin, A. Studer, Y.-H. Chen, H.-C. Hsiao, L.-H. Kuo, J. Lee, J. M. McCune, K.-H. Wang, M. Krohn, P.-L. Lin, A. Perrig, H.-M. Sun, and B.-Y. Yang, “SPATE: Small-Group PKI-Less Authenticated Trust Establishment,” *IEEE Trans. Mob. Comput.*, vol. 9, no. 12, pp. 1666–1681, 2010.
[63] Y.-H. Lin, B.-Y. Yang, A. Studer, H.-C. Hsiao, J. M. McCune, K.-H. Wang, M. Krohn, P.-L. Lin, A. Perrig, and H.-M. Sun, “SPATE: small-group PKI-less authenticated trust establishment,” presented at the 7th international conference on Mobile systems, applications, and services (MobiSys ’09), New York, NY, USA, 2009, pp. 1–14.
[64] Y. Xuan, Y. Shen, N. P. Nguyen, and M. T. Thai, “A Trigger Identification Service for Defending Reactive Jammers in WSN,” *IEEE Trans. Mob. Comput.*, vol. 11, no. 5, pp. 793–806, May 2012.
[65] M. M. M. Fouad, M. M. Mostafa, and A. R. Dawood, “A Pairwise Key Pre-distribution Scheme Based on Prior Deployment Knowledge,” in *2011 Third International Conference on Computational Intelligence, Communication Systems and Networks (CICSyN)*, 2011, pp. 184–189.
[66] D. Liu, P. Ning, and R. Li, “Establishing Pairwise Keys in Distributed Sensor Networks,” *ACM Trans Inf Syst Secur*, vol. 8, no. 1, pp. 41–77, Feb. 2005.
[67] J. Hur and D. K. Noh, “Attribute-Based Access Control with Efficient Revocation in Data Outsourcing Systems,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 7, pp. 1214–1221, Jul. 2011.
[68] P. Sarkar and A. Saha, “Secure Communication Using Reed-Muller Codes and Partially Balanced Design in Wireless Sensor Network,” in *2011 Ninth IEEE International Symposium on Parallel and Distributed Processing with Applications Workshops (ISPAW)*, 2011, pp. 210–215.
[69] Y. Jen-Ho and C. Chin-Chen, “An ID-based Remote Mutual Authentication with Key Agreement Scheme for Mobile Devices on Elliptic Curve Cryptosystem,” *Elsevier*, vol. 28, pp. 138–143, 2008.
[70] T. Liu and H. Zhu, “An ID-based Multi-server Authentication with Key Agreement Scheme without Verification Table on Elliptic Curve Cryptosystem,” in *2010 International Conference on Computational Aspects of Social Networks (CASoN)*, 2010, pp. 61–64.
[71] M. Shao, S. Zhu, W. Zhang, G. Cao, and Y. Yang, “pDCS: Security and Privacy Support for Data-Centric Sensor Networks,” *IEEE Trans. Mob. Comput.*, vol. 8, no. 8, pp. 1023–1038, Aug. 2009.
[72] C.-I. Fan, L.-Y. Huang, and P.-H. Ho, “Anonymous Multireceiver Identity-Based Encryption,” *IEEE Trans Comput*, vol. 59, no. 9, pp. 1239–1249, Sep. 2010.
[73] J. Wang, Y. Zhao, S. Jiang, and J. Le, “Providing privacy preserving in Cloud computing,” in *2010 3rd Conference on Human System Interactions (HSI)*, 2010, pp. 472–475.
[74] R. Mishra, S. K. Dash, D. P. Mishra, and A. Tripathy, “A privacy preserving repository for securing data across the cloud,” in *2011 3rd International Conference on Electronics Computer Technology (ICECT)*, 2011, vol. 5, pp. 6–10.
[75] U. Greveler, B. Justus, and D. Loehr, “A Privacy Preserving System for Cloud Computing,” in *2011 IEEE 11th International Conference on Computer and Information Technology (CIT)*, 2011, pp. 648–653.

[76] B. Liu, Y. Jiang, F. Sha, and R. Govindan, "Cloud-enabled Privacy-preserving Collaborative Learning for Mobile Sensing," in *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, New York, NY, USA, 2012, pp. 57–70.
[77] T. Jung, X. Li, Z. Wan, and M. Wan, "Privacy preserving cloud data access with multi-authorities," in *2013 Proceedings IEEE INFOCOM*, 2013, pp. 2625–2633.
[78] J. Li and N. Li, "OACerts: Oblivious Attribute Certificates," *IEEE Trans. Dependable Secure Comput.*, vol. 3, no. 4, pp. 340–352, 2006.
[79] M. Nabeel, N. Shang, and E. Bertino, "Privacy Preserving Policy-Based Content Sharing in Public Clouds," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 11, pp. 2602–2614, 2013.
[80] P. Vijayakumar, S. Bose, and A. Kannan, "Centralized key distribution protocol using the greatest common divisor method," *Comput. Math. Appl.*, vol. 65, no. 9, pp. 1360–1368, May 2013.
[81] H. Harney and E. Harder, "Logical Key Hierarchy Protocol." [Online]. Available: http://tools.ietf.org/html/draft-harney-sparta-lkhp-sec-00. [Accessed: 19-May-2014].
[82] P. Vijayakumar, S. Bose, A. Kannan, and D. Jegatha, "Computation and Communication Efficient Key Distribution Protocol for Secure Multicast Communication," *KSII Trans. Internet Inf. Syst.*, vol. 7, no. 4, pp. 878–894, 2013.

TRITA-ICT-EX-2014:107