CERIAS Tech Report 2012-12
Privacy Preserving Access Control on Third-Party Data Management Systems
by Mohamed Nabeel
Center for Education and Research
Information Assurance and Security
Purdue University, West Lafayette, IN 47907-2086

Graduate School ETD Form 9
(Revised 12/07)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Mohamed Yoosuf Mohamed Nabeel

Entitled
Privacy Preserving Access Control on Third-Party Data Management Systems

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

Elisa Bertino, Ph.D.
Chair

Ninghui Li, Ph.D.

Samuel S. Wagstaff, Ph.D.

Dongyan Xu, Ph.D.

To the best of my knowledge and as understood by the student in the *Research Integrity and Copyright Disclaimer (Graduate School Form 20)*, this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): Elisa Bertino, Ph.D.

Approved by: William J. Gorman, Ph.D. 07/18/2012
Head of the Graduate Program Date

Graduate School Form 20
(Revised 9/10)

PURDUE UNIVERSITY
GRADUATE SCHOOL

Research Integrity and Copyright Disclaimer

Title of Thesis/Dissertation:

Privacy Preserving Access Control on Third-Party Data Management Systems

For the degree of Doctor of Philosophy

I certify that in the preparation of this thesis, I have observed the provisions of *Purdue University Executive Memorandum No. C-22,* September 6, 1991, *Policy on Integrity in Research.**

Further, I certify that this work is free of plagiarism and all materials appearing in this thesis/dissertation have been properly quoted and attributed.

I certify that all copyrighted material incorporated into this thesis/dissertation is in compliance with the United States' copyright law and that I have received written permission from the copyright owners for my use of their work, which is beyond the scope of the law. I agree to indemnify and save harmless Purdue University from any and all claims that may be asserted or that may arise from any copyright violation.

Mohamed Yoosuf Mohamed Nabeel

Printed Name and Signature of Candidate

07/12/2012

Date (month/day/year)

*Located at http://www.purdue.edu/policies/pages/teach_res_outreach/c_22.html

PRIVACY PRESERVING ACCESS CONTROL FOR THIRD-PARTY DATA MANAGEMENT SYSTEMS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Mohamed Yoosuf Mohamed Nabeel

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2012

Purdue University

West Lafayette, Indiana


بسم الله الرحمن الرحيم

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to my adviser, Prof. Elisa Bertino, for her unwavering support, patience and guidance through out my PhD program. Without her constant support, advice and encouragement, this dissertation could not have been completed.

I would like to thank Prof. Ninghui Li, Prof. Samuel S. Wagstaff, Jr., Prof. Sunil Prabhakar and Prof. Dongyan Xu for taking time off their busy schedule to be in my committee and providing their invaluable input.

I am also grateful to my mentors and supervisors who I worked with during my summer internships and graduate assistantships: Ann Christine Catlin from Rosen Center for Advanced Computing, Dr. David G. Stork from Ricoh Innovations, and Dr. Mourad Ozzani from Cyber Center.

I am fortunate to be surrounded by an amazing group of fellow graduate students and friends at Purdue. Special thanks to my colleague Ning Shang whom I closely collaborated with during my initial research work. I would like to thank Purdue University for supporting my research through Purdue Research Foundation (PRF) scholarship and the Fulbright fellowship.

Finally and most importantly, words cannot express my gratitude to my parents, Yoosuf and Zeenathunnisa, my wife Muffarriha, my siblings Zahmy, Nasly, Shireen and Jasly for their unconditional love and always supporting me. I am very grateful to the Almighty God for giving me the strength to achieve my dreams.

TABLE OF CONTENTS

Page

LIST OF TABLES vii

LIST OF FIGURES viii

SYMBOLS x

ABBREVIATIONS xi

ABSTRACT xiii

1 INTRODUCTION 1
- 1.1 Privacy Preserving Access Control in Pull Based Systems 2
- 1.2 Privacy Preserving Access Control in Subscription-based Systems . 4
- 1.3 Attribute Based Group Key Management 6
- 1.4 Contributions and Document Structure 7

2 BROADCAST GROUP KEY MANAGEMENT 9
- 2.1 Requirements for a Secure and Effective GKM 10
- 2.2 Broadcast GKM 11
- 2.3 Our Construction: ACV-BGKM 15
- 2.4 Security Analysis 16
- 2.5 Improving the Performance of ACV-BGKM 21
  - 2.5.1 Bucketization 21
  - 2.5.2 Subset Cover 22
- 2.6 ACV-BGKM-2 23
  - 2.6.1 Security Analysis 25
- 2.7 Experimental Results 27

3 ATTRIBUTE BASED GROUP KEY MANAGEMENT 31
- 3.1 Scheme 1: Inline AB-GKM 32
  - 3.1.1 Our Construction 33
  - 3.1.2 Security 36
  - 3.1.3 Performance 39
- 3.2 Scheme 2: Threshold AB-GKM 39
  - 3.2.1 Our Construction 41
  - 3.2.2 Security 43
  - 3.2.3 Performance 44
- 3.3 Scheme 3: Access Tree AB-GKM 45
  - 3.3.1 Access Tree 45

Page

3.3.2 Our Construction . . . 46
3.3.3 Security . . . 49
3.3.4 Performance . . . 50
3.4 Example Application . . . 51
3.5 Experimental Results . . . 55

4 PRIVACY PRESERVING PULL BASED SYSTEMS: SINGLE LAYER APPROACH . . . 59
4.1 Overview of the SLE Approach . . . 60
4.2 Preserving the Privacy of Identity Attributes . . . 62
4.2.1 Discrete Logarithm Problem and Computational Diffie-Hellman Problem . . . 63
4.2.2 Pedersen Commitment . . . 63
4.2.3 OCBE Protocols . . . 64
4.2.4 Configurable Privacy . . . 67
4.3 Single Layer Encryption Approach . . . 68
4.3.1 Identity Token Issuance . . . 69
4.3.2 Identity Token Registration . . . 70
4.3.3 Data Management . . . 74
4.4 Improving Efficiency of Re-Encryption . . . 76
4.5 An Example Application . . . 80
4.6 Experimental Results . . . 85
4.6.1 Privacy Preserving Secret Delivery . . . 85
4.6.2 Data and Key Management . . . 87
4.6.3 Encryption Management . . . 91

5 PRIVACY PRESERVING PULL BASED SYSTEMS: TWO LAYER ENCRYPTION APPROACH . . . 93
5.1 Overview . . . 95
5.2 Policy Decomposition . . . 97
5.2.1 Policy Cover . . . 98
5.2.2 Policy Decomposition . . . 105
5.3 Two Layer Encryption Approach . . . 107
5.3.1 Identity Token Issuance . . . 107
5.3.2 Policy Decomposition . . . 108
5.3.3 Identity Token Registration . . . 108
5.3.4 Data Encryption and Upload . . . 108
5.3.5 Data Downloading and Decryption . . . 109
5.3.6 Encryption Evolution Management . . . 109
5.4 Analysis . . . 110
5.4.1 SLE vs. TLE . . . 110
5.4.2 Security and Privacy . . . 111
5.5 Experimental Results . . . 112

Page

6 PRIVACY PRESERVING SUBSCRIPTION BASED SYSTEMS 118
6.1 Overview 121
6.1.1 Interactions 124
6.1.2 Trust Model 126
6.2 Background 127
6.2.1 Pedersen Commitment 127
6.2.2 Zero-Knowledge Proof of Knowledge (Schnorr's Scheme) . . 128
6.2.3 Euler's Totient Function $\phi(\cdot)$ and Euler's Theorem 128
6.2.4 Composite Square Root Problem 128
6.2.5 Paillier Homomorphic Cryptosystem 129
6.3 Proposed Scheme 130
6.3.1 Initialize 131
6.3.2 Register 132
6.3.3 Subscribe 133
6.3.4 Publish 134
6.3.5 Match 135
6.3.6 Cover 137
6.3.7 The Distribution of Load 138
6.4 Experimental Results 138
6.4.1 Protocol Experiments 139
6.4.2 System Experiments 143

7 Survey of Related Work 147
7.1 Group Key Management (GKM) 147
7.2 Functional Encryption 148
7.3 Selective Publishing of Documents 149
7.4 Secure Data Outsourcing 150
7.5 Secret Sharing Schemes 151
7.6 Proxy Re-Encryption Systems 151
7.7 Searchable Encryption 152
7.8 Secure Multiparty Computation (SMC) 152
7.9 Private Information Retrieval (PIR) 153

8 SUMMARY 154

LIST OF REFERENCES 157

VITA 163

LIST OF TABLES

Table		Page
3.1	Access tree functions	46
3.2	Insurance plans supported by doctors/nurses	52
3.3	User attribute matrix	52
3.4	List of employees satisfying each insurance plan	53
3.5	List of employees satisfying attributes	53
3.6	Average time for CP-ABE algorithms	56
4.1	A table of secrets maintained by the Pub	73
4.2	Average computation time for running one round of the EQ-OCBE protocol	86
6.1	Matching decision	136
6.2	Average computation time for general operations	139

LIST OF FIGURES

Figure		Page
1.1	A typical pull based system	3
1.2	A typical publish-subscribe system	5
2.1	Average time to generate keys	28
2.2	Average time to derive keys	29
2.3	Average time to generate keys with different bucket sizes	29
2.4	Average time to derive keys with different bucket sizes	30
2.5	Average time to generate keys with the two optimizations	30
2.6	Average time to derive keys with the two optimizations	30
3.1	Average key generation time for different group sizes	56
3.2	Average encryption/decryption time for different group sizes	57
3.3	Average key generation time for varying attribute counts	58
4.1	Overall system architecture	61
4.2	Average computation time for running one round of GE-OCBE protocol	87
4.3	Time to generate an ACV for different user configurations	88
4.4	Key derivation time for different user configurations	89
4.5	Size of ACV for different user configurations	89
4.6	ACV generation and key derivation for different number of conditions per policy	90
4.7	Different incremental encryption modes	91
4.8	Average time to perform insert operation	91
5.1	Two layer encryption approach	96
5.2	The example graph	104
5.3	Size of ACCs for 100 attributes	113
5.4	Size of ACCs for 500 attributes	113

Figure Page

5.5 Size of ACCs for 1000 attributes 114
5.6 Size of ACCs for 1500 attributes 114
5.7 Policy decomposition time breakdown with the random cover algorithm 115
5.8 Policy decomposition time breakdown with the greedy cover algorithm 116
5.9 Average time to generate keys for the two approaches 116
5.10 Average time to derive keys for the two approaches 117
6.1 An example CBPS system 119
6.2 Sub registering with Pub 132
6.3 Sub authenticating itself to Broker 133
6.4 Time to blind subscriptions/notifications for different bit lengths of n . 141
6.5 Time to blind subscriptions/notifications for different l 142
6.6 Time to perform match/cover for different bit lengths of n 142
6.7 Time to perform match/cover for different l 143
6.8 Equality filtering time 144
6.9 Equality filtering time for different domain sizes 145
6.10 Inequality filtering time for different domain sizes 146

SYMBOLS

$\mathcal{KS}$	Keyspace
ACP	Policy
$\mathcal{A}$	Attribute universe
$\mathcal{SS}$	Secret space
$\mathbf{S}$	The set of issued secrets
$\mathbf{AS}$	The set of aggregated secrets
$\mathcal{T}$	Access tree

ABBREVIATIONS

ABAC	Attribute Based Access Control
ABE	Attribute Based Encryption
AB-GKM	Attribute Based Group Key Management
ACC	Attribute Condition Cover
ACP	Access Control Policy
ACV	Access Control Vector
AVP	Attribute Value Pair
BGKM	Broadcast Group Key Management
CBPS	Content Based Publish Subscribe
CP-ABE	Ciphertext Policy Attribute Based Encryption
DaaS	Data as a Service
EHR	Electronic Health Record
GKM	Group Key Management
KEV	Key Extraction Vector
KP-ABE	Key Policy Attribute Based Encryption
OCBE	Oblivious Commitment Based Envelope
PaaS	Platform as a Service
PI	Public Information tuple
PIR	Private Information Retrieval
RBAC	Role Based Access Control
SaaS	Software as a Service
SLE	Single Layer Encryption
SMC	Secure Multiparty Computation
TLE	Two Layer Encryption

TTP	Trusted Third Party
UA	User-Attribute matrix
ZKPK	Zero Knowledge Proof of Knowledge

ABSTRACT

Mohamed Nabeel, Mohamed Yoosuf Ph.D., Purdue University, August 2012. Privacy Preserving Access Control for Third-Party Data Management Systems. Major Professor: Elisa Bertino.

The tremendous growth in electronic media has made publication of information in either open or closed environments easy and effective. However, most application domains (e.g. electronic health records (EHRs)) require that the fine-grained selective access to information be enforced in order to comply with legal requirements, organizational policies, subscription conditions, and so forth. The problem becomes challenging with the increasing adoption of cloud computing technologies where sensitive data reside outside of organizational boundaries. An important issue in utilizing third party data management systems is how to selectively share data based on fine-grained attribute based access control policies and/or expressive subscription queries while assuring the confidentiality of the data and the privacy of users from the third party.

In this thesis, we address the above issue under two of the most popular dissemination models: pull based service model and subscription based publish-subscribe model. Encryption is a commonly adopted approach to assure confidentiality of data in such systems. However, the challenge is to support fine grained policies and/or expressive content filtering using encryption while preserving the privacy of users. We propose several novel techniques, including an efficient and expressive group key management scheme, to overcome this challenge and construct privacy preserving dissemination systems.

1 INTRODUCTION

In the cloud computing era, disseminating and sharing data through a third-party service provider has never been more economical and easier than now. However, such service providers cannot be trusted to assure the confidentiality of the data. In fact, data privacy and security issues have been major concerns for many organizations utilizing such services. Data (e.g. electronic health records (EHRs)) often encode sensitive information and should be protected in order to comply with various organizational policies, legal regulations, subscription conditions, and so forth. Encryption is a commonly adopted approach to protect the confidentiality of the data. Encryption alone however is not sufficient as organizations often have to enforce fine-grained access control on the data. Such control is often based on the attributes of users, referred to as *identity attributes*, such as the roles of users in the organization, projects on which users are working and so forth, as well as the attributes of data, referred to as *content attributes*. These systems, in general, are called *attribute based systems*. Therefore, an important requirement is to support fine-grained access control, based on policies and subscription conditions specified using identity and content attributes, over encrypted data.

With the involvement of the third-party services, a crucial issue is that the identity attributes in the access control policies (ACPs) often reveal privacy-sensitive information about users and leak confidential information about the data. The confidentiality of the data and the privacy of the users are thus not fully protected if the identity attributes are not protected. Further, privacy, both individual as well as organizational, is considered a key requirement in all solutions, including cloud services, for digital identity management [1–4]. Further, as insider threats [5] are one of the major sources of data theft and privacy breaches, identity attributes must be strongly protected even from accesses within organizations. With initiatives such

as cloud computing the scope of insider threats is no longer limited to the organizational perimeter. Therefore, protecting the identity attributes of the users while enforcing attribute-based access control both within the organization as well as in the third-party service is crucial.

In this thesis, we investigate the problem of providing privacy preserving access control on third-party systems under two of the most popular dissemination models: pull based service model and subscription based publish-subscribe model. In a pull based system, the data owner (Owner) uploads its data to a third-party server which acts as a data repository. Users having valid credentials are allowed to download data from the server. In a subscription based system, authorized users submit subscription queries, which specify their interests, to the third-party server, which acts as a brokering network. The Owner publishes data to the third-party server which in turn forwards the data to many matching users based on their subscriptions. For both models, we propose approaches to assure confidentiality of the data and privacy of users from the third party server. The challenge is to support fine grained policies and/or expressive data filtering using encryption while preserving the privacy of users. Group key management (GKM) is a fundamental building block used to address this challenge. We identify that the existing GKM schemes are not well designed to manage keys based on the attributes of users and to protect the privacy. As part of this thesis, we first address this issue by constructing a novel scheme called attribute based GKM (AB-GKM).

1.1 Privacy Preserving Access Control in Pull Based Systems

Figure 1.1 shows the architecture of a typical pull based system. Users initially registers with the Owner and obtains the keys for the data they are authorized to access. The Owner selectively encrypts the data and uploads to the third party server such as Amazon S3 or Rackspace Cloud Files. Users download encrypted data from

the third party and decrypt using the keys obtained from the Owner at the time of registration.

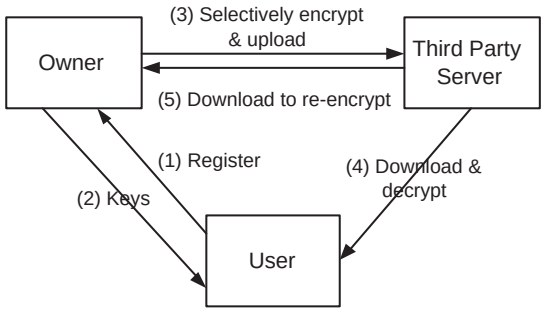

Figure 1.1.: A typical pull based system

We identify the following requirements to assure privacy of users and confidentiality of data from the third-party while at the same time assuring that the third-party enforces the ACPs specified by the data owner.

- The identity attributes of users must not be revealed to the third-party.
- The ACPs of the Owner must not be revealed to the third-party.
- The third-party must not learn the sensitive information in the data.
- Users must be granted access to portions of data only if their identity attributes satisfy the corresponding ACPs.

As shown in Figure 1.1, the most common approach to support fine-grained selective attribute-based access control before uploading the data to the third-party server is to encrypt each data item to which the same ACP (or set of ACPs) applies with the same key. One approach to deliver the correct keys to the users based on the policies they satisfy is to use a hybrid solution where the keys are encrypted using a public key cryptosystem such as attribute based encryption (ABE) and/or proxy re-encryption

(PRE). However, such an approach has several weaknesses: it cannot efficiently handle adding/revoking users or identity attributes, and policy changes; it requires to keep multiple encrypted copies of the same key; it incurs high computational cost. Therefore, a different approach is required.

It is worth noting that a simplistic group key management (GKM) scheme in which the Owner directly delivers the symmetric keys to corresponding users has some major drawbacks with respect to user privacy and key management. On one hand, user private information encoded in the user identity attributes is not protected in the simplistic approach. On the other hand, such a simplistic key management scheme does not scale well as the number of users becomes large and when multiple keys need to be distributed to multiple users. A key contribution of this thesis is to develop a key management scheme which does not have the above shortcomings. We observe that, without utilizing public key cryptography and by allowing users to dynamically derive the symmetric keys at the time of decryption, one can address the above weaknesses. Based on this idea, we first formalize a new GKM scheme called broadcast GKM (BGKM) and then give a secure construction of BGKM scheme and formally prove its security.

1.2 Privacy Preserving Access Control in Subscription-based Systems

Figure 1.2 shows the architecture of a content based publish subscribe (CBPS) system. The Owner plays the role of content publishers (Pubs) and users play the role of subscribers (Subs). The third-party brokering network manages subscriptions from users and distribute the data published by the Owner, called notifications, to users based on their subscriptions.

We identify the following requirements to assure privacy of users and confidentiality of data published by the Owner form the third-party brokering network while at the same time assuring that only authorized users can access the data.

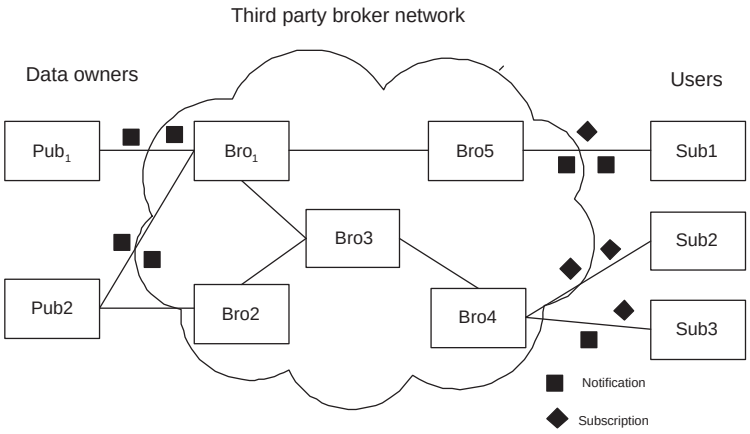

Figure 1.2.: A typical publish-subscribe system

- Publication confidentiality: The content of notifications must be hidden from the third party brokers.
- Subscription privacy: The content of the subscriptions must be hidden from the third party brokers.
- The third party brokers must make forwarding decisions on hidden notifications and subscriptions without learning the actual differences of notification and subscription values. In other words, a randomized comparison scheme must be provided.

Privacy and confidentiality issues in CBPS systems have long been identified [6], but little progress has been made to address these issues in a holistic manner. Most of prior work on data confidentiality techniques in the context of CBPS systems is based on the assumption that content brokers are trusted with respect to the privacy of the subscriptions by users [7–9]. With the absence of such an assumption the problem becomes challenging as brokers need to make decisions without knowing the actual notifications and subscriptions. In this thesis, we address this challenge by proposing a novel scheme which is inspired from the Paillier homomorphic cryptosystem [10], and

uses AB-GKM scheme and zero-knowledge proof of knowledge (ZKPK) protocols [11]. It should be noted that existing approaches that try to achieve similar goals as ours have limitations which undermine flexibility and/or accuracy [12–14].

1.3 Attribute Based Group Key Management

Group key management (GKM) plays a key role in building privacy preserving data dissemination systems under both pull based models as well as publish-subscribe models. Attribute based systems enable fine-grained access control among a group of users each identified by a set of attributes. Privacy preserving data dissemination systems need such flexible attribute based systems for managing and distributing group keys. However, current GKM schemes are not well designed to manage group keys based on the identity attributes of users.

In this thesis, we construct a new key management scheme called broadcast GKM (BGKM) that allows users whose attributes satisfy a certain policy to derive group keys. The idea is to give secrets to users based on the identity attributes they have and later allow them to derive actual symmetric keys based on their secrets and some public information. A key advantage of the BGKM scheme is that adding users/revoking users or updating ACPs can be performed efficiently and only requires updating the public information. Our BGKM scheme satisfies the requirements of *minimal trust*, *key indistinguishability*, *key independence*, *forward secrecy*, *backward secrecy* and *collusion resistance* as described in [15] with minimal computational, space and communication cost.

Using the BGKM scheme as a building block, we construct a more expressive GKM scheme called attribute based GKM (AB-GKM) which allows one to express any threshold or monotonic [1] conditions over a set of identity attributes as the group membership condition. It should be noted that the AB-GKM scheme recalls the notion of attribute-based encryption (ABE) [16–18]; however, as we discuss later in

[1] Monotone formulas are Boolean formulas that contain only conjunction and disjunction connectives, but no negation.

Chapter 3, ABE has several shortcomings when applied to GKM. In the pull based model, we use the AB-GKM scheme to manage the keys used to selectively encrypt data based on fine-grained policies. In the publish-subscribe model, we use AB-GKM to manage the keys to encrypt payload messages.

1.4 Contributions and Document Structure

This thesis studies how we can build privacy preserving access control on third party data management systems. Specifically, we propose privacy preserving access control for two of the most popular dissemination models: pull based service model and subscription based publish-subscribe model.

Chapter 2 proposes a new GKM scheme called broadcast GKM (BGKM) and provides detailed security proofs to show that the scheme is secure. Using the BGKM construct as a building block, in Chapter 3, we propose a more expressive scheme called attribute based GKM (AB-GKM) which can handle any monotonic policies over attribute conditions. We provide experimental results to show that our constructs are efficient and practical.

Chapter 4 proposes a novel approach to privacy preserving pull based system called *Single Layer Encryption* (SLE). To the best of our knowledge, it is the first approach to assure the confidentiality of the data from the third party server and preserve the privacy of users while enforcing attribute based ACPs on data. In the SLE approach, the Owner itself enforces all ACPs by selectively encrypting the data before uploading to the third party. While the SLE approach provides many benefits over existing solutions, the Owner has to incur high communication and computation cost to manage keys and encryptions whenever user credentials or organizational authorization policies change. A better approach should delegate the enforcement of fine-grained access control to the third party, so to minimize the overhead at the Owner, whereas at the same time assuring data confidentiality from the third-party server. In Chapter 5, we propose an extension to SLE approach called the *Two Layer*

Encryption (TLE) in order to address such requirement. Under the TLE approach, the Owner performs a coarse grained encryption and the third party performs a fine grained encryption. Since as much access control enforcement as possible is delegated to the third party, the TLE approach reduces the workload at the Owner. In both approaches, AB-GKM scheme is used to manage group keys and support attribute based ACPs through selective encryption. We provide experimental results for both approaches and compare their performance.

Chapter 6 proposes a novel privacy preserving subscription based system. Compared to pull based systems, additional mechanisms are required to preserve the privacy in subscription based systems as the third party needs to make decisions based on data in addition to the credentials of users. Our approach preserves the privacy of the subscriptions made by users and confidentiality of the data published by the Owner using a tweaked version of the Paillier homomorphic cryptosystem [10] when third-party content brokers are utilized to make routing decisions based on the content. The AB-BGKM scheme is used to manage the keys used to encrypt the payload of the data published. Our protocols are expressive to support any type of subscriptions and designed to work efficiently. We distribute the work such that the load on the third party content brokers, where the bottleneck is in a CBPS system, is minimized. We extend SIENA [19], a popular CBPS system using our protocols to implement a privacy preserving CBPS system.

Chapter 7 surveys the work related privacy preserving data dissemination systems as well as the cryptographic techniques we propose as part of this thesis.

Chapter 8 provides a summary of this thesis and discuss extensions and future work.

2 BROADCAST GROUP KEY MANAGEMENT

Group key management (GKM) plays a key role in building privacy preserving data dissemination systems under both pull based models as well as publish-subscribe models. Attribute based systems enable fine-grained access control among a group of users each identified by a set of attributes. Privacy preserving data dissemination systems need such flexible attribute based systems for managing and distributing group keys. However, current group key management schemes are not well designed to manage group keys based on the identity attributes of users.

A challenging well known problem in GKM is how to efficiently handle group dynamics, i.e., a new user joining or an existing group member leaving. When the group changes, a new group key must be shared with the existing members, so that a new group member cannot access the data transmitted before she joined (forward secrecy) and a user who left the group cannot access the data transmitted after she left (backward secrecy). The process of issuing a new key is called *rekeying* or *update*. Another challenging problem is to defend against collusion attacks by which a set of colluding fraudulent users are able to obtain group keys which they are not allowed to obtain individually.

In a traditional GKM scheme, when the group changes, the private information given to all or some existing group members must be changed which requires establishing private communication channels. Establishing such channels is a major shortcoming especially for highly dynamic groups. We observe that, without utilizing public key cryptography and by allowing users to dynamically derive the symmetric keys at the time of decryption, one can address this weaknesses. Based on this idea, in this chapter, we first propose a new GKM scheme called broadcast GKM (BGKM) scheme [20,21] that addresses this weakness. The scheme allows one to per-

form rekeying operations by only updating some public information without affecting private information existing group members possess.

In this section, we first list the requirements for an effective GKM, then give an overview of BGKM schemes and finally present our construction along with security proofs.

2.1 Requirements for a Secure and Effective GKM

Several requirements are identified and discussed by Challel and Seba [15] and others for effective GKM. Generally speaking, an efficient and practical GKM should address the following requirements.

- **Minimal trust** requires the GKM scheme to place trust on a small number of entities.
- **Key hiding** requires that with given public information, it is hard for anyone outside the group to gain the shared group key. Ideally, every element in the keyspace should have the same probability of being the real key.
- **Key independence** requires that the leak of one key does not compromise other keys.
- **Backward secrecy** means that a member who has left the group cannot access any future group keys.
- **Forward secrecy** means that a newly joining group member cannot access any old keys.
- **Collusion resistance** requires that a set of colluding fraudulent users should not obtain keys which they are not allowed to obtain individually.
- **Low bandwidth overhead** requires that the rekeying should not incur a high volume of messages.

- **Computational costs** should be acceptable at both the server and the group member.
- **Storage requirements** for keys and other relevant information should be minimal.
- **Ease of maintenance** requires that a single change of membership in the group does not need many changes to take place for the other group members.
- **Other requirements** include service availability, minimal packet delays, and so on. These factors are sometimes more affected by real-world settings and implementation, and less related to the high-level design of the GKM.

2.2 Broadcast GKM

In order to provide forward and backward secrecy, rekey operations should be performed whenever the users in the group change. Typical GKM schemes require $O(n)$ [22, 23] or at least $O(\log n)$ [24, 25] private communication channels to perform the rekey operation. In comparison, BGKM schemes make rekey a one-off process [26–28]. In such schemes, rekeying is performed with a single broadcast without using private communication channels. It should be noted that even though BGKM schemes have some similarity with secret sharing (SS) schemes, they are constructed for different purposes. "*k out of n*" SS schemes [29, 30] are constructed to split a secret among n users and allow to recover the secret by combining at least k secret shares. On the contrary, BGKM schemes allow each valid user to recover the secret by using only their secret share. Also, colluding users, who individually cannot recover the secret, are not able to recover the secret collectively. Unlike conventional GKM schemes, BGKM schemes do not give users the private keys. Instead users are given a secret which is combined with public information to obtain the actual private keys. Such schemes have the advantage that it requires a private communication only once for the initial secret sharing and the subsequent rekeying operations are performed

using one broadcast message. Further, such schemes can provide forward and backward security by only changing the public information and without affecting secret shares given to existing users. Based on our preliminary work [20], we propose a provably secure BGKM scheme, called ACV-BGKM (Access Control Vector BGKM), and formalize the notion of BGKM. Further we prove the security of ACV-BGKM.

Definition 2.2.1 (BGKM) *In general, a BGKM scheme consists of the following five algorithms:*

- ***Setup***(ℓ)*: It initializes the BGKM scheme using a security parameter* ℓ*. It also initializes the set of used secrets* $\boldsymbol{S}$*, the secret space* $\mathcal{SS}$ *and the key space* $\mathcal{KS}$*. All the parameters are collectively denoted as* Param*.*

- ***SecGen()****: It selects a random bit string* $s \notin \boldsymbol{S}$ *uniformly at random from the secret space* $\mathcal{SS}$*, adds* s *to* $\boldsymbol{S}$ *and outputs* s*.*

- ***KeyGen(S)****: It chooses a group key* K *uniformly at random from the key space* $\mathcal{KS}$ *and outputs the public information* PI *computed from the secrets in* $\boldsymbol{S}$ *and the group key* K*.*

- ***KeyDer(***s***,*** PI***)****: It takes the user's secret* s *and the public information* PI *to output the group key. The derived group key is equal to* K *if and only if* $s \in \boldsymbol{S}$*.*

- ***Update(S)*** *Whenever the set* $\boldsymbol{S}$ *changes, a new group key* K' *is generated. Depending on the construction, it either executes the* ***KeyGen*** *algorithm again or incrementally updates the output of the last* ***KeyGen*** *algorithm.*

Now we provide some basic notions and formally define security.

Negligible functions

We call a function $f : \mathbb{N} \to \mathbb{R}$ *negligible* if for every positive polynomial $p(\cdot)$ there exists an N such that for all $n > N$, we have $f(n) < 1/p(n)$ [31].

Random oracle model

The random oracle model is a paradigm introduced by Bellare and Rogaway [32] for

design and analysis of certain cryptographic protocols. Intuitively, a random oracle is a mathematical function that can be queried by anyone, and maps every query to a uniformly randomly chosen response from its output domain. In practice, random oracles can be used to model cryptographic hash functions in many cryptographic schemes.

A BGKM scheme should allow a valid group member to derive the shared group key, and prohibit anyone outside the group from doing so. Formally speaking, a BGKM scheme should satisfy the following security properties. It must be correct, sound, key hiding, and forward/backward key protecting. Let Svr be the group controller.

Definition 2.2.2 (Correctness) *Let Usr*[1] *be a current group member with a secret. Let K and PubInfo be Svr's output of the KeyGen algorithm. Let K' be Usr's output of the KeyDer algorithm. A BGKM scheme is* correct *if Usr can derive the correct group key K with overwhelming probability, i.e.,*

$$Pr[K = K'] \geq 1 - f(k),$$

where f is a negligible function in k.

Definition 2.2.3 (Soundness) *Let Usr be an individual without a valid secret. A BGKM scheme is* sound *if the probability that Usr can obtain the correct group key K by substituting the secret with a value val that is not one of the valid secrets and then following the key derivation phase KeyDer is negligible.*

We define the following security game to define the key hiding requirement.

Definition 2.2.4 ($KeyHide_{\mathcal{A},\Pi}$) *1. The Svr, as the challenger, runs the KeyGen algorithm of the BGKM scheme Π and gives the parameters Param to the adversary $\mathcal{A}$.*

[1]In what follows we use the term Usr; however in practice the steps are carried out by the client software transparently to the actual end user.

2. *$\mathcal{A}$ selects two random keys $K_0, K_1 \in \mathcal{KS}$ and give to the* Svr.

3. *The* Svr *flips a random coin $b \in \{0, 1\}$ and selects K_b as the group key and runs the* KeyGen *algorithm.*

4. *The* Svr *gives the public information* PubInfo *of the output of the* KeyGen *algorithm to $\mathcal{A}$.*

5. *$\mathcal{A}$ outputs a guess b' of b.*

6. *The output of the game is defined to be 1 if $b' = b$, and 0 otherwise. We write $KeyHide_{\mathcal{A},\Pi} = 1$ if the output is 1 and in this case we say that $\mathcal{A}$ wins the game.*

The advantage of $\mathcal{A}$ in this game is defined as $Pr[KeyHide_{\mathcal{A},\Pi} = 1] - 1/2$.

Definition 2.2.5 (Key hiding) *A BGKM scheme is* key hiding *if given* PubInfo*, any party which does not have a valid secret cannot distinguish the real group key from a randomly chosen value in the keyspace $\mathcal{KS}$ with nonnegligible probability. More specifically, a BGKM scheme, Π, is* key hiding *if for any adversary $\mathcal{A}$ as a probabilistic interactive Turing machine [33], has a negligible advantage in the key hiding security game 2.2.4:*

$$Pr[KeyHide_{\mathcal{A},\Pi} = 1] \leq 1/2 + f(k),$$

where f is a negligible function in k.

Definition 2.2.6 (Forward/backward key protecting) *Suppose* Svr *runs an* Update *algorithm to generate* Param *for a new shared group key K', and a previous member* Usr *is no longer a group member after the* Update *algorithm. Let K be a previous shared group key which can be derived by* Usr *with a secret. A BGKM scheme is* backward key protecting *if an adversary with knowledge of the secret, K, and the new* PubInfo *cannot distinguish the new key K' from a random value in the keyspace $\mathcal{KS}$ with nonnegligible probability. Similarly, a BGKM scheme is* forward key protecting *if a new group member* Usr *after running the* Update *algorithm cannot learn anything about the previous group keys.*

2.3 Our Construction: ACV-BGKM

We now provide our construction of BGKM, the ACV-BGKM scheme, under a client-server architecture. The ACV-BGKM scheme satisfies the requirements of *minimal trust*, *key indistinguishability*, *key independence*, *forward secrecy*, *backward secrecy* and *collusion resistance* as described earlier.

ACV-BGKM algorithms are executed with a trusted key server Svr and a group of users $\mathsf{Usr}_i, i = 1, 2, \ldots, n$.

Setup(ℓ): Svr initializes the following parameters: an ℓ-bit prime number q, a cryptographic hash function $H(\cdot) : \{0,1\}^* \rightarrow \mathbb{F}_q$, where $\mathbb{F}_q$ is a finite field with q elements, the keyspace $\mathcal{KS} = \mathbb{F}_q$, the secret space $\mathcal{SS} = \{0,1\}^\ell$ and the set of issued secrets $\mathbf{S} = \emptyset$.

SecGen(Usr_i): Svr chooses the secret $s_i \in \mathcal{SS}$ uniformly at random for Usr_i such that $s_i \notin \mathbf{S}$ and adds s_i to $\mathbf{S}$.

KeyGen(S): Svr picks a random $K \in \mathcal{KS}$ as the group key. Svr chooses n random bit strings $z_1, z_2, \ldots, z_n \in \{0,1\}^\ell$. Svr creates an $n \times (n+1)$ $\mathbb{F}_q$-matrix

$$A = \begin{pmatrix} 1 & a_{1,1} & a_{1,2} & \ldots & a_{1,n} \\ 1 & a_{2,1} & a_{2,2} & \ldots & a_{2,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & a_{n,1} & a_{n,2} & \ldots & a_{n,n} \end{pmatrix},$$

where

$$a_{i,j} = H(s_i||z_j), 1 \leq i \leq n, 1 \leq j \leq n, s_i \in \mathcal{S}. \tag{2.1}$$

Svr then solves for a nonzero $(n+1)$-dimensional column $\mathbb{F}_q$-vector Y such that $AY = 0$. Note that such a nonzero Y always exists as the nullspace of matrix A is

nontrivial by construction. Here we require that Svr chooses Y from the nullspace of A uniformly at random. Svr constructs an $(n+1)$-dimensional $\mathbb{F}_q$-vector

$$X = K \cdot e_1^T + Y,$$

where $e_1 = (1, 0, \ldots, 0)$ is a standard basis vector of $\mathbb{F}_q^{n+1}$, v^T denotes the transpose of vector v, and k is the chosen group key. The vector X is called an *ACV*, *access control vector*. Svr lets $PI = \langle X, (z_1, z_2, \ldots, z_n)\rangle$, and outputs public PI and private K.

KeyDer(s_i, PI): Using its secret s_i and the public information PI, Usr_i computes $a_{i,j}, 1 \leq j \leq n$, as in formula (2.1) and sets an $(n+1)$-dimensional row $\mathbb{F}_q$-vector $v_i = (1, a_{i,1}, a_{i,2}, \ldots, a_{i,n})$. Usr_i derives the group key as $K' = v_i \cdot X$.

Update(S): It runs the **KeyGen(S)** algorithm and outputs the new public information PI' and the new group key K'.

2.4 Security Analysis

In the security analysis of ACV-BGKM, we will model the cryptographic hash function H as a random oracle. We further assume $q = O(2^k)$ is a sufficiently large prime power. We first present two lemmas with their proofs and then prove the theorems introduced in Section 2.1.

The following lemmas are useful for the security analysis of ACV-BGKM. Lemma 1 says that in a vector space V over a large finite field, the probability that a randomly chosen vector is in a pre-selected subspace, strictly smaller than V, is very small. Lemma 2 will be used in the proof of Theorem 2.6.1.

Lemma 1 *Let $F = \mathbb{F}_q$ be a finite field of q elements. Let V be an n-dimensional F-vector space, and W be an m-dimensional F-subspace of V, where $m \leq n$. Let v be an F-vector uniformly randomly chosen from V. Then the probability that $v \in W$ is $1/q^{n-m}$.*

Proof The proof is straightforward. We show it here for completeness. Let $\{v_1, v_2, \ldots, v_m\}$ be a basis of W. Then it can be extended to a basis of V by adding another $n-m$ basis vector $v_{m+1}, \ldots, v_n$. Any vector $v \in V$ can be written as

$$v = \alpha_1 \cdot v_1 + \ldots + \alpha_n \cdot v_n, \quad \alpha_i \in F, 1 \leq i \leq n,$$

and $v \in W$ if and only if $\alpha_i = 0$ for $m+1 \leq i \leq n$. When v is uniformly randomly chosen from V, if follows

$$\Pr[v \in W] = 1/q^{n-m}.$$

■

Lemma 2 *Let $F = \mathbb{F}_q$ be a finite field of q elements. Let $v_i = (1, v_i^{(2)}, \ldots, v_i^{(n)}), i = 1, \ldots, m$, and $1 \leq m < n$, be n-dimensional F-vectors. Let $v = (1, v^{(2)}, \ldots, v^{(n)})$ be an n-dimensional F-vector with $v^{(j)}, j \geq 2$ independently and uniformly randomly chosen from F. Then the probability that v is linearly dependent of $\{v_i,\ 1 \leq i \leq m\}$ is no more than $1/q^{n-m}$.*

Proof Let $w_i = (v_i^{(2)}, \ldots, v_i^{(n)}), 1 \leq i \leq m$, and $w = (v^{(2)}, \ldots, v^{(n)})$. All w_i span an F-subspace W whose dimension is at most m in an $(n-1)$-dimensional F-vector space. w is a uniformly randomly chosen $(n-1)$-dimensional F-vector. By Lemma 1,

$$\Pr[w \in W] = 1/q^{n-1-\dim(W)} \leq 1/q^{n-1-m}.$$

It follows that

$$\begin{aligned}
&\Pr[v \text{ is linearly dependent of } \{v_i : 1 \leq i \leq m\}] \\
&= \Pr[v = \alpha_1 \cdot v_1 + \ldots + \alpha_m \cdot v_m \text{ for some } \alpha_i \in F] \\
&= \Pr\left[\sum_{i=1}^{m} \alpha_i = 1 \wedge w = \sum_{i=1}^{m} \alpha_i \cdot v_i \text{ for some } \alpha_i \in F\right] \\
&= \Pr\left[\sum_{i=1}^{m} \alpha_i = 1\right] \cdot \Pr[w \in W] \\
&\leq 1/q \cdot 1/q^{n-1-m} = 1/q^{n-m}.
\end{aligned}$$

■

Lemma 3 *Let $F = \mathbb{F}_q$ be a finite field of q elements. Let $v_i = e_i^T + (0, \ldots, 0, v_i^{(n+1)}, \ldots, v_i^{(2n)})$, e_i is the i^{th} standard basis vector of $\mathbb{F}_q^{2n}, i = 1, \ldots, m$, and $1 \leq m \leq n$, be $2n$-dimensional F-vectors. Let $v = e^T + (0, \ldots, 0, v^{(n+1)}, \ldots, v^{(2n)})$ be a $2n$-dimensional F-vector with $v^{(j)}, j \geq n+1$ chosen independently and uniformly at random from F and e from the $2n$-dimensional standard basis vectors with the position of the non-zero element $\leq m$. Then the probability that v is linearly dependent of $\{v_i,\ 1 \leq i \leq m\}$ is no more than $1/q^{n-m}$.*

Proof Let $w_i = (v_i^{(n+1)}, \ldots, v_i^{(2n)}), 1 \leq i \leq m$, $w = (v^{(n+1)}, \ldots, v^{(2n)})$, and $u_i = (v_i^{(1)}, \ldots, v_i^{(n)})$. All w_i span an F-subspace W whose dimension is at most m in an n-dimensional F-vector space. w and u are uniformly randomly chosen n-dimensional F-vectors. By Lemma 1,

$$\Pr[w \in W] = 1/q^{n-\dim(W)} \leq 1/q^{n-m}.$$

It follows that

$$\begin{aligned}
&\Pr[v \text{ is linearly dependent of } \{v_i : 1 \leq i \leq m\}] \\
&= \Pr[v = \alpha_1 \cdot v_1 + \ldots + \alpha_m \cdot v_m \text{ for some } \alpha_i \in F] \\
&= \Pr\ \sum_{i=1}^{m} \alpha_i \cdot u_i = e^T \wedge w = \sum_{i=1}^{m} \alpha_i \cdot v_i \text{ for some } \alpha_i \in F \\
&= \Pr\ \sum_{i=1}^{m} \alpha_i \cdot u_i = e^T \ \cdot \Pr[w \in W] \\
&\leq 1/q^n \cdot 1/q^{n-m} = 1/q^{2n-m}.
\end{aligned}$$

■

Theorem 2.4.1 *ACV-BGKM is correct.*

Proof The correctness of ACV-BGKM can be easily seen: Knowing its secret s_i and the public values $z_1, z_2, \ldots, z_n$, a group member Usr_i can compute one row of matrix A as

$$v_i = (1, a_{i,1}, a_{i,2}, \ldots, a_{i,n}),$$

where $a_{i,j}, 1 \leq j \leq n$ are as in formula (2.1). Therefore $v_i \cdot Y = 0$ for ACV Y, and thus the group key can be derived with probability 1 as

$$v_i \cdot X = v_i \cdot \left(K \cdot e_1^T + Y\right) = K \cdot v_i \cdot e_1^T = K.$$

■

Theorem 2.4.2 *ACV-BGKM is sound.*

Proof Let Y be a given access control vector. Let $\{v_i, 1 \leq i \leq n\}$ be a basis of the nullspace of A. Let $v = (1, v^{(2)}, \ldots, v^{(n+1)})$, where $v^{(i+1)} = H(\mathsf{val}||z_i), 1 \leq i \leq n$. Usr can derive the group key using v by following the KeyDer phase if and only if v is linearly dependent of $v_i, 1 \leq i \leq n$. When val is not a valid IST and H is a random oracle, v is indistinguishable from a vector whose first entry is 1 and the other entries are independently and uniformly chosen from $\mathbb{F}_q$. By Lemma 2, the probability that v is linearly dependent of $\{v_i, 1 \leq i \leq n\}$ is no more than $1/q^{n+1-n} = 1/q$, which is negligible. This proves the soundness of ACV-BGKM. ■

Theorem 2.4.3 *ACV-BGKM is key hiding.*

Proof Let $\mathsf{PubInfo} = \langle X, (z_1, \ldots, z_n)\rangle$ be the public information broadcast from Svr. This is the only piece of information seen by the adversary that is related to the group key. By construction, X must be linearly independent of the standard basis vector e_1^T, i.e., X has a nonzero entry after the first position. For any $K \in \mathcal{KS} = \mathbb{F}_q$, let

$$Y = X - K \cdot e_1^T.$$

Then it is clear that all $\mathbb{F}_q$-vectors v such that $v \cdot Y = 0$ form an n-dimensional $\mathbb{F}_q$-vector space, say W. It follows that the n basis vectors of W can be chosen in such a way that they all have nonvanishing first entries. Therefore, the number of vectors v with 1 as their first entry such that $v \cdot X = K$ is q^{n-1}, for all $K \in \mathcal{KS}$. When the cryptographic hash function $H(\cdot)$ is modeled as a random oracle and a valid IST is unknown, every such a vector v assumes the same probability when

computed as specified in the KeyDer algorithm. This implies that every $K \in \mathcal{KS}$ has the same probability, $1/q$, to be the designated group key in the view of the adversary. The key hiding property of ACV-BGKM follows as a direct consequence. Note that ACV-BGKM is key hiding against a computationally unbounded adversary. ■

It is clear that "forward/backward key protecting" is a stronger condition than "key hiding." However, we will use the proof of the latter to show the former.

Theorem 2.4.4 *ACV-BGKM is forward/backward key protecting.*

Proof (Sketch) We first consider the backward key protecting property of ACV-BGKM. Suppose that after the Update algorithm, an adversary has one secret s from the previous session S_0 which do not propagate to the new session S_1. As the choices of s and the nullspace of the ACV in session S_0 can be viewed as (statistically) jointly independent of the determination of the nullspace of the ACV in session S_1, when H is modeled as a random oracle and by design of the Update algorithm, Usr cannot learn the group key for session S_1 with non-negligible probability due to the key hiding property of ACV-BGKM. Similarly, ACV-BGKM is forward key protecting. ■

Other related GKM security aspects mentioned in Section 2.1 are briefly discussed as follows.

Minimal trust. In order to protect the shared group key from an adversary outside of the group, ACV-BGKM only requires to use a private channel once between Svr and each Usr, during the SecGen algorithm. The security of the ephemeral private channels needs to be guaranteed. Any other communications, including the ones for key issuance and rekeying, are executed via an open broadcast channel.

Key independence. It is clear that the group keys (of different sessions) are independent by ACV-BGKM construction. Furthermore, the secrets are also independent of each other, because they are randomly generated.

Collusion resistance. For BGKM, it only makes sense to consider collusion attacks from outside the group. The case that a valid group member passes its secret or the derived group key to others is not addressed by BGKM. Similar to the analysis for ACV-BGKM's forward/backward key protecting property, ACV-BGKM is resistant to polynomially computationally bounded adversaries. In particular, colluding group members are not able to get the secrets of other members to derive group keys of earlier or later sessions.

2.5 Improving the Performance of ACV-BGKM

In this section, we improve the performance of our basic ACV-BGKM scheme using two techniques: bucketization and subset cover.

2.5.1 Bucketization

The proposed key management scheme works efficiently even when there are thousands of users. However, as the upper bound n of the number of involved users gets large, solving the linear system $AY = 0$ over a large finite field $\mathbb{F}_q$ becomes the most computationally expensive operation in our scheme. Solving this linear system with the method of Gaussian-Jordan elimination [34] takes $O(n^3)$ time. Although this computation is executed at the Svr, which is usually capable of carrying on computationally expensive operations, when n is very large, e.g., $n = 100,000$, the resulting costs may be too high for the Svr. Due to the non-linear cost associated with solving a linear system, we can reduce the overall computational cost by breaking the linear system in to a set of smaller linear systems. We follow a two-level approach. In this case, the Svr divides all the involved Usrs into multiple "buckets" (say m) of a suitable size (e.g., 1000 each), computes an intermediate key for each bucket by executing the KeyGen algorithm, and then computes the actual group key for all the users by executing the KeyGen algorithm with the intermediate keys as the secrets. Note that the intermediate key generation can be parallelized as each bucket is inde-

pendent. The Svr executes $m+1$ KeyGen algorithms of smaller size. The complexity of the KeyGen algorithm is proportional to $O(n^3/m^2 + m^3)$. It can be shown that the optimal solution is achieved when m reaches close to $n^{3/5}$.

Each intermediate key is associated with a marker so that Usrs can identify if they have derived a valid intermediate key. For deriving the actual group key, Usrs are required to execute $m+1$ KeyDer algorithms in the worst case and 2 in the best case. Since the KeyDer algorithm is linear in n, in general, the bucketization optimization still improves the performance of the KeyDer algorithm. The complexity of the KeyGen algorithm is proportional to $O(n/m + m)$, but the average case runs faster.

2.5.2 Subset Cover

The bucketization approach becomes inefficient as the bucket size increases. The issue is that the bucketization still utilizes the basic ACV-BGKM scheme. In our basic ACV-BGKM scheme, as each user is given a single secret, it makes the complexity of PubInfo and all algorithms proportional to n, the number of users in the group. We utilize the result from previous research on broadcast encryption [35, 36] to improve the complexity to sub-linear in n. Based on that, one can make the complexity sub-linear in the number of users by giving more than one secret during SecGen for each attribute users possess. The secrets given to each user overlaps with different subsets of users. During the KeyGen, Svr identifies the minimum number of subsets to which all the users belong and uses one secret per the identified subset. During KeyDer, a user identifies the subset it belongs to and uses the corresponding secret to derive the group key. Group dynamics are handled by making some of the secrets given to users invalid.

We give a high-level description of the basic *subset-cover* approach. In the basic scheme, n users are organized as the leaves of a balanced binary tree of height $\log n$. A unique secret is assigned to each vertex in the tree. Each user is given $\log n$ secrets that correspond to the vertices along the path from its leaf node to the root node.

In order to provide backward secrecy when a single user is revoked, the updated tree is described by $\log n$ subtrees formed after removing all the vertices along the path from the user leaf node to the root node. To rekey, Svr executes **Update** using the $\log n$ secrets corresponding to the roots of these subtrees. Naor et al. [35] improve this technique to simultaneously revoke r users and describe the exiting users using $r \log (n/r)$ subtrees. Since then, there have been many improvements to the basic scheme. We implement Naor et al.'s complete subset scheme [35] in our experiments.

In our experimental results in Section 2.7, we show that combining the bucketization and the subset cover techniques, we can very efficiently execute ACV-BGKM algorithms and can support very large user groups.

2.6 ACV-BGKM-2

The modified ACV-BGKM works under similar conditions as ACV-BGKM, but instead of giving the same key k to all the users, the **KeyDer** algorithm gives each Usr_i a different key k_i when the public information tuple PI is combined with their unique secret s_i.

The algorithms are executed with a trusted key server Svr and a group of users Usr_i, $i = 1, 2, \cdots, n$ with the attribute universe $\mathcal{A} = \{\mathsf{attr}_1, \mathsf{attr}_2, \cdots, \mathsf{attr}_m\}$. The construction is as follows:

Setup(ℓ): Svr initializes the following parameters: an ℓ-bit prime number q, the maximum group size N ($\geq n$), a cryptographic hash function $H(\cdot) : \{0,1\}^* \rightarrow \mathbb{F}_q$, where $\mathbb{F}_q$ is a finite field with q elements, the key space $\mathcal{KS} = \mathbb{F}_q$, the secret space $\mathcal{SS} = \{0,1\}^\ell$ and the set of issued secret tuples $\mathbf{S} = \emptyset$. Each Usr_i is given a unique secret index $1 \leq i \leq N$.

SecGen(): The Svr chooses the secret $s_i \in \mathcal{SS}$ uniformly at random for Usr_i such that s_i is unique among all the users, adds the secret tuple $\langle i, s_i \rangle$ to $\mathbf{S}$, and outputs $\langle i, s_i \rangle$.

KeyGen(S, K): Given the set of secret tuples $\mathbf{S} = \{\langle i, s_i\rangle | 1 \leq i \leq N\}$ and a random set of keys $\mathbf{K} = \{k_i | 1 \leq i \leq N\}$, it outputs the public information tuple PI which allows each Usr_i to derive the key k_i using its secret s_i. The details follow.

Svr chooses N random bit strings $z_1, z_2, \ldots, z_N \in \{0,1\}^\ell$ and creates an $N \times 2N$ $\mathbb{F}_q$-matrix A where for a given row i, $1 \leq i \leq N$

$$a_{i,j} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } 1 \leq j \leq N \text{ and } i \neq j \\ H(s_i || z_j) & \text{if } N < j \leq 2N \end{cases}$$

Like in the ACV-BGKM scheme, Svr computes the null space of A with a set of its N basis vectors, and selects a vector Y as one of the basis vectors. Svr constructs an $2N$-dimensional $\mathbb{F}_q$-vector

$$ACV = (\sum_{i=1}^{N} k_i \cdot e_i^T) + Y,$$

where e_i is the i^{th} standard basis vector of $\mathbb{F}_q^{2N}$. Notice that, unlike ACV-BGKM, a unique key corresponding to Usr_i, $k_i \in \mathbf{K}$ is embedded into each location corresponding to a valid index i. Like, ACV-BGKM, Svr sets $PI = \langle ACV, (z_1, z_2, \ldots, z_N)\rangle$, and outputs PI via the broadcast channel.

KeyDer(s_i, PI): Usr_i, using its secret s_i and public PI, derives the $2N$-dimensional row $\mathbb{F}_q$-vector v_i which corresponds to a row in A. Then Usr_i derives the specific key as $k_i = v_i \cdot ACV$.

Update(S, K'): If a user leaves or join the group, a new set of keys K' is selected. **KeyGen(S, K')** is invoked to generate the updated public information PI'. Notice that the secrets shared with existing users are not affected by the group change. It outputs the public PI'.

2.6.1 Security Analysis

In this section, we prove the security of the modified ACV-BGKM scheme. Specifically we prove the soundness of the modified ACV-BGKM scheme. We will model the cryptographic hash function H as a random oracle. We further assume that $q = O(2^\ell)$ is a sufficiently large prime power and N is relatively small. We first present an additional lemma with its proof and then prove that the modified ACV-BGKM scheme is indeed sound.

Lemma 4 *Let $F = \mathbb{F}_q$ be a finite field of q elements. Let $v_i = e_i^T + (0, \ldots, 0, v_i^{(n+1)}, \ldots, v_i^{(2n)})$, e_i is the i^{th} standard basis vector of $\mathbb{F}_q^{2n}, i = 1, \ldots, m$, and $1 \leq m \leq n$, be $2n$-dimensional F-vectors. Let $v = e^T + (0, \ldots, 0, v^{(n+1)}, \ldots, v^{(2n)})$ be a $2n$-dimensional F-vector with $v^{(j)}, j \geq n+1$ chosen independently and uniformly at random from F and e from the $2n$-dimensional standard basis vectors with the position of the non-zero element $\leq m$. Then the probability that v is linearly dependent of $\{v_i,\ 1 \leq i \leq m\}$ is no more than $1/q^{n-m}$.*

Proof Let $w_i = (v_i^{(n+1)}, \ldots, v_i^{(2n)}), 1 \leq i \leq m$, $w = (v^{(n+1)}, \ldots, v^{(2n)})$, and $u_i = (v_i^{(1)}, \ldots, v_i^{(n)})$. All w_i span an F-subspace W whose dimension is at most m in an n-dimensional F-vector space. w and u are uniformly randomly chosen n-dimensional F-vectors. By Lemma 1, we have $\Pr[w \in W] = 1/q^{n-\dim(W)} \leq 1/q^{n-m}$. It follows that

$$
\begin{aligned}
&\Pr[v \text{ is linearly dependent of } \{v_i : 1 \leq i \leq m\}] \\
&= \Pr[v = \alpha_1 \cdot v_1 + \ldots + \alpha_m \cdot v_m \text{ for some } \alpha_i \in F] \\
&= \Pr\ \sum_{i=1}^{m} \alpha_i \cdot u_i = e^T \wedge w = \sum_{i=1}^{m} \alpha_i \cdot v_i \text{ for some } \alpha_i \in F \\
&= \Pr\ \sum_{i=1}^{m} \alpha_i \cdot u_i = e^T \ \cdot \Pr[w \in W] \\
&\leq 1/q^n \cdot 1/q^{n-m} = 1/q^{2n-m}.
\end{aligned}
$$

■

Definition 2.6.1 (Soundness of the modified ACV-BGKM scheme) *Let* Usr_i *be an individual without a valid secret and* Usr_j *with a valid secret* s_j, $1 \leq i, j \leq N$. *The modified ACV-BGKM is* sound *if*

- *The probability that* Usr_i *can obtain the correct key* k_i *by substituting the secret with a value* val *that is not one of the valid secrets and then running the key derivation algorithm* ***KeyDer*** *is negligible.*
- *The probability that* Usr_j *can obtain a correct key* k_r, *where* $j = r$ *and* $1 \leq r \leq N$, *by substituting* s_j *and then running the key derivation algorithm* ***KeyDer*** *is negligible.*

Theorem 2.6.1 *The modified ACV-BGKM scheme is sound.*

Proof Let $PI = \langle ACV, (z_1, \ldots, z_N) \rangle$ be the public information broadcast from Svr.
Case 1: Usr_i does not have a valid secret and tries to derive k_i.

Let Y be a vector orthogonal to the access control matrix A.

Let $\{v_i, 1 \leq i \leq N\}$, be a basis of the nullspace of Y.

Let $v = e^T + (0, \ldots, 0, v^{(N+1)}, \ldots, v^{(2N)})$, where $v^{(i+N)} = H(\mathsf{val}||z_i), 1 \leq i \leq N$.

Usr_i can derive the key using v by running the **KeyDer** algorithm if and only if v is linearly dependent from $v_i, 1 \leq i \leq N$. When val is not a valid secret and H is a random oracle, v is indistinguishable from a vector whose first N entries are from e^T and the rest of the N entries are independently and uniformly chosen from $\mathbb{F}_q$. By Lemma 4, the probability that v is linearly dependent from $\{v_i, 1 \leq i \leq N\}$ is no more than $1/q^{2N-N} = 1/q^N$, which is negligible. This proves that the modified ACV-BGKM scheme is sound in case 1.

Case 2: Usr_j has a valid secret s_j and tries to derive k_r, where $r = j$ and $1 \leq r \leq N$.

Since Usr_j has a valid secret s_j, it can construct the j^{th} row of A as follows:

$$v_j = e_j^T + (0, \ldots, 0, v_j^{(N+1)}, \ldots, v_j^{(2N)}), \text{ where } v_j^{(i+N)} = H(s_j||z_i), 1 \leq i \leq N.$$

Usr_j can obtain the key k_j using v_j:

$$k_j = ACV \cdot v_j.$$

In order to obtain the key k_r, Usr_j needs to compute $ACV \cdot v_r$ where v_r is defined as follows.

$$v_r = e_r^T + (0, \ldots, 0, v_r^{(N+1)}, \ldots, v_r^{(2N)}), \text{ where } v_r^{(i+N)} = H(\mathsf{val}||z_i), 1 \leq i \leq N.$$

By construction, v_r is linearly independent from v_j. When val is not a valid secret and H is a random oracle, v_r is indistinguishable from a vector whose first N entries are from e_r^T and the rest of the N entries are independently and uniformly chosen from $\mathbb{F}_q$. Thus, knowing v_j does not provide an advantage for Usr_j to compute v_r. Therefore, the probability of deriving k_r by running the **KeyDer** algorithm remains the same negligible value $1/q^N$ as in case 1. This proves that the modified ACV-BGKM scheme is sound in case 2.

■

2.7 Experimental Results

In this section, we present experimental results for the optimized ACV-BGKM. The experiments were performed on a machine running GNU/Linux kernel version 2.6.32 with an Intel® Core™ 2 Duo CPU T9300 2.50GHz and 4 Gbytes memory. Only one processor was used for computation. The code is built with 32-bit gcc version 4.4.3, optimization flag -O2. For the ACV-BGKM scheme, we use V. Shoup's NTL library [37] version 5.4.2 for finite field arithmetic, and SHA-1 implementation of OpenSSL [38] version 0.9.8 for cryptographic hashing.

We implemented the ACV-GKM scheme with both the bucketization and the subset cover optimizations. We utilized the complete subset algorithm introduced by Naor et. al. [35] for the subset cover. We assumed that 5% of the users satisfying a given Pc are revoked. With the bucketization optimization, we assumed the average case for the KeyDer algorithm where Usrs require to derive half of the intermediate

keys before deriving the group key. For the experiments involving fixed number of buckets, 10 buckets are utilized. All finite field arithmetic operations in our scheme are performed in an 512-bit prime field.

Figure 3.1 reports the average time spent to execute the KeyGen algorithm of the ACV-BGKM scheme without any optimizations, with bucketization, and with subset cover optimization for different group sizes. The bucketization outperforms the base scheme as it divides the non-linear KeyGen algorithm into smaller and more efficient computations. Subset-cover optimization provides even better performance as it reduces the effective group size considerably by sharing secrets among multiple Usrs. As shown in Figure 2.2, the KeyDer algorithm has similar results.

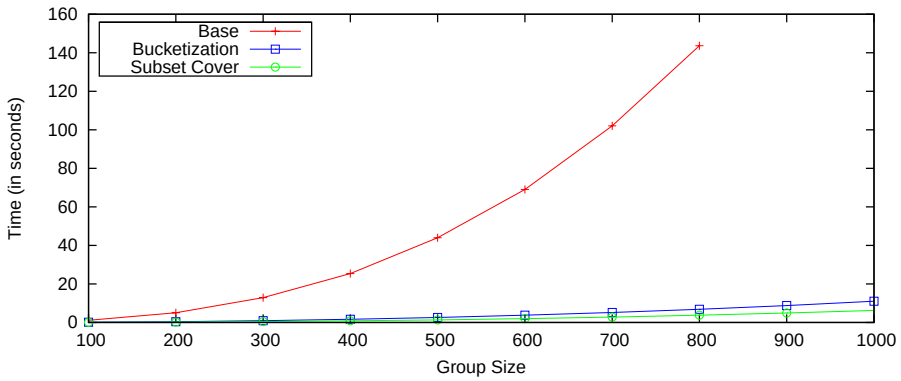

Figure 2.1.: Average time to generate keys

Figure 2.3 shows the average time to execute the KeyGen algorithm for 2500 and 5000 user groups with an increasing number of buckets. When more buckets are utilized, the size of the problem the KeyGen has to solve reduces and, hence, the bucketization provides a better performance. However, as mentioned in Section 2.5.1, the performance starts to degrade as the number of buckets is greater than the the optimal number of buckets. For $n = 2500$ and 5000, the optimal number of buckets are around 100 and 150 respectively. These values are consistent with the theoretical minimum overhead. Under similar settings, Figure 2.4 shows the time to execute the

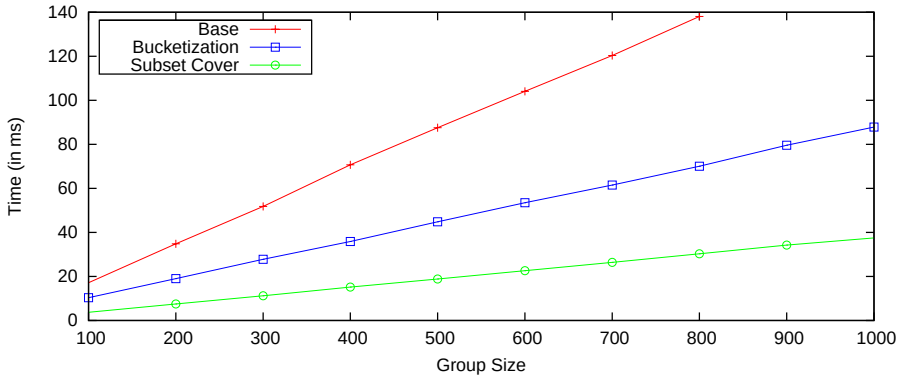

Figure 2.2.: Average time to derive keys

KeyDer algorithm. The key derivation time slowly increases as the number of buckets increases because the complexity of the second level KeyDer function increases.

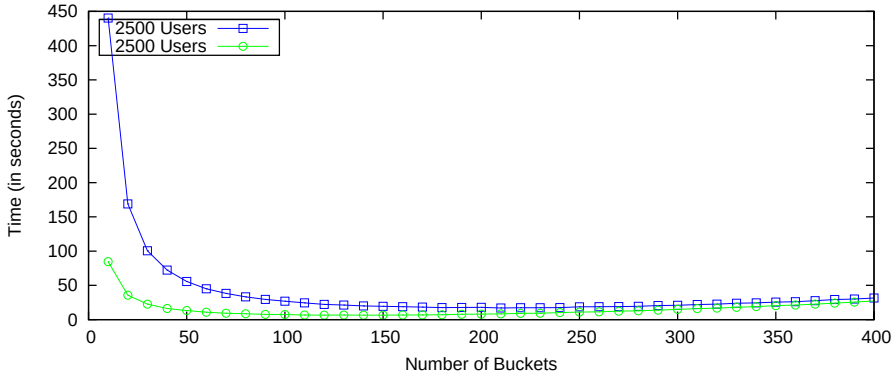

Figure 2.3.: Average time to generate keys with different bucket sizes

We closely analyzed the two optimizations. Figure 2.5 shows the average time to execute the KeyGen algorithm with the bucketization, the subset cover and both where the bucketization is applied after the subset cover technique. Both techniques together provides a huge performance improvement. Under the similar setting, as shown in Figure 2.6, the KeyGen also performs much better compared to the individual optimizations.

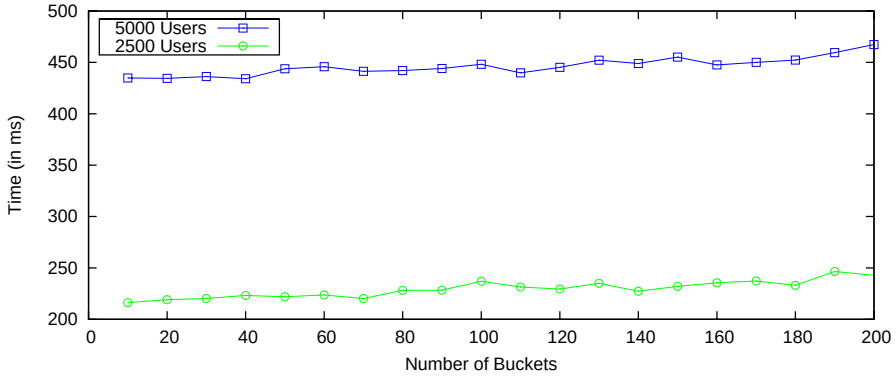

Figure 2.4.: Average time to derive keys with different bucket sizes

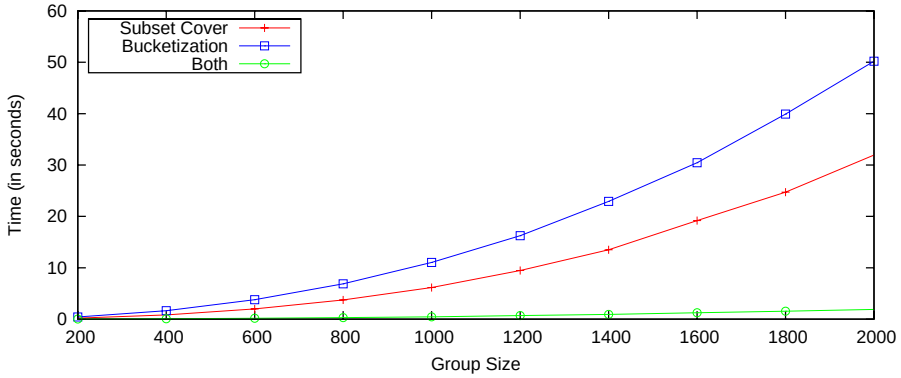

Figure 2.5.: Average time to generate keys with the two optimizations

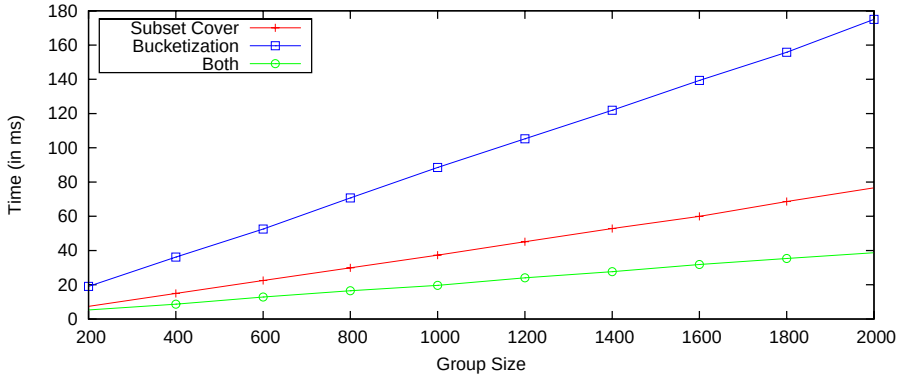

Figure 2.6.: Average time to derive keys with the two optimizations

3 ATTRIBUTE BASED GROUP KEY MANAGEMENT

While BGKM schemes provide efficient rekeying, they do not support expressive group membership policies over a set of attributes. In their basic form, they can only support 1-*out-of-n* threshold policies by which a group member possessing 1 attribute out of the possible n attributes is able to derive the group key. In order to address this issue, in this chapter, we develop novel expressive attribute based GKM (AB-GKM) schemes which allow one to express any threshold or monotonic policies over a set of attributes.

A possible approach to construct an AB-GKM scheme is to utilize attribute-based encryption (ABE) primitives [16–18]. Such an approach would work as follows. A key generation server issues each group member a private key (a set of secret values) based on the attributes and the group membership policies. The group key, typically a symmetric key, is then encrypted under a set of attributes using the ABE encryption algorithm and broadcast to all the group members. The group members whose attributes satisfy the group membership policy can obtain the group key by using the ABE decryption primitive. One can use such an approach to implement an expressive collusion-resistant AB-GKM scheme. However, such an approach suffers from some major drawbacks. Whenever the group dynamic changes, the rekeying operation requires to update the private keys given to existing members in order to provide backward/forward secrecy. This in turn requires establishing private communication channels with each group member which is not desirable in a large group setting. Further, in applications involving stateless members where it is not possible to update the initially given private keys and the only way to revoke a member is to exclude it from the public information, an ABE based approach does not work. Another limitation is that whenever the group membership policy changes, new private

keys must be re-issued to members of the group. Our constructions address these shortcomings.

Our AB-GKM schemes are able to support a large variety of conditions over a set of attributes. When the group changes, the rekeying operations do not affect the private information of existing group members and thus our schemes eliminate the need of establishing private communication channels. Our schemes provide the same advantage when the group membership conditions change. Furthermore, the group key derivation is very efficient as it only requires a simple vector inner product and/or polynomial interpolation. Additionally, our schemes are resistant to collusion attacks. Multiple group members are unable to combine their private information in a useful way to derive a group key which they cannot derive individually.

Our AB-GKM constructions are based on an optimized version of the ACV-BGKM (Access Control Vector BGKM) scheme presented in Chapter 2, a provably secure BGKM scheme, and Shamir's threshold scheme [29]. In this paper, we construct three AB-GKM schemes each of which is more suitable over others under different scenarios. The first construction, inline AB-GKM, is based on the ACV-BGKM scheme. Inline AB-GKM supports arbitrary monotonic policies over a set of attributes. In other words, a user whose attributes satisfy the group policies is able to derive the symmetric group key. However, inline AB-GKM does not efficiently support *d-out-of-m* ($d \leq m$) attribute threshold policies over m attributes. The second construction, threshold AB-GKM, addresses this requirement. The third construction, access tree AB-GKM, is an extension of threshold AB-GKM and is the most expressive scheme. It efficiently supports arbitrary policies. The second and third schemes are constructed by using a modified version of ACV-BGKM, also proposed in this paper.

3.1 Scheme 1: Inline AB-GKM

Recall that in its basic form, a BGKM scheme can be considered as a 1-*out-of-m* AB-GKM scheme. If Usr_i possesses the attribute attr_j, Svr shares a unique secret

$s_{i,j}$ with Usr_i. Usr_i is thus able to derive the symmetric group key if and only if Usr_i shares at least one secret with Svr and that secret is included in the computation of the public information tuple PI. In order for Svr to revoke Usr_j, it only needs to remove the secrets it shares with Usr_j from the computation of PI; the secrets issued to other group members are not affected. We extend this scheme to support arbitrary monotonic policies, ACPs, over a set of attributes. A user is able to derive the symmetric group key if and only if the set of attributes the user possesses satisfy ACP.

As in the basic BGKM scheme, Usr_i having attr_j is associated with a unique secret value $s_{i,j}$. However, unlike the basic BGKM scheme, PI is generated by using the aggregated secrets that are generated combining the secrets issued to users according to ACP. For example, if ACP is a conjunction of two attributes, that is $\mathsf{attr}_r \wedge \mathsf{attr}_s$, the corresponding secrets $s_{i,r}$ and $s_{i,s}$ for each Usr_i are combined as one aggregated secret $s_{i,r}||s_{i,s}$ and PI is computed using these aggregated secrets. By construction, the aggregated secrets are unique since the constituent secrets are unique. Any Usr_i is able to derive the symmetric group key if and only if Usr_i has at least one aggregated secret used to compute PI. Notice that multiple users cannot collude to create an aggregated secret which they cannot individually create since $s_{i,j}$'s are unique and each aggregated secret is tied to one specific user. Hence, colluding users cannot derive the group symmetric key. Now we give a detailed description of our first AB-GKM scheme, inline AB-GKM.

3.1.1 Our Construction

Inline AB-GKM consists of the following five algorithms:

Setup(ℓ): The Svr initializes the following parameters: an ℓ-bit prime number q, a cryptographic hash function $H(\cdot) : \{0,1\}^* \rightarrow \mathbb{F}_q$, where $\mathbb{F}_q$ is a finite field with q elements, the keyspace $\mathcal{KS} = \mathbb{F}_q$, the secret space $\mathcal{SS} = \{0,1\}^\ell$, and the set of issued secrets $\mathbf{S} = \emptyset$. The user-attribute matrix UA is initialized with empty elements and

the maximum group size N is decided in the **KeyGen**. It defines the universe of attributes $\mathcal{A} = \{\mathsf{attr}_1, \mathsf{attr}_2, \cdots, \mathsf{attr}_m\}$.

SecGen(γ_i): For each attribute $\mathsf{attr}_j \in \gamma_i$, where $\gamma_i \subset \mathcal{A}$ and γ_i is the attribute set of Usr_i, the Svr chooses the secret $s_{i,j} \in \mathcal{SS}$ uniformly at random for Usr_i such that $s_{i,j} \notin \mathbf{S}$, adds $s_{i,j}$ to $\mathbf{S}$, sets $UA(i,j) = s_{i,j}$, where $UA(i,j)$ is the $(i,j)^{th}$ element of the user-attribute matrix UA, and finally outputs $s_{i,j}$.

KeyGen(ACP): We first give a high-level description of the algorithm and then the details. Svr transforms the policy ACP to disjunctive normal form (DNF). For each disjunctive clause of ACP in DNF, it creates an aggregated secret ($\widehat{s}$) from the secrets corresponding to each of the attributes in the conjunctive clause. $\widehat{s}$ is formed by concatenation only if secrets exist for all the attributes in a given row of the user-attribute matrix UA. The construction creates a unique aggregated secret $\widehat{s}$ since the corresponding secrets are unique. For example, if the conjunctive clause is $\mathsf{attr}_p \wedge \mathsf{attr}_q \wedge \mathsf{attr}_r$, for each row i in UA, the aggregated secret $\widehat{s}_i$ is formed only if all elements $UA(i,p)$, $UA(i,q)$ and $UA(i,r)$ have secrets assigned. All the aggregated secrets are added to the set **AS**. Finally, Svr invokes algorithm **KeyGen**(**AS**) from the underlying BGKM scheme to output the public information PI and the symmetric group key k.

Now we give the details of the algorithm. Svr converts ACP to DNF as follows

$$\mathsf{ACP} = \bigvee_{i=1}^{\alpha} \mathsf{conjunct}_i \text{ where there are } \alpha \text{ conjuncts and}$$

$$\mathsf{conjunct}_i = \bigwedge_{j=1}^{\phi_i} \mathsf{cond}_j^{(i)},$$

where each $\mathsf{conjunct}_i$ has ϕ_i conditions.

A simple multiplication of clauses ($x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$) and then application of the absorption law ($x \vee (x \wedge y = x)$) are sufficient to convert monotone policies to DNF. Even though there can be an exponential blow up of clauses during

multiplication, it has been shown that with the application of the absorption law the number of clauses in the DNF, at the end, is always polynomially bounded. Svr selects N such that

$$N \geq \sum_{i=1}^{\alpha} N_{U_i} = N_U$$

where N_{U_i} is the number of users satisfying $\mathsf{conjunct}_i$ [1]. Svr creates N_U $\widehat{s}_i$'s and adds them to **AS**. Svr picks a random $k \in \mathcal{KS}$ as the shared group key. Svr chooses N random bit strings $z_1, z_2, \ldots, z_N \in \{0,1\}^\ell$. Svr creates an $m \times (N+1)$ $\mathbb{F}_q$-matrix A such that for $1 \leq i \leq N_U$

$$a_{i,j} = \begin{cases} 1 & \text{if } j = 1 \\ H(\widehat{s}_i || z_j) & \text{if } 2 \leq j \leq N; \widehat{s}_i \in \mathbf{AS} \end{cases} \tag{3.1}$$

Svr then solves for a nonzero $(N+1)$-dimensional column $\mathbb{F}_q$-vector Y such that $AY = 0$ and sets

$$ACV = k \cdot e_1^T + Y, \text{ and}$$
$$PI = \langle ACV, (z_1, z_2, \ldots, z_N) \rangle$$

KeyDer(β_i, PI): Given β_i, the set of secrets for Usr_i, it computes the aggregated secret $\widehat{s}$. Using $\widehat{s}$ and the public information PI, it computes $a_{i,j}, 1 \leq j \leq N$, as in formula 3.1 and sets an $(N+1)$-dimensional row $\mathbb{F}_q$-vector $v_i = (1, a_{i,1}, a_{i,2}, \ldots, a_{i,N})$. Usr_i derives the group key k' by the inner product of the vectors v_i and ACV: $k' = v_i \cdot ACV$. The derived group key k' is equal to the actual group key k if and only if the computed aggregated secret $\widehat{s} \in \mathbf{AS}$.

Update(S): The composition of the user group changes when one of the following occurs:

[1] It should be noted that N_U can be reduced to n, the number of users in the group, by exploiting the relationships between conjuncts and letting the users know the conjunct, out of the many they satisfy, they have to use to derive the key. We leave this optimization to keep the scheme simple.

- Identity attributes are added or removed resulting in the change in S and UA [2].
- The underlying policy ACP changes.

When such a change occurs, a new symmetric key k' is selected and **KeyGen(ACP)** is invoked to generate the updated public information PI'. Notice that the secrets shared with existing users are not affected by the group change. It outputs the public PI' and private k'.

3.1.2 Security

We can easily show that if an *unbounded* adversary $\mathcal{A}$ can break the inline AB-GKM scheme in the random oracle model, a simulator $\mathcal{S}$ can be constructed to break the ACV-BGKM scheme.

Definition 3.1.1 (Security game for AB-GKM)
Setup *The challenger runs the Setup algorithm of AB-GKM and gives the public parameters to the adversary.*

Phase 1 *The adversary is allowed to request secrets for any set of attributes γ_i and the public information tuples for a policy satisfying these attributes. The public information along with the secrets allows the adversary to derive the private key.*

Challenge *The adversary declares the set of attributes γ that it wishes to challenged upon. γ is different from any of the attribute sets γ_i that the adversary queried earlier. The adversary submits two keys k_0 and k_1. The challenger flips a random coin b and chooses k_b. The challenger generates public information for a policy P satisfying γ, but not any γ_i, using the **KeyGen** algorithm and give it to the adversary. The public information hides the group key k_b.*

[2] A change in a user attribute is viewed as two events; removing the existing attribute and adding a new attribute.

Phase 2 *Phase 1 is repeated as many times provided that the adversary's attribute set does not satisfy P.*

Guess *The adversary outputs a guess b' of b.*

The advantage of an adversary $\mathcal{A}$ in this game is defined as $Pr[b' = b] - 1/2$.

Definition 3.1.2 (Security under the random oracle model) *An AB-GKM scheme is secure under the random oracle model of security if all adversaries have at most a negligible advantage in the above game.*

Shang et al. [20, 39] have shown that the probability of breaking ACV-BGKM is a negligible $1/q$, where q is the ℓ bit large prime number initialized in **Setup**. We capture the hardness of the ACV-BGKM scheme in the following assumption:

Definition 3.1.3 (ACV-BGKM Assumption) *No adversary without any valid secrets in the random oracle model can break the ACV-BGKM scheme with more than a negligible probability.*

Theorem 3.1.1 *If an adversary can break the inline AB-GKM scheme in the random oracle model, then a simulator can be constructed to break the ACV-BGKM scheme with non-negligible advantage.*

Proof Suppose that there exists an adversary $\mathcal{A}$ that can break our scheme in the random oracle model with advantage ϵ. We build a simulator $\mathcal{B}$ that can break the ACV-BGKM scheme with the advantage at most ϵ. The simulation proceeds as follows:

The challenger runs the setup algorithm of ACV-BGKM and generates secrets for each attributes per user outside of $\mathcal{B}$'s view. The simulator $\mathcal{B}$ runs $\mathcal{A}$. $\mathcal{B}$ is given an instance of ACV-BGKM and gives the public parameters to $\mathcal{A}$. We assume that all policies are in DNF such that each conjunctive term has only one attribute. The intuition behind the assumption is that inline AB-GKM is an extension of ACV-BGKM

to support aggregate secrets and, therefore, in the absent of aggregate secrets, inline AB-GKM is equivalent to ACV-BGKM.

Phase 1 $\mathcal{A}$ submits sets of attributes γ_i to $\mathcal{B}$ and $\mathcal{B}$ sends the secrets using the ACV-BGKM instance.

Challenge $\mathcal{A}$ submits the attribute set $\gamma = \gamma_i$ as the challenge and two keys k_0 and k_b. $\mathcal{B}$ flips a random coin b and chooses k_b and then using the ACV-BGKM instance, it generates the public information for a policy P that only γ satisfies hiding k_b.

Phase 2 $\mathcal{A}$ and $\mathcal{B}$ repeats Phase 1 as many times provided $\mathcal{A}$'s attribute sets do not satisfy P.

Guess Using the public information and the information gathered from the two phases, $\mathcal{A}$ outputs a guess b' of b. Notice that the view of $\mathcal{A}$ when it is run as a subroutine of $\mathcal{B}$ and when it is run directly with the inline AB-GKM scheme is identical. In other words, $\mathcal{B}$ simulates an instance of the inline AB-GKM for $\mathcal{A}$ using an instance of the ACV-BGKM scheme. The simulation is trivial as the aggregate secrets in AB-GKM is the same the secrets in ACV-BGKM. It should be noted that $\mathcal{A}$ does not have an advantage more than ϵ from the information gather from the repeated execution of Phase 1 due to the key indistinguishability and key independence properties of the ACV-BGKM scheme [39].

It can easily be seen that $\mathcal{B}$ has the same advantage of breaking the ACV-BGKM scheme as $\mathcal{A}$ has on the inline AB-GKM scheme. As per the definitions, $\mathcal{B}$ breaks the ACV-BGKM with $Pr[b' = b] = 1/2 + \epsilon$. According to the assumption on the hardness of the ACV-BGKM scheme in Theorem 3.1.1, it follows that ϵ must be negligible.

■

3.1.3 Performance

Now, we discuss the efficiency of inline AB-GKM with respect to computational costs and required bandwidth for rekeying.

For any Usr_i in the group, deriving the shared group key requires N hashing operations (evaluations of $H(\cdot)$) and an inner product computation $v_i \cdot ACV$ of two $(N+1)$-dimensional $\mathbb{F}_q$-vectors, where N is the maximum group size. Therefore the overall computational complexity is $O(n)$.

For every rekeying operation, Svr needs to form a matrix A by performing N^2 hashing operations, and then solve a linear system of size $N \times (N+1)$. Solving the linear system is the most costly operation as N gets large for computation on Svr. It requires $O(n^3)$ field operations in $\mathbb{F}_q$ when the method of Gauss-Jordan elimination [34] is applied. Experimental results about the ACV-BGKM scheme [20] have shown that this can be performed in a short time when N is small.

When a rekeying process takes place, the new information to be broadcast is $PI = \langle ACV, (z_1, \ldots, z_N) \rangle$, where ACV is a vector consisting of $(N+1)$ elements in $\mathbb{F}_q$, and without loss of generality we can pick z_i to be strings of fixed length. This gives an overall communication complexity $O(n)$. An advantage of inline AB-GKM is that no peer-to-peer private channel is needed for any persisting group members when rekeying is executed.

Nowadays we generally care less about storage costs on both Svr and Usrs. Nevertheless, for a group of maximum N users, in the worst case, inline AB-GKM only requires each Usr to store $(O(|\mathcal{A}|))$ secrets, one secret per attribute that Usr possesses, and Svr to keep track of all $O(n|\mathcal{A}|)$ secrets.

3.2 Scheme 2: Threshold AB-GKM

Consider now the case of policies by which a user can derive the symmetric group key k, if it possesses at least d attributes out of the m attributes associated with the group. We refer to such policies as *threshold policies*. Under the inline AB-GKM

scheme presented in Section 3.1, with such threshold policies the size of the access control matrix (A) increases exponentially if users are not informed which attributes to use. Specifically, to support *d-out-of-m*, the inline AB-GKM scheme may require creating a matrix of dimension up to $O(nm^d)$ where n is the number of users in the group. Thus, the inline AB-GKM scheme is not suitable for threshold policies. In this section, we construct a new scheme, threshold AB-GKM, which overcomes this shortcoming.

An initial construction to enforce threshold policies is to associate each user with a random $d-1$ degree polynomial, $q(x)$, with the restriction that each polynomial has the same value at $x = 0$ and $q(0) = k$, where k is the symmetric group key. For each attribute users have, they are given a secret value. The secret values given to a user are tied to its random polynomial $q(x)$. A user having d or more secrets can perform a Lagrange interpolation to obtain $q(x)$ and thus the symmetric group key $k = q(0)$. Since the secrets are tied to random polynomials, multiple users are unable to combine their secrets in any way that makes possible collusion attacks. However, revocation is difficult in this simple approach and requires re-issuing all the secrets again.

Our approach to address the revocation problem is to use a layer of indirection between the secrets given to users and the random polynomials such that revocations do not require re-issuing all the secrets again. We use a modified ACV-BGKM construction as the indirection layer. We cannot directly use the ACV-BGKM construction since, multiple instances of ACV-BGKM allow collusion attacks in which colluding users can recover the group key which they cannot obtain individually. We first show the details of the modified ACV-BGKM scheme and then present the threshold AB-GKM which uses the modified ACV-BGKM scheme and Shamir's secret sharing scheme.

3.2.1 Our Construction

Now we provide our construction of the threshold AB-GKM scheme which utilizes the modified ACV-BGKM scheme, ACV-BGKM-2, presented in Section 2.6.

Recall that in this scheme, we wish to allow a user to derive the symmetric group key k if the user possesses at least d attributes out of m. For each user Usr_i we associate a random $d-1$ degree polynomial $q_i(x)$ with the restriction that each polynomial has the same value k, the symmetric group key, at $x = 0$, that is, $q_i(0) = k$. We associate a random secret value with each user attribute. For each attribute attr_i, we generate a public information tuple (PI_i) using the modified ACV-BGKM scheme with the restriction that the temporary key that each Usr_j derives is tied to its random polynomial $q_j(x)$, that is $q_j(i) = k_i$. Notice that each user obtains different temporary keys from the same PI. If a user can derive d temporary keys corresponding to d attributes, it can compute its random function $q(x)$ and obtain the group symmetric key k. Notice that, since the temporary keys are tied to a unique polynomial, multiple users are unable to collude and combine their temporary keys in order to obtain the symmetric group key which they are not allowed to obtain individually. Thus, our construction prevents collusion attacks.

A detailed description of our threshold AB-GKM scheme follows.

Setup(ℓ) Svr initializes the parameters of the underlying modified ACV-BGKM scheme: the ℓ-bit prime number q, the maximum group size N ($\geq n$), the cryptographic hash function H, the key space $\mathcal{KS}$, the secret space $\mathcal{SS}$, the set of issued secrets $\mathbf{S}$, the user-attribute matrix UA and the universe of attributes $\mathcal{A} = \{\mathsf{attr}_1, \mathsf{attr}_2, \cdots, \mathsf{attr}_m\}$.

Svr defines the Lagrange coefficient $\Delta_{i,\mathbf{Q}}$ for $i \in \mathbb{F}_q$ and a set, $\mathbf{Q}$ of elements in $\mathbb{F}_q$ as

$$\Delta_{i,\mathbf{Q}}(x) = \prod_{j \in \mathbf{Q}, j \neq i} \frac{x-j}{i-j}.$$

SecGen(γ_i) For each attribute $\mathsf{attr}_j \in \gamma_i$, where $\gamma_i \subset \mathcal{A}$ and γ_i is the attribute set of

Usr_i, Svr invokes **SecGen()** of the modified ACV-BGKM scheme in order to obtain the random secret $s_{i,j}$. It returns β_i, the set of secrets for all the attributes in γ_i.

KeyGen(α, d) Taking α, a subset of attributes from the attribute universe $\mathcal{A}$ and d, the threshold value, for each user Usr_i, Svr assigns a random degree $d-1$ polynomial $q_i(x)$ with $q_i(0)$ set to the group symmetric key k. For each attribute attr_j in the set of attributes α ($\alpha \subset \mathcal{A}$ and $|\alpha| \geq d$), it selects the set of secrets corresponding to attr_j, $\mathbf{S}_j$ and invokes **KeyGen(S_j, $\{q_1(j), q_2(j), \cdots, q_N(j)\}$)** of the modified ACV-BGKM scheme to obtain PI_j, the public information tuple for attr_j. It outputs the private group key k and the set of public information tuples $\mathbf{PI} = \{PI_j |$ for each $\mathsf{attr}_j \in \alpha\}$.

KeyDer(β_i, PI) Using the set of d secrets $\beta_i = \{s_{i,j} | 1 \leq j \leq N\}$ for the d attributes attr_j, $1 \leq j \leq N$, and the corresponding d public information tuples $PI_j \in \mathbf{PI}$, $1 \leq j \leq N$, it derives the group symmetric key k as follows.

First, it derives the temporary key k_j for each attribute attr_j using the underlying modified ACV-BGKM scheme as **KeyDer**($s_{i,j}$, PI_j). Then, using the set of d points $\mathbf{Q}_i = \{(j, k_j) | 1 \leq j \leq N\}$, it computes $q_i(x)$ as follows:

$$\Delta_{j,\mathbf{Q}_i}(x) = \prod_{j \in \mathbf{Q}_i, j \neq i} \frac{x-j}{i-j}$$

$$q_i(x) = \sum_{j \in \mathbf{Q}_i} k_j \Delta_{j,\mathbf{Q}_i}(x).$$

It outputs the group key $k = q_i(0)$.

Update(α, d) The Update algorithm is invoked whenever α, the attribute set considered, or d, the threshold value, or the group members satisfying the threshold policy change. The group membership changes due to similar reasons mentioned under the **Update** algorithm in Section 3.1.1. In such a situation, a new symmetric group key k' is selected and **KeyGen(α, d)** is invoked to generate the set of new public infor-

mation tuples **PI**'. Notice that the secrets shared with existing users are not affected by the group change.

3.2.2 Security

If an unbounded adversary can break our threshold AB-GKM scheme, a simulator can be constructed to break the modified ACV-BGKM scheme. We only give a high-level detail of the reduction based proof as the proof is similar to the proof for the inline AB-GKM scheme.

Proof Suppose that an unbounded adversary $\mathcal{A}$ having a set of $d-1$ attributes α can break our scheme in the random oracle model with advantage ϵ. Note that this is the most powerful adversary as it possesses $d-1$ attributes out of the d attributes required to derive the group key. We build a simulator $\mathcal{B}$ that can derive the key k_d from PI_d corresponding to $\mathsf{attr}_d \notin \alpha$ with the same advantage ϵ using $\mathcal{A}$ as subroutine. In other words, we build a simulator to break the modified ACV-BGKM scheme.

The intuition behind our proof is that, by construction, the modified ACV-BGKM instances corresponding to the attributes are independent. In other words, a user who can access the key for one attribute only has a negligible advantage in obtaining the key for another attribute using the known attributes due to the key indistinguishability and independence properties of the ACV-BGKM scheme.

The challenger creates an instance of the modified ACV-BGKM scheme for each of the n attributes. $\mathcal{A}$ obtains secrets $\{s_i | i = 1, 2, \cdots, d-1\}$ for the attributes α it has from $\mathcal{B}$. The challenger constructs the public information tuples $\{PI_i | i = 1, 2, \cdots, d\}$, each having a random key k_i and gives them to $\mathcal{B}$. $\mathcal{B}$ in turn gives them to $\mathcal{A}$. Notice that the view of $\mathcal{A}$ is identical to that of $\mathcal{A}$ interacting directly with an instance of the threshold AB-GKM scheme, even though it is simulated. The random keys correspond to a random degree $d-1$ polynomial $q(x)$. Notice that $\mathcal{A}$ possesses secrets to obtain the random keys k_i, $1 \leq i \leq d-1$ and can derive the secret key k_d with an advantage ϵ from the public information tuples.

We omit the details of the security game defined in the previous section. As mentioned in the game, $\mathcal{A}$ may execute the threshold AB-GKM scheme for different sets of attributes that do not satisfy the challenge threshold policy and do not include $attr_d$. As mentioned earlier, $\mathcal{A}$ does not gain any additional advantage by such executions.

After executing the phase 1 of the security game as many times, $\mathcal{A}$ outputs k, which is equal to $q(0)$. This allows $\mathcal{B}$ to fully determine $q(x)$ as it now has d points and derive the key $k_d = q(d)$. In other words, it allows $\mathcal{B}$ to break the modified ACV-BGKM scheme to recover the intermediate key k_d from the public information tuple PI_d without the knowledge of the secret s_d. In our technical report [40], we show that the probability of breaking the modified ACV-BGKM scheme is a negligible $1/q^N$ where q is the ℓ bit prime number and N is the maximum number of users. Therefore, it follows that ϵ must be negligible. ■

3.2.3 Performance

We now discuss the efficiency of the threshold AB-GKM with respect to computational costs and required bandwidth for rekeying.

For any Usr_i in the group deriving the shared group key requires: $\sum_{i=1}^{d} N_i$ hashing operations (evaluations of $H(\cdot)$), where N_i is the maximum number of users having attr_i; and d inner product computations $v_i \cdot ACV_i$ of two $(2N_i)$-dimensional $\mathbb{F}_q$-vectors and the Lagrange interpolation $O(m \log^2 m)$, where $m = |\mathcal{A}|$. Therefore, the overall computational complexity is $O(dn + m \log^2 m)$. Notice that the inner product computations are independent and can be parallelized to improve performance.

For every rekeying phase, for each attr_i, Svr needs to form a matrix A_i by performing N_i^2 hashing operations, and then solve a linear system of size $N_i \times (2N_i)$. Solving the linear system is the most costly operation as N_i gets large for computation on Svr; it requires $O(\sum_{i=1}^{m} n^3)$ field operations in $\mathbb{F}_q$.

When a rekeying process takes place, the new information to be broadcast is $PI_i = \langle ACV_i, (z_1, \ldots, z_{N_i}) \rangle$, $i = 1, 2, \cdots, m$, where ACV_i is a vector consisting of

$(2N_i)$ elements in $\mathbb{F}_q$, and without loss of generality we can pick z_i to be strings with a fixed length. This gives an overall communication complexity $O(\sum_{i=1}^{m} n)$.

For a group of maximum N users, in the worst case, the threshold AB-GKM only requires each Usr to store $(O(m))$ secrets, one secret per attribute that Usr possesses and Svr to keep track of all $O(nm)$ secrets.

3.3 Scheme 3: Access Tree AB-GKM

In the inline AB-GKM scheme, the policy ACP is embedded into the BGKM scheme itself. As discussed in Section 3.2, while this approach works for many different types of policies, such an approach is not able to efficiently support threshold access control policies. Scheme 2, threshold AB-GKM, on the other hand, is able to efficiently support threshold policies, but it is unable to support other policies. In order to support more expressive policies, we extend the threshold AB-GKM scheme. Like threshold AB-GKM, instead of embedding ACP in the BGKM scheme, we construct a separate BGKM instance for each attribute. Then, we embed ACP in an access structure $\mathcal{T}$. $\mathcal{T}$ is a tree with the internal nodes representing threshold gates and the leaves representing attributes. The construction of $\mathcal{T}$ is similar to that of the approach by Goyal et al. [17]. However, unlike Goyal et al.'s approach, the goal of our construction is to derive the group key for the users whose attributes satisfy the access structure $\mathcal{T}$.

3.3.1 Access Tree

Let $\mathcal{T}$ be a tree representing an access structure. Each internal node of the tree represents a threshold gate. A threshold gate is described by its child nodes and a threshold value. If n_x is the number of children of a node x and t_x is its threshold value, then $0 < t_x \leq n_x$. Notice that when $t_x = 1$, the threshold gate is an OR gate and when $t_x = n_x$, it is an AND gate. Each leaf node x of the tree is described by

Table 3.1: Access tree functions

Function	**Description**
$\mathsf{index}(x)$	Returns the index of node x
$\mathsf{parent}(x)$	Returns the parent node of node x
$\mathsf{attr}(x)$	Returns the index of the attribute associated with a leaf node x
q_x	The polynomial assigned to node x
$\mathsf{sat}(\mathcal{T}_x, \alpha)$	Returns 1 if the set of attributes α satisfies $\mathcal{T}_x$, the subtree rooted at node x, and 0 otherwise

an attribute, a corresponding BGKM instance and a threshold value $t_x = 1$. The children of each node x are indexed from 1 to n_x.

We define the functions in Table 3.1 in order to construct our scheme. All the functions except sat are straightforward to implement. A brief description of sat follows:

The function $\mathsf{sat}(\mathcal{T}_x, \alpha)$ works as a recursive function. If x is a leaf node, it returns 1, provided that the attribute associated with x is in the set of attributes α and 0 otherwise. If x is an internal node, if at least t_x child nodes of x return 1, then $\mathsf{sat}(\mathcal{T}_x, \alpha)$ returns 1 and 0 otherwise.

3.3.2 Our Construction

The access tree AB-GKM scheme consists of five algorithms:

Setup(ℓ): Svr initializes the parameters of the underlying modified ACV-BGKM scheme: the prime number q, the maximum group size N ($\geq n$), the cryptographic hash function H, the key space $\mathcal{KS}$, the secret space $\mathcal{SS}$, the set of issued secrets $\mathbf{S}$, the user-attribute matrix UA and the universe of attributes $\mathcal{A} = \{\mathsf{attr}_1, \mathsf{attr}_2, \cdots, \mathsf{attr}_m\}$.

Svr defines the Lagrange coefficient $\Delta_{i,\mathbf{Q}}$ for $i \in \mathbb{F}_q$ and a set, $\mathbf{Q}$ of elements in $\mathbb{F}_q$:

$$\Delta_{i,\mathbf{Q}}(x) = \prod_{j \in \mathbf{Q}, j \neq i} \frac{x-j}{i-j}.$$

SecGen(γ_i): Taking γ_i, the attribute set of Usr_i, as input, for each attribute $\mathsf{attr}_j \in \gamma_i$, where $\gamma_i \subset \mathcal{A}$, Svr invokes **SecGen()** of the modified ACV-BGKM scheme to obtain the random secret $s_{i,j}$. It returns β_i, the set of secrets for all the attributes in γ_i.

KeyGen(ACP): Svr transforms the policy ACP into an access tree $\mathcal{T}$. The algorithm outputs the public information which a user can use to derive the group key if and only if the user's attributes satisfy the access tree $\mathcal{T}$ built for the policy ACP. The algorithm constructs the public information as follows.

For each user Usr_i having the intermediate set of keys $\mathbf{K}_i = \{k_{i,j} | 1 \leq j \leq m\}$, where $k_{i,j}$ represents the intermediate key for Usr_i and attr_j, the following construction is performed. For each attribute attr_i, there is a leaf node in $\mathcal{T}$. The construction of the tree is performed top-down. Each node x in the tree is assigned a polynomial q_x. The degree d_x of the polynomial q_x is set to $t_x - 1$, that is, one less than the threshold value of the node. For the root node r, $q_r(0)$ is set to the group key k and d_r other points are chosen uniformly at random so that q_r is a unique polynomial of degree d_r fully defined through Lagrange interpolation. For any other node x, $q_x(0)$ is set to $q_{parent(x)}(index(x))$ and d_x other points are chosen uniformly at random to uniquely define q_x. For each leaf node x corresponding to a unique attribute attr_j, $q_x(0)$ is set to $q_{parent(x)}(1)$ and $k_{i,j} = q_x(0)$.

At the end of the above computation, we have all the sets of intermediate keys $\mathbf{K} = \{\mathbf{K}_i | \mathsf{Usr}_i, 1 \leq i \leq N\}$. For each leaf node x, the modified BGKM algorithm **KeyGen($\mathbf{S}_x$, $\mathbf{K}_x$)**, where $\mathbf{S}_x$ is the set of secrets corresponding to the attribute associated with the node x and $\mathbf{K}_x = \{k_{i,j} | 1 \leq i \leq N, \mathsf{attr}_j\}$, $j = \mathsf{attr}(x)$, is invoked to

generate public information tuple PI_x. We denote the set of all the public information tuples $\mathbf{PI} = \{PI_j | \mathsf{attr}_j, 1 \leq j \leq m\}$.

KeyDer(β_i, PI): Given β_i, a set of secret values corresponding to the attributes of Usr_i, and the set of public information tuples **PI**, it outputs the group key k.

The key derivation is a recursive procedure that takes β_i and **PI** to derive k bottom-up. Note that a user can obtain the key if and only if its attributes satisfy the access tree $\mathcal{T}$, i.e., $\mathsf{sat}(\mathcal{T}_r, \beta_i) = 1$. The high-level description of the key derivation is as follows.

For each leaf node x corresponding to the attribute with the user's secret value $s_x \in \beta_i$, the user derives the intermediate key k_x using the underlying modified BGKM scheme $\mathbf{KeyDer}(s_x, PI_x)$. Using Lagrange interpolation, the user recursively derives the intermediate key k_x for each internal ancestor node x until the root node r is reached and $k_r = k$. Notice that since intermediate keys are tied to unique polynomials, users cannot collude to derive the group key k if they are unable to derive it individually. A detailed description follows.

If x is a leaf node, it returns an empty value $\perp$ if $\mathsf{attr}(x) \in \beta_i$, otherwise it returns the key $k_x = v_x \cdot ACV_x$, where v_x is the key derivation vector corresponding to the attribute $\mathsf{attr}_{\mathsf{attr}(x)}$ and ACV_x the access control vector in PI_x.

If x is an internal node, it returns an empty value $\perp$ if the number of children nodes having a non-empty key is less than t_x, otherwise it returns k_x as follows:

Let the set $\mathbf{Q}_x$ contain the indices of t_x children nodes having non-empty keys $\{k_i | i \in \mathbf{Q}_x\}$.

$$\Delta_{i,\mathbf{Q}_x}(y) = \prod_{i \in \mathbf{Q}_x, i \neq j} \frac{y - i}{j - i}$$

$$q_x(y) = \sum_{i \in \mathbf{Q}_x} k_i \Delta_{i,\mathbf{Q}_x}(y)$$

$$k_x = q_x(0).$$

The above computation is performed recursively until the root node is reached. If Usr_i satisfies $\mathcal{T}$, Usr_i gets $k = q_r(0)$, where r is the root node. Otherwise, Usr_i gets an empty value $\perp$.

Update(ACP) The group members change due to the similar reasons mentioned for the **Update** algorithm in Section 3.1.1. In such a situation, a new symmetric group key k' is selected and **KeyGen(ACP)** is invoked to generate the set of new public information tuples **PI**'. Like the previous two schemes, the secrets shared with existing users are not affected by the group change.

3.3.3 Security

If an unbounded adversary can break our access tree AB-GKM scheme, a simulator can be constructed to break the modified ACV-BGKM scheme. Like the previous scheme, we only give a high-level detail of the reduction based proof.

Proof Suppose that an unbounded adversary $\mathcal{A}$ using a set of attributes α as the challenge set that does not satisfy the access tree $\mathcal{T}$ breaks our scheme in the random oracle model with advantage at most ϵ. Let the root node of $\mathcal{T}$ be r and the group key $k = q_r(0)$. Notice that since $\mathcal{A}$ does not satisfy $\mathcal{T}$ and $q_r(x)$ a t_r-*out-of*-n_r threshold scheme, which represents any type of threshold node, $\mathcal{A}$ satisfies no more than $t_r - 1$ subtrees rooted at children of r out of the n_r subtrees. By inference, it is easy to see that $\mathcal{A}$ does not satisfy at least one leaf node.

The challenger constructs modified ACV-BGKM instances for each of the attributes and gives them to $\mathcal{B}$. $\mathcal{A}$ obtains secrets for each of the attributes in α. $\mathcal{B}$ sends the public information tuples and the access tree $\mathcal{T}$ to $\mathcal{A}$. Notice that $\mathcal{A}$ can easily derive the keys for any attribute in α, but it can derive the keys for any other attribute only with an advantage of ϵ. According to the assumption, $\mathcal{A}$ does not satisfy at least one attribute required to satisfy $\mathcal{T}$. Let that attribute be $attr_x$. $\mathcal{A}$

derives k_x from PI_x corresponding to one such unsatisfied leaf node with advantage ϵ. Therefore, $\mathcal{A}$ derives the group key k with an advantage of at most ϵ.

Like the proof in Section 3.2, $\mathcal{A}$ derives the group key k, after executing the phase 1 of the security game as many times and give k to $\mathcal{B}$. Now, $\mathcal{B}$ works downwards $\mathcal{T}$ to recover the keys for nodes originally unsatisfied by $\mathcal{A}$ using Lagrange interpolation. For example, using k and $t_r - 1$, $\mathcal{B}$ obtains the key k_{t_r} for the t_r^{th} child node of r. Finally, $\mathcal{B}$ obtains the key k_x for an unsatisfied leaf node x corresponding to $attr_x$. In other words, it allows $\mathcal{B}$ to break the modified ACV-BGKM scheme to recover the key k_x from the public information tuple PI_x without the knowledge of the secret s_x. As mentioned earlier, the probability of breaking the modified ACV-BGKM scheme by applying the **KeyDer** algorithm is a negligible $1/q^N$ where q is the ℓ bit prime number and N is the maximum number of users. Therefore, it follows that ϵ must be negligible. ■

3.3.4 Performance

We now discuss the efficiency of access tree AB-GKM with respect to computational costs and required bandwidth for rekeying.

For any Usr_i in the group, deriving the shared group key requires: $\sum_{i=1}^{d} N_i$ hashing operations (evaluations of $H(\cdot)$), where $d = |\beta_i|$, N_i is the maximum number of users having attr_i, and d inner product computations $v_i \cdot ACV_i$ of two $(2N_i)$-dimensional $\mathbb{F}_q$-vectors and M Lagrange interpolations $O(Mm \log^2 m)$, where M is equal to the number of internal nodes in $\mathcal{T}$ and $m = |\mathcal{A}|$. Therefore, the overall computational complexity is $O(dn + Mm \log^2 m)$. Notice that the inner product computations are independent and can be parallelized to improve performance.

The cost of rekeying, communication and storage are comparable to those of the threshold scheme presented in Section 3.2.

3.4 Example Application

Among other applications, fine-grained access control in a group setting using broadcast encryption is an important application of the AB-GKM schemes. We illustrate the access-tree AB-GKM scheme using a healthcare scenario [20, 41]. We refer the reader to our technical report [40] for more examples. A hospital (Svr) supports fine-grained access control on electronic health records (EHRs) [42, 43] by encrypting and making the encrypted records available to hospital employees (Usrs). Typical hospital users include employees playing different roles such as receptionist, cashier, doctor, nurse, pharmacist, system administrator and non-employees such as patients. An EHR document is divided into data items including BillingInfo, ContactInfo, Medication, PhysicalExam, LabReports and so on. In accordance with regulations such as health insurance portability and accountability act (HIPAA), the hospital policies specify which users can access which data item(s). A cashier, for example, need not have access to data in EHRs except for the BillingInfo, while a doctor or a nurse need not have access to BillingInfo. These policies can be based on the content of EHRs itself. An example of such policies is that "information about a patient with cancer can only be accessed by the primary doctor of the patient". In addition, patients define their own privacy policies to protect their EHRs. For example, a patient's policy may specify that "only the doctors and nurses who support her insurance plan can view her EHR".

In order to support content-based access control, the hospital maintains some associations among users and data. Table 3.2 shows the insurance plans supported by each doctor and nurse, identified by the pseudonym "Employee ID".

The hospital runs **Setup** algorithm to initialize system parameters and issues secrets to employees by running the **SecGen** algorithm. Table 3.3 shows the content of the user attribute matrix UA that the hospital maintains. (Small numbers are used for illustrative purposes.)

Table 3.2: Insurance plans supported by doctors/nurses

EmployeeID	**Role/level**	**Insurance Plan(s)**
emp_1	doctor	MedB, ACME
emp_2	doctor	ACME
emp_3	nurse/junior	ACME
emp_4	nurse/senior	MedA
emp_5	nurse/senior	MedC
emp_6	doctor	MedA
emp_7	doctor	MedB, ACME
emp_8	nurse/senior	MedA
emp_9	nurse/senior	MedA, MedB, ACME

Table 3.3: User attribute matrix

Emp ID	**doctor**	**nurse**	**senior**	**junior**	**MedA**	**MedB**	**MedC**	**ACME**
emp_1	100	$\perp$	$\perp$	$\perp$	$\perp$	111	$\perp$	102
emp_2	120	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	$\perp$	105
emp_3	$\perp$	106	$\perp$	120	$\perp$	$\perp$	$\perp$	121
emp_4	$\perp$	103	150	$\perp$	175	$\perp$	$\perp$	$\perp$
emp_5	$\perp$	133	151	$\perp$	$\perp$	$\perp$	161	$\perp$
emp_6	129	$\perp$	$\perp$	$\perp$	141	$\perp$	$\perp$	$\perp$
emp_7	119	$\perp$	$\perp$	$\perp$	$\perp$	133	$\perp$	137
emp_8	$\perp$	143	152	$\perp$	115	$\perp$	$\perp$	$\perp$
emp_9	$\perp$	109	156	$\perp$	117	119	$\perp$	124

Now we illustrate the use of the access tree AB-GKM scheme. Consider the following policy specification on the Medication data item of the EHR. "A senior nurse supporting at least two insurance plans can access Medication of any patient". In order to implement this access control policy, we need to consider attributes role, level and insurance plan. The access control policy looks as follows:

ACP = ("role = nurse" $\wedge$ "level = senior" $\wedge$ "2-*out-of*-{MedA, MedB, MedC, ACME}")

Table 3.4: List of employees satisfying each insurance plan

Attribute	**Employee IDs**
MedA	$\mathsf{emp}_4, \mathsf{emp}_6, \mathsf{emp}_8, \mathsf{emp}_9$
MedB	$\mathsf{emp}_1, \mathsf{emp}_7, \mathsf{emp}_9$
MedC	emp_5
ACME	$\mathsf{emp}_1, \mathsf{emp}_2, \mathsf{emp}_3, \mathsf{emp}_7, \mathsf{emp}_9$

In addition to Table 3.4 containing the list of employees satisfying insurance plans, the hospital maintains the list of employees satisfying the attributes nurse and senior as shown in Table 3.5.

Table 3.5: List of employees satisfying attributes

Attribute	**Employee IDs**
nurse	$\mathsf{emp}_3, \mathsf{emp}_4, \mathsf{emp}_5, \mathsf{emp}_8, \mathsf{emp}_9$
senior	$\mathsf{emp}_4, \mathsf{emp}_5, \mathsf{emp}_8, \mathsf{emp}_9$

The above policy can be represented using an access tree with two internal nodes and six leaf nodes. The root node is an AND gate and has three children. The first and second children of the root node represent the attributes nurse and senior,

respectively, and the third child of the root node is a 2-*out-of*-4 threshold gate which has four children representing the four insurance plans.

The hospital executes the **KeyGen** algorithm to generate six PI tuples and encrypts the Medication data items with the group symmetric key k:

$$
\begin{aligned}
PI_{MedA} &= \langle ACV_{MedA}, (z_1, z_2, z_3, z_4)\rangle \\
PI_{MedB} &= \langle ACV_{MedB}, (z_5, z_6, z_7)\rangle \\
PI_{MedC} &= \langle ACV_{MedC}, (z_8)\rangle \\
PI_{ACME} &= \langle ACV_{ACME}, (z_9, z_{10}, z_{11}, z_{12}, z_{13})\rangle \\
PI_{nurse} &= \langle ACV_{nurse}, (z_{14}, z_{15}, z_{16}, z_{17}, z_{18})\rangle \\
PI_{senior} &= \langle ACV_{senior}, (z_{19}, z_{20}, z_{21}, z_{22})\rangle
\end{aligned}
$$

Expressive access control. Notice that only one employee, emp_9, can derive the group key k using **KeyDer** algorithm to decrypt Medication data items.

Collusion resistance. Notice that emp_4 supports MedA and emp_5 supports MedC and both of them are senior nurses. It may appear that these two employees can collude to derive the group key k. Since, in this particular example, the access tree AB-GKM scheme associates each user with two unique polynomials, one for the AND gate and another for the threshold gate, none of them individually satisfies the access tree and **KeyDer** results in an incorrect key.

Handling user dynamics. Assume that emp_4 starts to support the insurance plan ACME in addition to MedA. The hospital re-generates the public information by adding emp_4 to the calculation of PI_{ACME} and associating a new group key k'. Now emp_4 is able to derive k' using **KeyDer** as its attributes satisfy the access tree. Notice that the change in the user attributes does not affect the secret information each existing employees have. A similar approach is taken when one or more of these attributes are revoked from an existing employee. It should be noted that, like the

first two schemes, this scheme has the added flexibility to support changes to the access tree by requiring only changes to the public information.

3.5 Experimental Results

In this section we provide experimental results for the underlying optimized ACV-BGKM scheme used with all three AB-GKM schemes presented earlier. We compare our results with CP-ABE scheme with comparable security parameters.

The experiments were performed on a machine running GNU/Linux kernel version 2.6.32 with an Intel® Core™ 2 Duo CPU E8400 3.00GHz and 3.2 Gbytes memory. Only one processor was used for computation. Our prototype system is implemented in C/C++. We use V. Shoup's NTL library [37] version 5.4.2 for finite field arithmetic, and SHA-1 and AES-128 implementations of OpenSSL [38] version 1.0.0d for cryptographic hashing and symmetric key encryption. We use Bethencourt et. al.'s cpabe [44] library to gather experimental results for CP-ABE. The cpabe library uses PBC library [45] for pairing based cryptography.

We implemented the ACV-BGKM scheme with subset cover optimization. We utilized the complete subset algorithm introduced by Naor et al. [35] as the subset cover. All finite field arithmetic operations in ACV-BGKM scheme are performed in an 512-bit prime field. We used comparable and efficient pairing parameters for CP-ABE. The size of the base finite field is set to the 512-bit prime number
8780710799663312522437781984754049815806883199414208211028653399266475630
8802229570786251794226622214231558587695823174592777133673174813249251299
98224791
and the group order to the 160-bit number 7307508186654516213611192455715049014
05976559617.

Following the well-known security practice, we generate symmetric keys and use them for encrypting documents. Then we encrypt such encryption keys with either the ACV-BGKM generated symmetric keys or the CP-ABE generated public keys.

Table 3.6: Average time for CP-ABE algorithms

Algorithm	**Time (ms)**
Setup	34.395
Key generation	26.725
Encryption	24.453
Decryption	13.415

Therefore, in the experiments we measure the time to encrypt and decrypt the document encryption keys only. For all the ACV-BGKM experiments, we assume that 5% of users have left the group after executing the setup.

First we give experimental results for the most simplest case where a single attribute condition is considered. Then we provide, experimental results for multiple attribute conditions.

Table 3.6 shows the average time required to execute setup, key generation, encryption and decryption algorithms of CP-ABE scheme for one attribute condition.

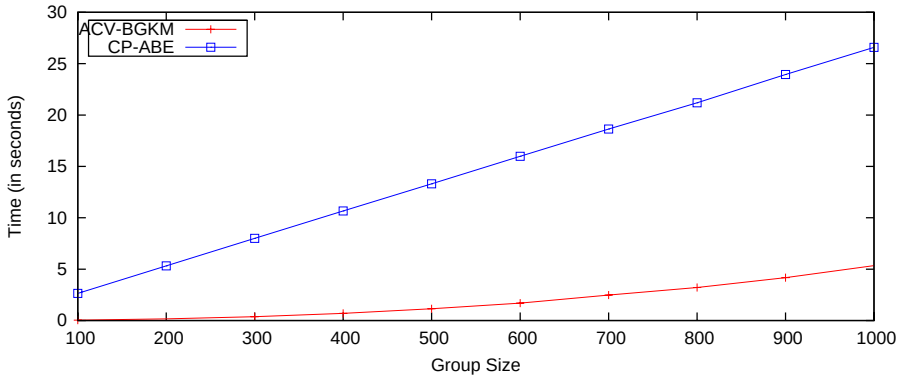

Figure 3.1.: Average key generation time for different group sizes

Figure 3.1 reports the average time required to execute the key generation algorithm of ACV-BGKM and CP-ABE with different group sizes. In both ACV-BGKM and CP-ABE the time increases linearly with the group size. However, ACV-BGKM

is much more efficient as it does not involve any expensive pairing operations. It only uses efficient hashing and binary operations over a finite field. Further, the subset cover technique applied to ACV-BGKM reduces the computational complexity of the underlying scheme. Without the subset cover optimization, ACV-BGKM has a non-linear computational complexity and becomes inefficient for large groups. We omit the comparison experimental result due to lack of space.

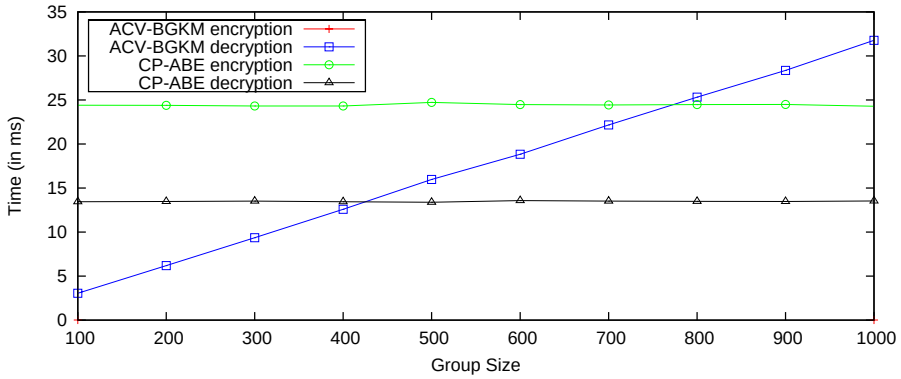

Figure 3.2.: Average encryption/decryption time for different group sizes

Figure 3.2 reports the average time required to perform encryption and decryption in ACV-BGKM and CP-ABE schemes for one attribute condition with different group sizes. The decryption time of ACV-BGKM is taken as the time to derive the key as well as to decrypt the encryption key. The encryption and decryption times of CP-ABE remain constant whereas the decryption time of ACV-BGKM increases linearly with the group size. As the group size increases, the key derivation algorithm of ACV-BGKM requires to spend more time to build larger KEVs. The encryption time of ACV-BGKM is negligible and remains constant as it involves an efficient symmetric encryption only. The average encryption time of ACV-BGKM is 8.8 microseconds (as these times are very small, the line plotting them is very close to zero in the graph in Figure 3.2 and thus overlaps with the x-axis). It should be noted that if one caches the KEVs, the decryption time of ACV-BGKM also becomes negligible as it involves only modular multiplications.

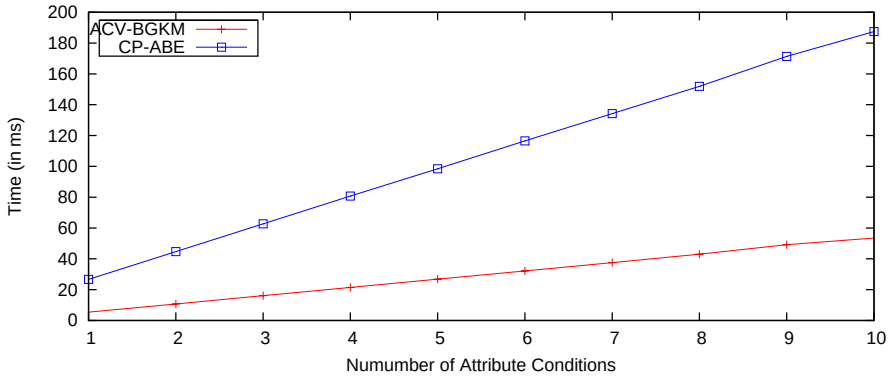

Figure 3.3.: Average key generation time for varying attribute counts

Figure 3.3 reports the average time required to execute the key generation algorithm with varying number of attribute conditions with the group size set to 1000. The time of both techniques increases linearly with the number of attribute conditions. However, similar to Figure 3.1, the ACV-BGKM key generation is much more efficient than the CP-ABE key generation.

As can be seen from the experiments, our constructs are more efficient in handling scenarios where the key generation algorithm has to be executed frequently due to changes in user dynamics.

4 PRIVACY PRESERVING PULL BASED SYSTEMS: SINGLE LAYER APPROACH

We apply the GKM schemes constructed in Chapter 3 to build privacy preserving pull based systems. Consistent with the current technological trends, we refer to the third party server as the Cloud.

An approach to support fine-grained selective attribute-based access control before uploading the data to the Cloud is to encrypt each data item to which the same ACP (or set of ACPs) applies with the same key. One approach to deliver the correct keys to the users based on the policies they satisfy is to use a hybrid solution where the keys are encrypted using a public key cryptosystem such as attribute based encryption (ABE) and/or proxy re-encryption (PRE). However, such an approach has several weaknesses: it cannot efficiently handle adding/revoking users or identity attributes, and policy changes; it requires to keep multiple encrypted copies of the same key; it incurs high computational cost. Therefore, a different approach is required.

It is worth noting that a simplistic group key management (GKM) scheme in which the Owner directly delivers the symmetric keys to corresponding users has some major drawbacks with respect to user privacy and key management. On one hand, user private information encoded in the user identity attributes is not protected in the simplistic approach. On the other hand, such a simplistic key management scheme does not scale well as the number of users becomes large and when multiple keys need to be distributed to multiple users. The goal of this paper is to develop an approach which does not have these shortcomings.

We observe that, without utilizing public key cryptography and by allowing users to dynamically derive the symmetric keys at the time of decryption, one can address the above weaknesses. Based on this idea, in Chapter 2, we first formalized a new GKM scheme called broadcast GKM (BGKM) and then gave a secure construction

of BGKM scheme and formally prove its security. The idea is to give secrets to users based on the identity attributes they have and later allow them to derive actual symmetric keys based on their secrets and some public information. A key advantage of the BGKM scheme is that adding users/revoking users or updating access control policies can be performed efficiently and only requires updating the public information. As shown in Chapter 2, our BGKM scheme satisfies the requirements of *minimal trust*, *key indistinguishability*, *key independence*, *forward secrecy*, *backward secrecy* and *collusion resistance* as described in [15] with minimal computational, space and communication cost.

In Chapter 3, using the ACV-BGKM scheme as a key building block, we constructed a more expressive GKM scheme called AB-GKM. Using our Inline AB-GKM scheme, we develop an attribute-based access control mechanism whereby a user is able to decrypt the data if and only if its identity attributes satisfy the Owner's policies, whereas the Owner and the Cloud learn nothing about user's identity attributes. The mechanism is fine-grained in that different policies can be associated with different data items. A user can derive only the encryption keys associated with the data items that the user is entitled to access.

The rest of the chapter is organized as follows. Section 4.1 provides an overview of our overall SLE approach. Section 4.2 shows how to preserve the privacy of identity attributes from both the data owner and the third-party. Section 4.3 provides detailed description of our scheme. Section 4.4 proposes utilizing incremental unforgeable encryption to improve the efficiency at the Owner when the re-encryption operation is performed. Section 4.6 presents experimental results on the OCBE protocols and key management.

4.1 Overview of the SLE Approach

As shown in Figure 4.1, our scheme for policy based content sharing in the cloud involves four main entities: the *Data Owner* (Owner), the *Users* (Usrs) , the *Iden-*

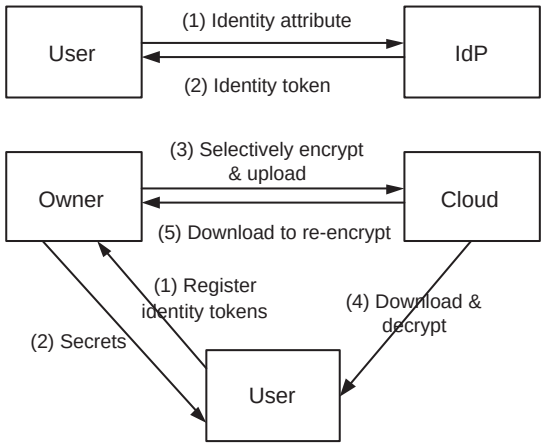

Figure 4.1.: Overall system architecture

tity Providers (IdPs), and the *Cloud Storage Service* (Cloud). The interactions are numbered in the figure. Our approach is based on three main phases: *identity token issuance*, *identity token registration*, and *data management*.

1) Identity token issuance

IdPs issue *identity tokens* for certified identity attributes to Usrs. An identity token is a Usr's identity in a specified electronic format in which the involved identity attribute value is represented by a semantically secure cryptographic *commitment*.[1] We use the Pedersen commitment scheme and it is described in Section 4.2.2. Identity tokens are used by Usrs during the registration phase.

2) Identity token registration

In order to be able to decrypt the data that will be downloaded from the Cloud, Usrs have to register at the Owner. During the registration, each Usr presents its identity tokens and receives from the Owner a set of secrets for each identity attribute based on the SecGen algorithm of the AB-GKM scheme. These secrets are later used by Usrs to derive the keys to decrypt the data items for which they satisfy the ACP

[1]A cryptographic commitment allows a user to commit to a value while keeping it hidden and preserving the user's ability to reveal the committed value later.

using the KeyDer algorithm of the AB-GKM scheme. The Owner delivers the secrets to the Usrs using a privacy-preserving approach based on the OCBE protocols [46] with the Usrs. The OCBE protocols ensure that a Usr can obtain secrets if and only if the Usr's committed identity attribute value (within Usr's identity token) satisfies the matching condition in the Owner's ACP, while the Owner learns nothing about the identity attribute value. Note that not only the Owner does not learn anything about the actual value of Usrs' identity attributes but it also does not learn which policy conditions are verified by which Usrs, thus the Owner cannot infer the values of Usrs' identity attributes. Thus Usrs' privacy is preserved in our scheme. We give more details about the OCBE protocols in Section 4.2.3.

3) Data Management

The Owner groups the ACPs into *policy configurations* (Pcs). The data are divided into data items based on the Pcs. The Owner generates the keys based on the ACPs in each Pc using the KeyGen algorithm of the AB-GKM scheme and selectively encrypts the data. These encrypted data are then uploaded to the Cloud. Usrs download encrypted data from the Cloud. The KeyDer algorithm of the AB-GKM scheme allows Usrs to derive the key K for a given Pc using their secrets in an efficient and secure manner. With this scheme, our approach efficiently handles new users and revocations to provide forward and backward secrecy. The system design also ensures that ACPs can be flexibly updated and enforced by the Owner without changing any information given to Usrs.

4.2 Preserving the Privacy of Identity Attributes

We observe that by preserving the privacy of the SecGen algorithm of the AB-GKM scheme we can preserve the privacy of the whole AB-GKM scheme. We utilize cryptographic techniques to protect the privacy of the identity attributes of the users from the Svr while executing the SecGen algorithm. Our technique makes sure that Usrs receive secrets only for valid identity attributes while the Svr does not learn

the actual identity attribute values. We now give you an overview of the two cryptographic constructs, Pedersen commitments and oblivious commitment based envelope protocols, that we use in this regard. Further, we introduce the notion of configurable privacy for the identity attributes.

4.2.1 Discrete Logarithm Problem and Computational Diffie-Hellman Problem

Definition 4.2.1 *Let G be a (multiplicatively written) cyclic group of order q and let g be a generator of G. The map $\varphi : \mathbb{Z} \to G, \varphi(n) = g^n$ is a group homomorphism with kernel $\mathbb{Z}_q$. The problem of computing the inverse map of φ is called the* discrete logarithm problem (DLP) to the base of g.

Definition 4.2.2 *For a cyclic group G (written multiplicatively) of order q, with a generator $g \in G$, the* Computational Diffie-Hellman problem (CDH) *is the following problem: Given g^a and g^b for randomly-chosen secret $a, b \in \{0, \ldots, q-1\}$, compute g^{ab}.*

4.2.2 Pedersen Commitment

First introduced in [47], the Pedersen Commitment scheme is an unconditionally hiding and computationally binding commitment scheme which is based on the intractability of the discrete logarithm problem. We describe how it works as follows.

Setup

A trusted third party T chooses a finite cyclic group G of large prime order p so that the computational Diffie-Hellman problem is hard in G. Write the group operation in G as multiplication. T chooses two generators g and h of G such that it is hard to find the discrete logarithm of h with respect to g, i.e., an integer α such that $h = g^\alpha$. Note that T may or may not know the number α. T publishes (G, p, g, h) as the system's parameters.

Commit

The domain of committed values is the finite field $\mathbb{F}_p$ of p elements, which can be implemented as the set of integers $\mathbb{F}_p = \{0, 1, \ldots, p-1\}$. For a party U to commit a value $x \in \mathbb{F}_p$, U chooses $r \in \mathbb{F}_p$ at random, and computes the commitment $c = g^x h^r \in G$.

Open

U shows the values x and r to open a commitment c. The verifier checks whether $c = g^x h^r$.

4.2.3 OCBE Protocols

The Oblivious Commitment-Based Envelope (OCBE) protocols, proposed by Li and Li [46], provide the capability of delivering information to qualified users in an oblivious way. There are three communications parties involved in OCBE protocols: a receiver R, a sender S, and a trusted third party T. The OCBE protocols make sure that the receiver R can decrypt a message sent by S if and only if R's committed value satisfies a condition given by a predicate in S's access control policy, while S learns nothing about the committed value. Note that S does not even learn whether R is able to correctly decrypt the message or not. The supported predicates by OCBE are comparison predicates $>, \geq, <, \leq, =$ and $=$.

The OCBE protocols are built with several cryptographic primitives:

1. The Pedersen commitment scheme.
2. A semantically secure symmetric-key encryption algorithm $\mathcal{E}$, for example, AES, with key length k-bits. Let $\mathcal{E}_{\mathsf{Key}}[M]$ denote the encrypted message M under the encryption algorithm $\mathcal{E}$ with symmetric encryption key Key.
3. A cryptographic hash function $H(\cdot)$. When we write $H(\alpha)$ for an input α in a certain set, we adopt the convention that there is a canonical encoding which

encodes α as a bit string, i.e., an element in $\{0,1\}^*$, without explicitly specifying the encoding.

Given the notations as above, we summarize the OCBE protocol for $=$ (EQ-OCBE) and $\geq$ (GE-OCBE) predicates as follows. The OCBE protocols for other predicates can be derived and described in a similar fashion. The protocols' description is tailored to our work, and is stated in a slightly different way than in [46].

EQ-OCBE Protocol

Parameter generation

T runs a Pedersen commitment setup protocol to generate system parameters $\mathsf{Param} = \langle G, g, h\rangle$. T outputs the order of G, p, and $\mathcal{P} = \{\mathsf{EQ}_{x_0} : x_0 \in \mathbb{F}_p\}$, where

$$\mathsf{EQ}_{a_0} : \mathbb{F}_p \rightarrow \{\mathsf{true}, \mathsf{false}\}$$

is an equality predicate such that $\mathsf{EQ}_{x_0}(x)$ is true if and only if $x = x_0$.

Commitment

T first chooses an element $x \in \mathbb{F}_p$ for R to commit. T then randomly chooses $r \in \mathbb{F}_p$, and computes the Pedersen commitment $c = g^x h^r$. T sends x, r, c to R, and sends c to S.

Alternatively, in an offline version, T digitally signs c and sends x, r, c together with the signature of c to R. Then the validity of the commitment c can be ensured by verifying T's signature. In this way, after S obtains T's public key for signature verification, no further communication is needed between T and S.

Interaction

- R makes a data request to S.
- Based on this request, S sends an equality predicate $\mathsf{EQ}_{x_0} \in \mathcal{P}$.
- Upon receiving this predicate, R sends S a Pedersen commitment $c = g^x h^r$.

- S picks $y \in \mathbb{F}_p^*$ at random, computes $\sigma = (cg^{-x_0})^y$, and sends R a pair $\langle \eta = h^y, C = \mathcal{E}_{H(\sigma)}[M] \rangle$, where M is a message containing the requested data.

Open

Upon receiving $\langle \eta, C \rangle$ from S, R computes $\sigma' = \eta^r$, and decrypts C using $H(\sigma')$.

The **GE-OCBE** Protocol works in a bit-by-bit fashion, for attribute values of at most ℓ bits long, where ℓ is a system parameter which specifies an upper bound for the bit length of attribute values such that $2^\ell < p/2$. The GE-OCBE protocol is more complex in terms of description and computation compared to EQ-OCBE (=). It works as follows.

GE-OCBE Protocol

Parameter generation

T runs a Pedersen commitment setup protocol to generate system parameters $\mathsf{Param} = \langle G, g, h \rangle$, and outputs the order of G, p. In addition, T chooses another parameter ℓ, which specifies an upper bound for the length of attribute values, such that $2^\ell < p/2$. T outputs $\mathcal{V} = \{0, 1, \ldots, 2^\ell - 1\} \subset \mathbb{F}_p$, and $\mathcal{P} = \{\mathsf{GE}_{x_0} : x_0 \in \mathcal{V}\}$, where

$$GE_{x_0} : \mathcal{V} \rightarrow \{\mathsf{true}, \mathsf{false}\}$$

is a predicate such that $\mathsf{GE}_{x_0}(x)$ is true if and only if $x \geq x_0$.

Commitment

T chooses an integer $x \in \mathcal{V}$ for R to commit. T then randomly chooses $r \in \mathbb{F}_p$, and computes the Pedersen commitment $c = g^x h^r$. T sends x, r, c to R, and sends c to S.

Similarly, an offline alternative also works here.

Interaction

- R makes a data request to S.
- Based on the request, S sends to R a predicate $\mathsf{GE}_{x_0} \in \mathcal{P}$.

- Upon receiving this predicate, R sends to S a Pedersen commitment $c = g^x h^r$.
- Let $d = (x - x_0) \pmod p$. R picks $r_1, \ldots, r_{\ell-1} \in \mathbb{F}_p$, and sets $r_0 = r - \sum_{i=1}^{\ell-1} 2^i r_i$. If $\mathsf{GE}_{x_0}(x)$ is true, let $d_{\ell-1} \ldots d_1 d_0$ be d's binary representation, with d_0 the lowest bit. Otherwise if GE_{x_0} is false, R randomly chooses $d_{\ell-1}, \ldots, d_1 \in \{0, 1\}$, and sets $d_0 = d - \sum_{i=1}^{\ell-1} 2^i d_i \pmod p$. R computes ℓ commitments $c_i = g^{d_i} h^{r_i}$ for $0 \leq i \leq \ell - 1$, and sends all of them to S.
- S checks that $cg^{-x_0} = \prod_{i=0}^{\ell-1} (c_i)^{2^i}$. S randomly chooses ℓ bit strings $k_0, \ldots, k_{\ell-1}$, and sets $k = H(k_0 \parallel \ldots \parallel k_{\ell-1})$. S picks $y \in \mathbb{F}_p^*$, and computes $\eta = h^y, C = \mathcal{E}_k[M]$, where M is the message containing requested data. For each $0 \leq i \leq \ell-1$ and $j = 0, 1$, S computes $\sigma_i^j = (c_i g^{-j})^y, C_i^j = H(\sigma_i^j) \oplus k_i$. S sends to R the tuple

$$\langle \eta, C_0^0, C_0^1, \ldots, C_{\ell-1}^0, C_{\ell-1}^1, C \rangle.$$

Open

After R receives the tuple $\langle \eta, C_0^0, C_0^1, \ldots, C_{\ell-1}^0, C_{\ell-1}^1, C \rangle$ from S as above, R computes $\sigma_i' = \eta^{r_i}$, and $k_i' = H(\sigma_i') \oplus C_i^{d_i}$, for $0 \leq i \leq \ell - 1$. R then computes $k' = H(k_0' \parallel \ldots \parallel k_{\ell-1}')$, and decrypts C using key k'.

EQ-OCBE protocol is simpler and more efficient compared GE-OCBE protocol. The OCBE protocol for the $\leq$ predicates (LE-OCBE) can be constructed in a similar way as GE-OCBE. Other OCBE protocols (for $=, <, >$ predicates) can be built on EQ-OCBE, GE-OCBE and LE-OCBE.

All these OCBE protocols guarantee that the receiver R can decrypt the message sent by S if and only if the corresponding predicate is evaluated as true at R's committed value, and that S does not learn

4.2.4 Configurable Privacy

In order to assure maximum privacy, Usr should register its identity token for *all* attribute conditions whose attribute names match the id-tag field in the identity token. While providing maximum privacy for Usr, it also inevitably increases the

number of OCBE protocol executions and the complexity of the AB-GKM algorithms in almost all cases. However, in an application scenario where it is not crucial for a Usr to achieve maximum privacy for certain identity attributes, Usrs are allowed to register as few as possible attribute conditions for an id-tag, while at the same time feel comfortable about the level of guaranteed privacy. In this way, the complexity of the AB-GKM algorithms can be effectively reduced. We introduce a notion similar to the idea of k-anonymity [48]. The following formula (4.1) shows an example of computing privacy level for an id-tag.

Let privacy be measured by a number from 0 to 1, where 0 means "no privacy" and 1 maximum privacy. Let $M \geq 2$ be the total number of attribute conditions which apply to an id-tag in the system. Suppose all attribute conditions corresponding to one id-tag has the same level of privacy. Let m be the number of attribute conditions a Usr registers for an identity token that it holds. Suppose a Usr holding an identity token always registers for the attribute condition which this identity token satisfies. Then the level of privacy for this registered identity token of Usr can be calculated as

Formula 1 (Privacy formula)

$$\mathcal{P} = \frac{m-1}{M-1}. \tag{4.1}$$

The above formula can be easily verified: for example, if there are overall $M = 2$ attribute conditions "role = doc" and "role = nur" for id-tag = role, then registering for $m = 1$ attribute condition reveals the attribute value, i.e., $\mathcal{P} = 0$, and registering for both ($m = 2$) attribute conditions gives maximum privacy $\mathcal{P} = 1$. Usrs may use such a quantitative measure the level of privacy they have and the system may use the same measure to impose a minimum privacy requirement, for example, to maintain organizational privacy policies.

4.3 Single Layer Encryption Approach

Section 4.1, our scheme has three phases: identity token issuance, identity token registration and data management. We did not consider the technical details and

privacy in Section 4.1. In this section we make our scheme privacy preserving using the techniques introduced in Section 4.2. We explain our approach using the AB-GKM scheme with the subset cover optimization as a key building block.

4.3.1 Identity Token Issuance

The IdP runs a Pedersen commitment setup algorithm to generate system parameters $\mathsf{Param} = \langle G, g, h \rangle$. The IdP publishes Param as well as the order p of the finite group G. The IdP also publishes its public key for the digital signature algorithm it is using. Such parameters are used by the IdP to issue *identity tokens* to Usrs. We assume that the IdP first checks the valid of identity attributes Usrs hold [2]. Usrs present to the IdP their identity attributes to receive *identity tokens* as follows. For each identity attribute shown by a Usr, the IdP encodes the identity attribute value as $x \in \mathbb{F}_p$ in a standard way, and issues the Usr an identity token. An identity token is a tuple

$$\mathcal{IT} = (\mathsf{nym}, \mathsf{id\text{-}tag}, c, \sigma),$$

where nym is a pseudonym for uniquely identifying the Usr in the system, id-tag is the tag of the identity attribute under consideration, $c = g^x h^r$ is a Pedersen commitment for the value x, and σ is the IdP's digital signature for nym, id-tag and c. The IdP passes values x and r to the Usr for the Usr's private use. We require that all identity tokens of the same Usr have the same nym,[3] so that the Usr and its identity tokens can be uniquely matched with a nym. Once the identity tokens are issued, they are used by Usrs for proving the satisfiability of the Pub's ACPs; Usrs keep their identity attribute values hidden, and never disclose them in clear during the interactions with other parties.

[2]The IdP can verify the validity of Usr's identity either in a traditional way, e.g., through a on-the-spot registration, or digitally over computer networks. We will not dive into the details of identity validity check in this thesis.

[3]In practice, this can be achieved by requesting the Usr to present a strong identifier that correlates with the identity being registered. Again, we will not discuss this process in this thesis.

Example 1

Suppose a Usr Bob presents his driver's license to IdP to receive an identity token for his age. IdP assigns Bob a pseudonym pn-1492. IdP deduces from the birth date on Bob's driver's license that Bob's age is $x = 28$. The IdP randomly chooses a value $r = 9270$, and computes a Pedersen commitment $c = g^x h^r$. The IdP then digitally signs the message containing Bob's pseudonym, a tag for "age" and the commitment c. The identity token Bob receives from the IdP may look like this:
$\mathcal{IT} = (\mathsf{pn\text{-}1492}, \mathsf{age}, 6267292101, 949148425702313975)$.

4.3.2 Identity Token Registration

We assume that the Owner defines a set of ACPs denoted as $\mathcal{ACPB}$ that specifies which data items Usrs are authorized to access. ACPs are formally defined as follows.

Definition 4.3.1 *(**Attribute Condition**).*
An attribute condition cond *is an expression of the form: "*name_A $\mathtt{op}$ l*", where* name_A *is the name of an identity attribute* A*,* $\mathtt{op}$ *is a comparison operator such as* $=, <, >, \leq, \geq, =$*, and* l *is a value that can be assumed by attribute* A*.*

Definition 4.3.2 *(**Access control policy**).*
An access control policy (ACP) is a tuple $(s, o, \mathcal{D})$ *where:* o *denotes a set of data items* $\{\mathcal{D}_1, \ldots, \mathcal{D}_t\}$ *of data* $\mathcal{D}$*; and* s *is a Boolean formula of attribute conditions* $\mathsf{cond}_1, \ldots, \mathsf{cond}_n$ *that must be satisfied by a* Usr *to have access to* o*.* [4]

Different ACPs can apply to the same data items because such data items may have to be accessed by different categories of Usrs. We denote the set of ACPs that apply to a data item as *policy configuration*.

Definition 4.3.3 *(**Policy configuration**).*
A policy configuration (Pc) for a data item $\mathcal{D}_1$ *of data* $\mathcal{D}$ *is a set of policies* $\{\mathsf{ACP}_1, \ldots, \mathsf{ACP}_k\}$ *where* $\mathsf{ACP}_i, i = 1, \ldots, k$ *is an ACP* $(s, o, \mathcal{D})$ *such that* $\mathcal{D}_1 \in o$*.*

[4] In what follow we use the dot notation to denote the different components of an ACP.

Example 2

The ACP ("level ≥ 58" $\wedge$ "role = nurse", {physical exam, treatment plan}, "EHR.xml") states that a Usr of level no lower than 58 and holding a nurse position has access to the data items "physical exam" and "treatment plan" of document EHR.xml.

There can be multiple data items in $\mathcal{D}$ which have the same Pc. For each Pc of $\mathcal{D}$, the Owner randomly chooses a key K for a symmetric key encryption algorithm (e.g, AES), and uses K to encrypt all data items associated with this policy configuration. Therefore, if a Usr satisfies $\mathsf{ACP}_1, \ldots, \mathsf{ACP}_m$, Owner must make sure that the Usr can derive all the symmetric keys to decrypt those data items to which a policy configuration containing at least one $\mathsf{ACP}_i (i = 1, \ldots, m)$ applies.

As in our AB-GKM based scheme the actual symmetric keys are not delivered along with the encrypted data, a Usr has to register its identity tokens at the Owner in order to derive the symmetric encryption key from the PubInfo stored at the Cloud. The SecGen algorithm of the AB-GKM scheme and the OCBE techniques are used to register user identity tokens in a privacy preserving manner. During the registration, a Usr receives a set of secrets, based on the identity attribute names corresponding to the attribute names in the identity tokens. Note that secrets are generated by the Owner only based on the names of identity attributes and not on their values. Therefore, a Usr may receive an encrypted set of secrets corresponding to a condition which has a value that the Usr' identity attribute does not satisfy. However, in this case, the Usr will not be able to extract the secrets from the message delivering it as shown in Section 4.2.3. Proper secrets are later used by a Usr to compute symmetric decryption keys for particular data items of the encrypted data, as discussed in the data management phase. The delivery of secrets are performed in such a way that the Usr can correctly receive secrets if and only if the Usr has an identity token whose committed identity attribute value satisfies an attribute condition in Owner's ACP, while the Owner does not learn any information about the Usr's identity attribute value and does not learn whether Usr has been able to obtain the secret.

To enable Usrs registration, the Owner first chooses the OCBE parameters: an ℓ'-bit prime number q, a cryptographic hash function $H(\cdot)$ whose output bit length is no shorter than ℓ', and a semantically secure symmetric-key encryption algorithm with key length ℓ' bits. The Owner publishes these parameters. The Owner also constructs a subset cover tree with n leaf nodes corresponding to each Usr for each distinct attribute condition in ACPs. Let SC_j be the subset cover for the attribute condition cond_j. Then for an ACP in $\mathcal{ACPB}$ that a subscriber Usr_i under pseudonym nym_i wants to satisfy, it selects and registers an identity token $\mathcal{IT} = (\mathsf{nym}_i, \mathsf{id\text{-}tag}, c, \sigma)$ with respect to each attribute condition cond_j in ACP. Note that Usr_i does not register only for the attribute condition which the Usr_i's identity token satisfies; to assure privacy, Usr_i registers its identity token for more attribute conditions whose identity attribute name matches the id-tag contained in the identity token. In this way, the Owner cannot infer from Usr_i's registration which condition Usr_i is actually interested in. Such measures greatly reduce the leaking of identity attributes due to insider threats.

The Owner checks if id-tag matches the name of the identity attribute in cond_j, and verifies the IdP's signature σ using the IdP's public key. If either of the above steps fails, the Owner aborts the interaction. Otherwise, the Owner selects the corresponding secrets from the subset cover SC_j for Usr_i. The Owner then starts an OCBE session as a sender (S) to obliviously transfer these secrets to Usr_i who acts as a receiver (R). The Owner maintains a matrix $\mathcal{T}$ to store if secrets are delivered to each Usr_i for each cond_j. Upon the completion of the OCBE session the Owner performs the following actions:

- If nym_i does not exist in the matrix, it first creates a row for it.
- It sets $r_{i,j}$ cell of $\mathcal{T}$ with respect to nym_i and cond_j.

We remark that all secrets are independent, so the above secret delivery process can be executed in parallel. Matrix $\mathcal{T}$ is used by the Owner to execute the KeyGen algorithm of the AB-GKM scheme.

Example 3

Matrix 4.1 shows an example of matrix $\mathcal{T}$. A Usr under pseudonym pn-0012 who has an identity token with respect to identity tag role registers for all attribute conditions ("role = doc" and "role = nur" are shown in Table 4.1) involving identity attribute role. This Usr does not register for attribute conditions "level ≥ 59", "YoS ≥ 5" [5] and "YoS < 5", either because it does not hold an identity token with identity tag level or YoS, thus cannot register, or because it chooses not to register as it only needs to access data items whose associated ACP does not require conditions for these attributes. A drawback of registering only for the conditions required is that it may allow an attacker to infer certain attributes about the Usr with high confidence. To protect against such attacks the Usr may choose to register for more than one condition as explained earlier. Note that the Usr under pn-0829 registers for both conditions YoS ≥ 5 and YoS < 5, which are mutually exclusive and thus both cannot be satisfied by any Usr. The registration for both conditions is crucial for privacy in that it prevents the Pub from inferring from the Usr's registration behavior which condition the Usr is actually interested in. A Usr under pn-1492 registers for all five attribute conditions.

Table 4.1: A table of secrets maintained by the Pub

nym	level ≥ 59	YoS ≥ 5	YoS < 5	role = doc	role = nur	...
pn-0012	$\perp$	$\perp$	$\perp$	1	1	...
pn-0829	1	1	1	$\perp$	$\perp$	...
pn-1492	1	1	1	1	1	...
...	...	...	...	...	...	...

[5] YoS means "years of service".

4.3.3 Data Management

Recall that the Owner encrypts all data items with the same Pc applicable with the same symmetric key. Therefore, the Owner execute the KeyGen algorithm of the AB-GKM for each Pc. For a given Pc, the Owner first identifies the secrets to be considered as follows.

- The Owner first converts each ACP into DNF (Disjunctive Normal Form). For each unique conjunctive term, it executes the remaining steps.
- Let i^{th} conjunctive term be $\bigwedge_{j=1}^{\phi_i} \mathsf{cond}_j$, where the term has ϕ_i conditions. The Owner iterates through the secrets matrix $\mathcal{T}$, and finds the set of users who satisfy all the conditions in each conjunctive term.
- At the end of the previous step, the Owner has the list of Usrs who satisfy the Pc, their association with the subset covers SC_i for each applicable cond_i. The Owner identifies the covers in each SC_i and the secrets corresponding the covers. The Owner aggregates by concatenating secrets in the order of the conditions in the conjunctive terms to produce a single secret for each user satisfying the conjunctive terms. For example, if the conjunctive term is $\mathsf{cond}_1 \wedge \mathsf{cond}_3$ and Usr_5 satisfies the term, the Owner obtains the cover secrets s_1 and s_3 from SC_1 for Usr_5 and SC_3 for Usr_5 respectively. The aggregated secret is $s_1||s_3$.

The set of aggregated secrets from the above algorithm is used as the input to the KeyGen algorithm which produces the public information PubInfo and the symmetric group key k. The Owner creates an index of the public information tuples and associate with the encrypted data, and uploads them to the Cloud.

If a Usr with nym_i wants to view the data item $\mathcal{D}_1$, it first downloads the encrypted data item along with the PubInfo. It then picks an ACP_k that it satisfies and derive the key using the KeyDer algorithm.

Now we look at how to handle system dynamics such as adding/revoking credentials and ACP updates.

When a new user Usr registers at the Owner, the Owner delivers corresponding secrets to Usr, and updates the matrix $\mathcal{T}$. The Owner then performs a rekey process for all involved data items (or equivalently, policy configurations) using the Update algorithm. When Owner uploads new data, it also uploads the updated PubInfo index.

During credential revocations, the conditions under which a Usr needs to be revoked is out of the scope of this paper. We assume that the Owner will be notified when a Usr with a pseudonym nym_i is revoked from those who may satisfy cond_j. In this case, the Owner simply reset the value $r_{i,j}$ from matrix $\mathcal{T}$, and performs a rekey process for all involved data items. Allowing particular secrets to be deleted from $\mathcal{T}$ enables a fine-tuned user management.

A Usr's credentials may have to be updated over time for various reasons such as promotions, change of responsibilities, etc. In this case, the Usr with a pseudonym nym_i submits updated credential $cond_j$ to the Owner. The Owner simply resets the old $r_{i,j}$ entry and set a new entry in the matrix $\mathcal{T}$, and performs a rekey process only for the data items involved.

When a Usr with a pseudonym nym_i needs to be removed, the Owner removes the row corresponding to nym_i from the matrix $\mathcal{T}$, and performs a rekey process only for the data items involved.

Note that in all cases of new subscription, credential revocation, credential update and subscription revocation, the rekey process does not introduce any cost to Usrs in that except for those whose identity attributes are added, updated or revoked, no Usr needs to directly communicate with the Owner to update secrets–new encryption/decryption keys can be derived by using the original secrets and updated public values stored at the Cloud. The ability to derive the secret encryption/decryption keys using public values is a key point to achieve transparency in subscription handling. Most of the existing GKM scheme fails to achieve this objective.

4.4 Improving Efficiency of Re-Encryption

In the current SLE scheme, the Owner has to download full encrypted data to perform re-encryption whenever group dynamics changes. In order to improve the efficiency of the re-encryption operation, in this section, we propose to utilize incremental unforgeable encryption [49, 50] technique. It requires only re-encrypt only the modified blocks of data instead of all the blocks. We give an overview of the technique below and later provide experimental results to show that it does improve the efficiency of the overall system where frequent re-encryptions of data items are performed.

The main motivation for incremental cryptography [49] is to devise cryptographic algorithms whose output can be updated very efficiently when the underlying input changes. Incremental cryptography has been applied to hashing, signing, message authentication, and encryption. Since in our work we utilize existing incremental encryption algorithms [50] only, we limit our discussion to incremental encryption.

We view a message M as a set of blocks $m_1, m_2, \cdots, m_n$, where the block size b is decided by a security parameter ℓ. Our system should be able to perform the following modifications operations:

- Insert operation: (insert, i, m) inserts the message block m between blocks i^{th} and $(i+1)^{th}$.
- Delete operation: (delete, i) deletes the i^{th} message block.
- Replace operation: (replace, i, m) replaces the i^{th} message block with the message block m.

Definition 4.4.1 (Modification Space) *The modification space, denoted by $\mathcal{U}$, is defined as the set of all possible modification operations that can be performed on any block of a message.*

Definition 4.4.2 (Incremental Encryption) *An incremental (private-key) encryption scheme $\prod$ defined over modification space $\mathcal{U}$ is a symmetric key block cipher*

scheme that consists of the following four algorithms: ***KeyGen****,* ***Enc****,* ***Dec*** *and* ***IncEnc****. The first three algorithms are defined as in traditional block cipher schemes.*

We give an overview of the algorithms below.

KeyGen(ℓ):

The key generation algorithm is a probabilistic poly(ℓ)-time algorithm that takes as input security parameter ℓ and generates a random symmetric key k. The security parameter also fixes a block size b.

Enc(k, M):

The encryption algorithm is a probabilistic poly($\ell, |M|$)-time algorithm that takes as input the symmetric key k and the plaintext message $M \in (\{0,1\}^b)^+$, and produces the ciphertext C.

Dec(k, C):

The decryption algorithm is a deterministic poly($\ell, |C|$)-time algorithm that takes as input the symmetric key k and the ciphertext C, and produces either the plaintext message M or a special symbol $\perp$ to indicate that the ciphertext C is invalid.

IncEnc(k, U, C):

The incremental encryption algorithm is a probabilistic poly($\ell, |C|, |M|$)-time algorithm that takes as input the symmetric key k, the modification operation $U \in \mathcal{U}$, the previous ciphertext C corresponding to M, and produces the modified ciphertext C' which is the encryption of the plaintext M with the modification operation U applied.

Security requirements for the incremental encryption scheme are as follows:

- Indistinguishability: The encryption algorithm should be semantically secure.
- Unforgeability (integrity): A malicious adversary who views a sequence of encryptions and incremental update operations should be unable to generate any new ciphertext which decrypts to a valid plaintext.

- Obliviousness: The ciphertext should not reveal information about the revision history of the underlying plaintext.

A practical incremental encryption scheme should at least satisfy the indistinguishability and obliviousness requirements. We call such scheme *confidentiality only* scheme. If data integrity guarantee is required, the incremental encryption scheme should satisfy the above three security requirements. We call such scheme *confidentiality and integrity* scheme.

An incremental encryption scheme $\prod$ is called *ideal* if the running time of its incremental encryption algorithm is independent of $|M|$ and $|C|$ and depends on the type of modification only. In practice, when also data integrity must be verified, it is not possible to construct an ideal incremental encryption scheme. However, if the incremental encryption scheme can run in time sublinear to $|M|$, it is still better than the conventional encryption schemes which requires time $O(|M|)$ to compute the ciphertext from scratch. With such incremental schemes, when large messages change frequently, considerable efficiency improvements are possible.

Algorithm 1 rECB mode

1: Break the message M into b-bit blocks $m_1, m_2, \cdots, m_n$
2: Select random value $r_0 \leftarrow \{0,1\}^b$
3: Enc(k, r_0)
4: **for** Each block m_i, $i = 1$ to n **do**
5: $\quad r_i \leftarrow \{0,1\}^b$
6: $\quad c_i = (\text{Enc}(k,\ m_i \oplus r_i),\ (k,\ r_i \oplus r_0))$
7: **end for**
8: Return $c_1, c_2, \cdots, c_n$

In our work, we implement two incremental encryption schemes for confidentiality only and for both confidentiality and integrity. We use randomized ECB (rECB) and RPC modes with a block cipher [50] for confidentiality only, and confidentiality and

integrity schemes respectively. We give a high-level description of these two modes of encryption below.

Randomized ECB (rECB) Mode

Recall that rECB mode provides confidentiality only. Algorithm 1 describes encrypting with this mode.

Decryption is performed by computing $\text{Dec}(k, c_i)$, $i = 1, 2, \cdots, n$. It is easy to see that it supports replace, delete and insert operations. Incremental update operations result in only small changes to the ciphertext as each block is encrypted independently.

RPC Mode

RPC mode provides both confidentiality and integrity. Algorithm 2 describes encrypting with this mode.

Algorithm 2 RPC mode

1: Break the message M into $b - 2r$-bit blocks $m_1, m_2, \cdots, m_n$
2: **for** $i = 0$ to n **do**
3: Select random value $r_i \leftarrow \{0, 1\}^r$
4: **end for**
5: $c_0 = \text{Enc}(k, r_0||START||r_1)$
6: **for** Each block m_i, $i = 1$ to $n - 1$ **do**
7: $c_i = \text{Enc}(k, r_i||m_i||r_{i+1})$
8: **end for**
9: $c_n = \text{Enc}(k, r_n||m_n||r_0)$
10: $r^* = \oplus_{i=1}^{n} r_i$
11: $c^* = \text{Enc}(k, r^* \oplus r_0||0^{b-2r}||r^*)$
12: Return $c_0, c_1, c_2, \cdots, c_n, c^*$

We assume that the keyword "START" is not part of the valid message space. c_0 identifies the start of the message and c^* identifies the end of the message and also contains the checksum. Decryption is performed by computing $\mathrm{Dec}(k, c_i)$, $i = 0, 1, \cdots, n$ and $\mathrm{Dec}(k, c^*)$. The following checks are performed to verify the integrity:

- The first block contains the keyword "START".
- The r_i values are chained correctly.
- The decryption of c^* contains the correct r_0 and the checksum.

If the integrity checks succeed, the decryption algorithm outputs the message M, otherwise $\perp$.

Similar to rECB mode, this mode supports replace, insert and delete operations.

A main challenge in implementing an incremental encryption scheme is to manage the blocks in order to efficiently support insert, delete and replace operations.

4.5 An Example Application

We now illustrate how the internals of our inline AB-GKM scheme works through a simplified example in a healthcare scenario. This discussion is based on the information available at [42].

A hospital's data center Owner has to broadcast an XML file "EHR.xml" which contains the electronic health record (EHR) of a patient to the hospital's employees.

```
<PatientRecord>
  <ContactInfo>
    ... ...
  </ContactInfo>
  <BillingInfo>
    ... ...
  </BillingInfo>
```

```
<ClinicalRecord>
  <HistoryOfPresentIllness>
    ... ...
  </HistoryOfPresentIllness>
  <PastMedicalHistory>
    ... ...
  </PastMedicalHistory>
  <Medication>
    // This has the current prescription
    ... ...
  <Medication>
  <AlergiesAndAdverseReactions>
    ... ...
  </AlergiesAndAdverseReactions>
  <FamilyHistory>
    ... ...
  </FamilyHistory>
  <SocialHistory>
    // Smoking, drinking, etc.
    ... ...
  <SocialHistory>
  <PhysicalExams>
    // Weight, body temperature, skin tests, etc.
    ... ...
  </PhysicalExams>
  <LabRecords>
    // X-rays, etc.
    ... ...
  </LabRecords>
```

```
    <Plan>
      // What needs to be done, etc.
      ... ...
    </Plan>
  </ClinicalRecord>
</PatientRecord>
```

The subdocuments of "EHR.xml", marked with different XML tags, need to be accessed by different employees based on their roles and other identity attributes. Suppose the roles for the hospital's employees are: receptionist (rec), cashier (cas), doctor (doc), nurse (nur), data analyst (dat), and pharmacist (pha). The involved access control policies for "EHR.xml" are

1. $\mathsf{ACP}_1 = (\text{“role} = \text{rec”}, \{\langle\text{ContactInfo}\rangle\}, \text{“EHR.xml”})$
2. $\mathsf{ACP}_2 = (\text{“role} = \text{cas”}, \{\langle\text{BillingInfo}\rangle\}, \text{“EHR.xml”})$
3. $\mathsf{ACP}_3 = (\text{“role} = \text{doc”}, \{\langle\text{ClinicalRecord}\rangle\}, \text{“EHR.xml”})$
4. $\mathsf{ACP}_4 = (\text{“role} = \text{nur} \wedge \text{level} \geq 59\text{”}, \{\langle\text{ContactInfo}\rangle, \langle\text{Medication}\rangle, \langle\text{PhysicalExams}\rangle, \langle\text{LabRecords}\rangle, \langle\text{Plan}\rangle\}, \text{“EHR.xml”})$
5. $\mathsf{ACP}_5 = (\text{“role} = \text{dat”}, \{\langle\text{ContactInfo}\rangle, \langle\text{LabRecords}\rangle\}, \text{“EHR.xml”})$
6. $\mathsf{ACP}_6 = (\text{“role} = \text{pha”}, \{\langle\text{BillingInfo}\rangle, \langle\text{Medication}\rangle\}, \text{“EHR.xml”})$

"EHR.xml" is divided into subdocuments based on these access control policies:

- $\langle$ContactInfo$\rangle$: ACP_1, ACP_4, ACP_5
- $\langle$BillingInfo$\rangle$: ACP_2, ACP_6
- $\langle$Medication$\rangle$: ACP_3, ACP_4, ACP_6
- $\langle$PhysicalExams$\rangle$: ACP_3, ACP_4
- $\langle$LabReports$\rangle$: ACP_3, ACP_4, ACP_5

- ⟨Plan⟩: ACP_3, ACP_4
- Other stuff: none

The policy configurations and their associated subdocuments are:

- $\mathsf{Pc}_1 = \{\mathsf{ACP}_1, \mathsf{ACP}_4, \mathsf{ACP}_5\} \leftrightarrow \langle$ContactInfo$\rangle$
- $\mathsf{Pc}_2 = \{\mathsf{ACP}_2, \mathsf{ACP}_6\} \leftrightarrow \langle$BillingInfo$\rangle$
- $\mathsf{Pc}_3 = \{\mathsf{ACP}_3, \mathsf{ACP}_4, \mathsf{ACP}_6\} \leftrightarrow \langle$Medication$\rangle$
- $\mathsf{Pc}_4 = \{\mathsf{ACP}_3, \mathsf{ACP}_4\} \leftrightarrow \langle$PhysicalExams$\rangle, \langle$Plan$\rangle$
- $\mathsf{Pc}_5 = \{\mathsf{ACP}_3, \mathsf{ACP}_4, \mathsf{ACP}_5\} \leftrightarrow \langle$LabReports$\rangle$
- $\mathsf{Pc}_6 = \{\} \leftrightarrow$ Other XML tags

Assume that the involved hospital employees have already obtained their identity tokens and have received their secrets through the delivery phase described earlier, and that the secret table $\mathcal{T}$ has been created by Owner. Owner chooses an encryption key K_i for each policy configuration Pc_i to encrypt the associated subdocuments.

Without loss of generality, we focus on the case of $\mathsf{Pc}_4 = \{\mathsf{ACP}_3, \mathsf{ACP}_4\}$ and use the visible records in Table 4.1 for demonstration. An SQL-styled database query
SELECT * FROM $\mathcal{T}$ WHERE 'role = doc' <> NULL
returns two rows containing pseudonyms pn-0012 and pn-1492, corresponding to the employees which can potentially access subdocuments to which ACP_3 applies. Similarly, it can be easily seen that an employee under pn-1492 is the only one who may satisfy ACP_4. The Owner then chooses $N = 3$, and random values z_1, z_2, z_3. For the employee under pn-0012 whose secret for the attribute condition "role = doc" is 86571, the Owner computes values

$$a_{1,1} = H(86571||z_1), a_{1,2} = H(86571||z_2), a_{1,3} = H(86571||z_3).$$

The Owner executes a similar computation for the user under pn-1492 thus obtaining the values

$$a_{2,1} = H(13011||z_1), a_{2,2} = H(13011||z_2), a_{2,3} = H(13011||z_3).$$

By now the Owner has computed both required rows of matrix A for ACP_3, and will process ACP_4. In this case, for pn-1492 whose secrets corresponding to the two conditions "role = nur" and "level $\geq$ 59" are $r_{3,1}$ and $r_{3,2}$, respectively, the Owner computes

$a_{3,1} = H(11109||60987||z_1), a_{3,2} = H(11109||60987||z_2),$

$a_{3,3} = H(11109||60987||z_3).$

For simplicity and illustration purpose, assume $q = 17$, and the resulting matrix over $\mathbb{F}_{17}$

$$A = \begin{pmatrix} 1 & 15 & 3 & 4 \\ 1 & 4 & 13 & 3 \\ 1 & 12 & 5 & 6 \end{pmatrix}.$$

The Owner solves $AY = 0$ for a non-trivial $Y = (4, 4, 3, 3)^T$. Let $K_4 = 11$. The Owner sets

$$X = Y + (K_4, 0, 0, 0)^T = (15, 4, 3, 3)^T.$$

The Owner publishes X, z_1, z_2, z_3 with the associated subdocuments $\langle$PhysicalExams$\rangle$, $\langle$Plan$\rangle$, which are encrypted with a symmetric encryption key $K_4 = 11$.

Suppose that the employee under pn-0012 is a doctor, thus satisfies ACP_3 and has correctly received the secret during the delivery process. To obtain the decryption key K_4, the doctor computes $a_{1,1} = 15$, $a_{1,2} = 3$ and $a_{1,3} = 4$ as the Owner did, then calculates

$$K_4 = (1, a_{1,1}, a_{1,2}, a_{1,3}) \cdot X = (1, 15, 3, 4) \cdot (15, 4, 3, 3)^T = 11.$$

The doctor can now use this key to decrypt the subdocuments $\langle$PhysicalExams$\rangle$, $\langle$Plan$\rangle$.

Suppose that the employee under pn-1492 is a nurse of level 58. Then it satisfies neither ACP_3 nor ACP_4; therefore it cannot receive the secrets 11109 or 13001. Al-

though this nurse has the correct secret 60987 for attribute condition "role = nur", it is not able to compute any of $a_{2,i}$ or $a_{3,i}$, $i = 1, 2, 3$, and thus is not able to obtain a KEV to derive the decryption key K_4. Hence it cannot access the subdocuments $\langle$PhysicalExams$\rangle$, $\langle$Plan$\rangle$.

The process is similar for the other policy configurations. It is worth remarking, though, that for the policy configuration Pc_6, which is an empty set, the Owner can just encrypt the associated subdocuments with an encryption key K_6 without the need of publishing X or z_i, because in this case no employee is authorized to access this portion of data.

4.6 Experimental Results

In this section, we present experimental results for various parameters in our system. We have built a fully functioning system in C/C++ that incorporates our techniques for privacy preserving secret delivery based on the OCBE protocols, and efficient key management using the inline AB-GKM scheme.

The experiments were performed on a machine running GNU/Linux kernel version 2.6.27 with an Intel® Core™ 2 Duo CPU T9300 2.50GHz and 4 Gbytes memory. Only one processor was used for computation. The code is built with 64-bit gcc version 4.3.2, optimization flag -O2. The code is built over the G2HEC C++ library [51], which implements the arithmetic operations in the Jacobian groups of genus 2 curves. For the secret delivery and group key management phases, we use V. Shoup's NTL library [37] version 5.4.2 for finite field arithmetic, and SHA-1 implementation of OpenSSL [38] version 0.9.8 for cryptographic hashing.

4.6.1 Privacy Preserving Secret Delivery

The secret delivery phase uses the OCBE protocols, which consist of three major steps: 1) extra commitments generation (OCBE for inequality conditions only) at

the Usr, 2) envelope composition at the Owner, and 3) envelope opening at the Usr.[6] In this section, we evaluate the performance of these three steps for both EQ- and GE-OCBE protocols.

We choose the group G to be the rational points of the Jacobian variety (aka. Jacobian group) of a genus 2 curve

$$C : y^2 = x^5 + 26828108228393556449007 36x^3$$
$$+22659135529599310290211 6x^2 + 25476747159529297178999 18x$$
$$+47973099597084896730593 50$$

over the prime field $\mathbb{F}_q$, with $q = 5 \cdot 10^{24} + 8503491$ (83 bits). The Jacobian group of this curve has a prime order
p =2499999999999413043860099940220946396619751607569 9 (164 bits).[7]

Table 4.2: Average computation time for running one round of the EQ-OCBE protocol

Computation	Time (in ms)
Create Extra Commitments (Usr)	0.00
Open Envelope (Usr)	35.25
Compose Envelope (Owner)	11.80

The OCBE parameter generation program chooses non-unit points g and h in the Jacobian group as the base points for constructing the Pedersen commitments.

We use attribute values that satisfy the attribute conditions in the policy. We expect a similar running time if the attribute values do not satisfy the attribute conditions in the policy. For GE-OCBE, we vary the value of the ℓ parameter, which controls the range of the difference between the committed value x and the value x_0 specified in the policy, from 5 to 40, and performed evaluation accordingly. In this

[6]Interested readers may refer to [46, 52] for details.
[7]The data is taken from [53].

experiment, we run both EQ- and GE-OCBE protocols for randomly chosen data, for 50 rounds, and take the average values. Figure 4.2 and Table 4.2 report the average running time of one round of the GE-OCBE protocol and the EQ-OCBE protocol, respectively.

The experimental results show that the overall computation takes at most a few seconds for the privacy preserving registration through the OCBE protocols when all possible identity attribute values lie within an interval of width up to 2^{40}. Because of the impact of the values of ℓ on the performance of the secret delivery, it is important to choose ℓ as small as possible, while at the same time large enough to upper-bound the attribute values. For example, the identity attribute "age" (in years) usually has values from 0 to 200 and can be represented using 8 bits. In this case, it is sufficient to choose ℓ to be 8. We expect other OCBE protocols for inequality predicates to have a performance similar to that of GE-OCBE, because the design and operations are similar.

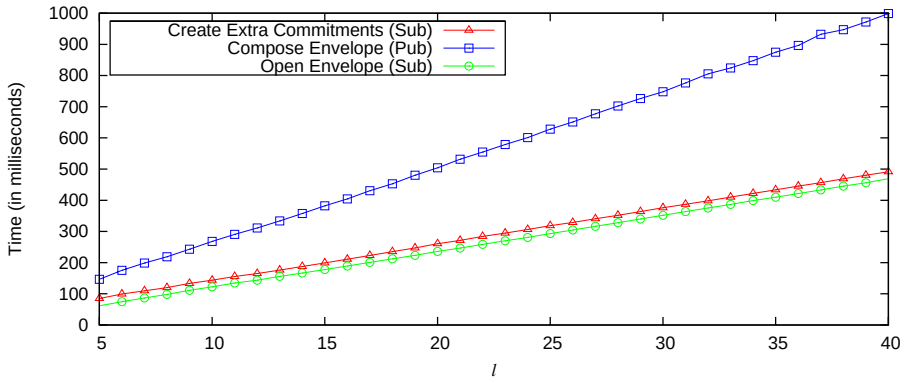

Figure 4.2.: Average computation time for running one round of GE-OCBE protocol

4.6.2 Data and Key Management

In Chapter 3, we provided experimental results only for the Access Tree AB-GKM. In this section, we report experimental results for the Inline AB-GKM which

is the AB-GKM scheme used in this work. We perform experiments to evaluate the performance of generation of the ACVs at the Owner and the key derivation from the ACVs at the Usr, and the size of the ACVs for different system parameters including the number of maximum users and the number of attribute conditions. All finite field arithmetic operations are performed in an 80-bit prime field.

The following experiments are performed with different *user configurations*. A user configuration indicates the number of current Usrs and the maximum user limit N. For example, the configuration '25% Usrs' with $N = 1000$, has 250 Usrs. We use 25 policies, each on average containing two conditions. Each Usr satisfies the policy in the policy configuration under consideration. We illustrate the experiments for one data item, as computations related to different data items are independent and similar, and thus can be performed in parallel.

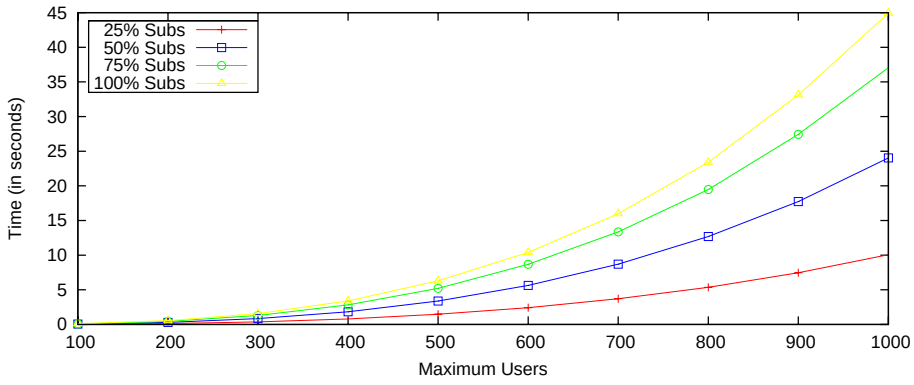

Figure 4.3.: Time to generate an ACV for different user configurations

Figure 4.3 reports the average time spent in computing an ACV corresponding to the matrix A for different user configurations. An ACV is a random vector in the null space of matrix A. We generate an ACV by first computing a basis of the null space of A, then choosing the ACV as a random linear combination of the basis vectors. For a given N, the ACV computation time increases with the number of current users. This is consistent with the fact that as the number of current users increases, the number of rows in the matrix A (consequently the rank of A) increases, requiring an

increasing amount of elementary matrix operations to compute the null space for the linear solver of NTL. As shown in Figure 4.3, this computation is efficient (less than 45 seconds on a personal computer) for reasonably large N values.

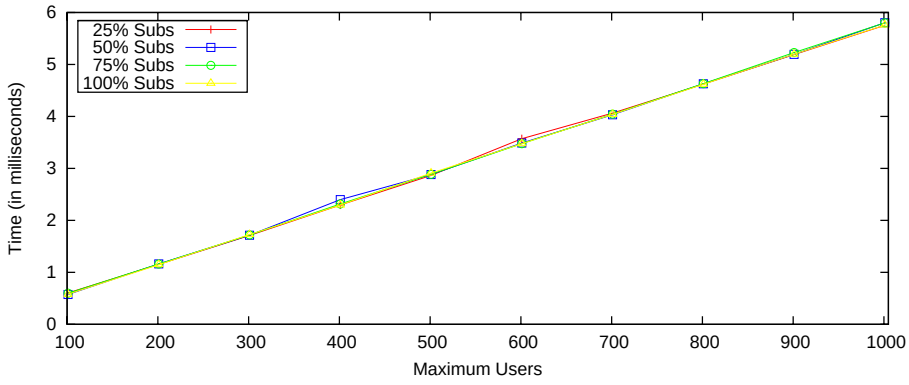

Figure 4.4.: Key derivation time for different user configurations

Figure 4.4 reports the average time for Usrs to derive the symmetric keys from ACVs and KEVs for different user configurations. Key derivation is performed by Usrs whose computational capabilities may be limited. Therefore, an efficient decryption key derivation process is desired. As Figure 4.4 shows it not only incurs minimal computational costs (a few milliseconds), but also increases only linearly with N.

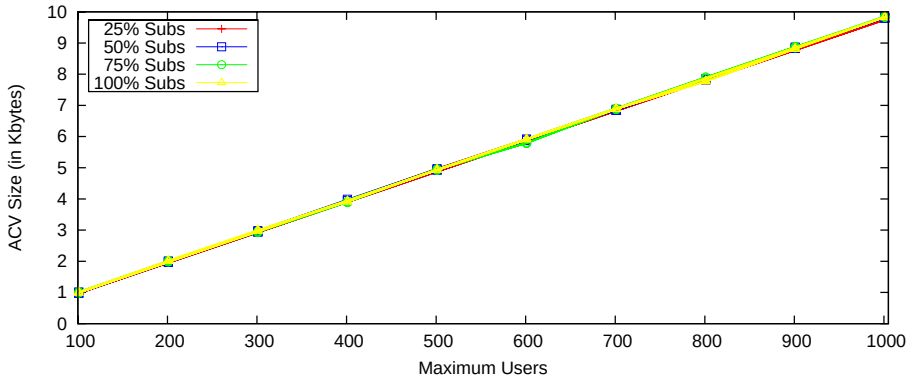

Figure 4.5.: Size of ACV for different user configurations

Figure 4.5 shows the average size of ACVs for different user configurations. Another design goal of our approach is to keep the additional communication overhead minimum. In order to achieve this goal, the Owner compresses the ACVs before broadcasting them with the encrypted data. As Figure 4.5 indicates, our approach only requires a few kilobytes to transmit these vectors, and the size increases only linearly with N.

In the following experiment, we measure the time for ACV generation (at Owner) and key derivation (at Usr) by varying the average number of attribute conditions per policy, and keeping the number of policies and the maximum number of users fixed at 25 and 500, respectively.

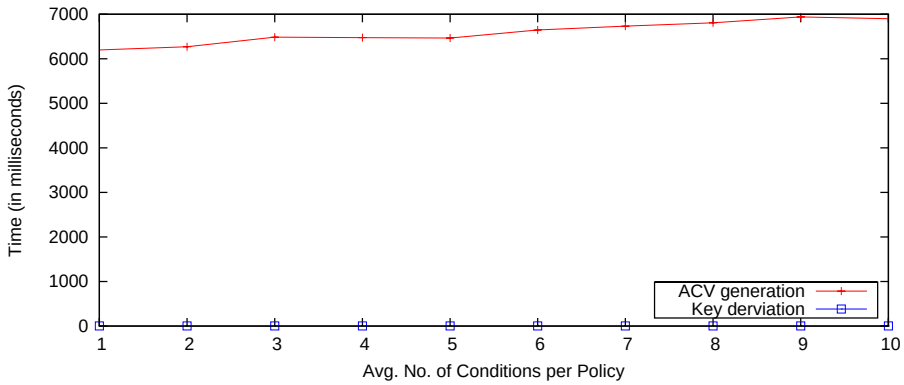

Figure 4.6.: ACV generation and key derivation for different number of conditions per policy

Figure 4.6 shows the average running time for ACVs generation at Owner and symmetric decryption key derivation at Usr, for different number of conditions per policy. As the number of conditions per policy increases, the key derivation time remains almost constant but the ACV generation time slightly increases (by less than 100 milliseconds).

4.6.3 Encryption Management

In this section, we compare the incremental encryption proposed as an improvement to the SLE approach against the traditional encryption.

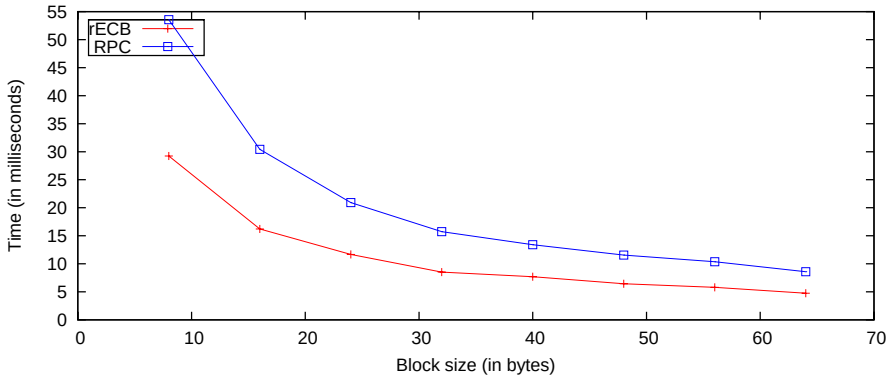

Figure 4.7.: Different incremental encryption modes

Figure 4.7 shows the average overall encryption time as the block size varies while the size of the document remains at 1K. The RPC mode requires more time as it adds integrity checks in addition to encrypting each block. The average time decreases as the size of the block increases since the number of blocks that have to be handled decreases.

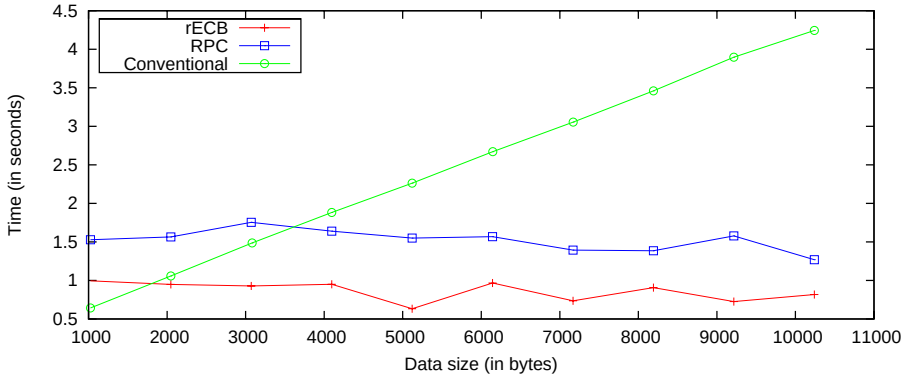

Figure 4.8.: Average time to perform insert operation

Figure 4.8 reports the average time to perform a random insert operation of data of different sizes while the block size remains at 16 bytes. The time remains almost constant for different data sizes. The RPC mode requires more time than the rECB mode since it additionally has to read additional blocks and update the checksum. It is clear that with large data, incremental encryption can save a considerable amount of time. Other modification operations also demonstrate similar pattern.

5 PRIVACY PRESERVING PULL BASED SYSTEMS: TWO LAYER ENCRYPTION APPROACH

In the previous chapter, we proposed an approach called single layer encryption (SLE) follows the conventional data outsourcing scenario where the Owner enforces *all* ACPs through selective encryption and uploads encrypted data to the untrusted Cloud. The SLE approach supports fine-grained attribute based ACPs and preserves the privacy of users from the Cloud. However, in such an approach, the Owner is in charge of encrypting the data before uploading them on the third-party server as well re-encrypting the data whenever user credentials or authorization policies change and managing the encryption keys. The Owner has to download all affected data before before performing the selective encryption. The Owner thus incurs high communication and computation costs, which then negate the benefits of using a third party service. A better approach should delegate the enforcement of fine-grained access control to the Cloud, so to minimize the overhead at the Owner, whereas at the same time assuring data confidentiality from the third-party server.

In this chapter, we propose an approach, based on two layers of encryption, that addresses such requirement. Under our approach, referred to as *two layer encryption* (TLE), the Owner performs a coarse grained encryption, whereas the Cloud performs a fine grained encryption on top of the data encrypted by the coarse grained encryption. A challenging issue in our approach is how to decompose attribute based access control policies (ACPs) such that the two layer encryption can be performed. In order to delegate as much access control enforcement as possible to the Cloud, one needs to decompose the ACPs such that the Owner manages minimum number of attribute conditions in those ACPs that assures the confidentiality of data from the Cloud. Each ACP should be decomposed to two sub ACPs such that the conjunction of the two sub ACPs result in the original ACP. The two layer encryption should

be performed such that the Owner first encrypts the data based on one set of sub ACPs and the Cloud re-encrypts the encrypted data using the other set of ACPs. The two encryptions together enforce the ACP as users should perform two decryptions to access the data. For example, if the ACP is $(C_1 \wedge C_2) \vee (C_1 \wedge C_3)$, the ACP can be decomposed as two sub ACPs C_1 and $C_2 \vee C_3$. Notice that the decomposition is consistent; that is, $(C_1 \wedge C_2) \vee (C_1 \wedge C_3) = C_1 \wedge (C_2 \vee C_3)$. The Owner enforces the former by encrypting the data for the users satisfying the former and the Cloud enforces the latter by re-encrypting the Owner encrypted data for the users satisfying the latter. Since the Cloud does not handle C_1, it cannot decrypt Owner encrypted data and thus confidentiality is preserved. Notice that users should satisfy the original ACP to access the data by performing two decryptions. We show that the problem of decomposing ACPs for coarse and fine grained encryption while assuring the confidentiality of data from the third party and the two encryptions together enforcing the ACPs is NP-complete. We propose novel optimization algorithms to construct near optimal solutions to this problem. Under our approach, the third party server supports two services - the storage service, which stores encrypted data, and the access control service, which performs the fine grained encryption.

We utilize the efficient Access Tree AB-GKM scheme introduced in Chapter 3 allows users whose attributes satisfy a certain ACP to derive the group key and decrypt the content they are allowed to access from the Cloud. Our system assures the confidentiality of the data and preserves the privacy of users from the access control service as well as the cloud storage service while delegating as much of the access control enforcement as possible to the third party through the two layer encryption technique.

The TLE approach has many advantages. When the policy or user dynamics changes, only the outer layer of the encryption needs to be updated. Since the outer layer encryption is performed at the third party, no data transmission is required between the Owner and the third party. Further, both the Owner and the third party service utilize the AB-GKM scheme introduced in Chapter 3 for key management

whereby the actual keys do not need to be distributed to the users. Instead, users are given one or more secrets which allow them to derive the actual symmetric keys for decrypting the data.

The rest of the chapter is organized as follows. An overview of the TLE approach is given in Section 5.1. Section 5.2 provides a detailed treatment of the policy decomposition for the purpose of two layer encryption. Section 5.3 gives a detailed description of the TLE approach. We briefly analyze the trade-offs, the security and the privacy of the overall systems in Section 5.4. Section 5.5 reports experimental results for policy decomposition algorithms and the SLE vs. the TLE approaches.

5.1 Overview

We now give an overview of our solution to the problem of delegated access control to outsourced data in the cloud. A detailed description is provided in Section 4.3. Like the SLE system described in Section 4.3, the TLE system consists of the four entities, Owner, Usr, IdP and Cloud. However, unlike the SLE approach, the Owner and the Cloud collectively enforce ACPs by performing two encryptions on each data item. This two layer enforcement allows one to reduce the load on the Owner and delegates as much access control enforcement duties as possible to the Cloud. Specifically, it provides a better way to handle data updates, user dynamics, and policy changes. Figure 5.1 shows the system diagram of the TLE approach. The system goes through one additional phase compared to the SLE approach. We give an overview of the six phases below:

Identity token issuance: IdPs issue identity tokens to Usrs based on their identity attributes.

Policy decomposition: The Owner decomposes each ACP into at most two sub ACPs such that the Owner enforces the minimum number of attributes to assure confidentiality of data from the Cloud. It is important to make sure that the decomposed

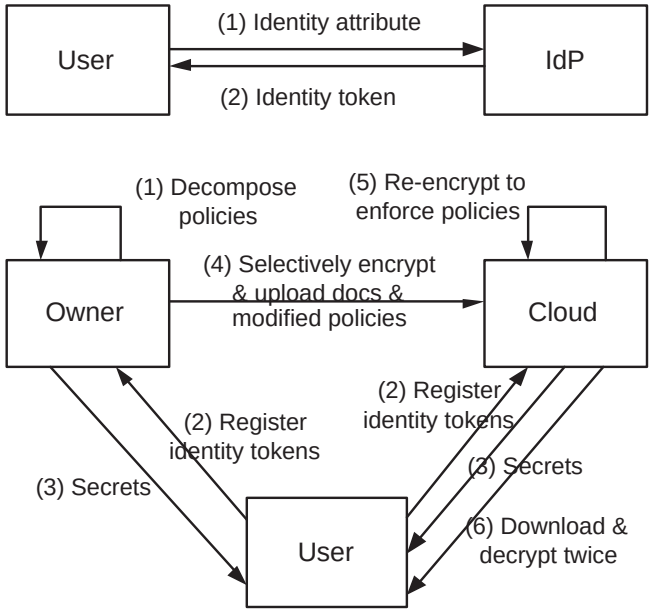

Figure 5.1.: Two layer encryption approach

ACPs are consistent so that the sub ACPs together enforce the original ACPs. The Owner enforces the confidentiality related sub ACPs and the Cloud enforces the remaining sub ACPs.

Identity token registration: Usrs register their identity tokens in order to obtain secrets to decrypt the data that they are allowed to access. Usrs register only those identity tokens related to the Owner's sub ACPs and register the remaining identity tokens with the Cloud in a privacy preserving manner. It should be noted that the Cloud does not learn the identity attributes of Usrs during this phase.

Data encryption and uploading: The Owner first encrypts the data based on the Owner's sub ACPs in order to hide the content from the Cloud and then uploads them along with the public information generated by the AB-GKM::KeyGen algorithm and the remaining sub ACPs to the Cloud. The Cloud in turn encrypts the data

based on the keys generated using its own AB-GKM::KeyGen algorithm. Note that the AB-GKM::KeyGen at the Cloud takes the secrets issued to Usrs and the sub ACPs given by the Owner into consideration to generate keys.

Data downloading and decryption: Usrs download encrypted data from the Cloud and decrypt the data using the derived keys. Usrs decrypt twice to first remove the encryption layer added by the Cloud and then by the Owner. As access control is enforced through encryption, Usrs can decrypt only those data for which they have valid secrets.

Encryption evolution management: Over time, either ACPs or user credentials may change. Further, already encrypted data may go through frequent updates. In such situations, data already encrypted must be re-encrypted with a new key. As the Cloud performs the access control enforcing encryption, it simply re-encrypts the affected data without the intervention of the Owner.

5.2 Policy Decomposition

Recall that in the SLE approach, the Owner incurs a high communication and computation overhead since it has to manage all the authorizations when user dynamics or ACPs change. If the access control related encryption is somehow delegated to the Cloud, the Owner can be freed from the responsibility of managing authorizations through re-encryption and the overall performance would thus improve. Since the Cloud is not trusted for the confidentiality of the outsourced data, the Owner has to initially encrypt the data and upload the encrypted data to the cloud. Therefore, in order for the Cloud to allow to enforce authorization policies through encryption and avoid re-encryption by the Owner, the data may have to be encrypted again to have two encryption layers. We call the two encryption layers as *inner encryption layer* (IEL) and *outer encryption later* (OEL). IEL assures the confidentiality of the data

with respect to the Cloud and is generated by the Owner. The OEL is for fine-grained authorization for controlling accesses to the data by the users and is generated by the Cloud.

An important issue in the TLE approach is how to distribute the encryptions between the Owner and the Cloud. There are two possible extremes. The first approach is for the Owner to encrypt all data items using a single symmetric key and let the Cloud perform the complete access control related encryption. The second approach is for the Owner and the Cloud to perform the complete access control related encryption twice. The first approach has the least overhead for the Owner, but it has the highest information exposure risk due to collusions between Usrs and the Cloud. Further, IEL updates require re-encrypting all data items. The second approach has the least information exposure risk due to collusions, but it has the highest overhead on the Owner as the Owner has to perform the same task initially as in the SLE approach and, further, needs to manage all identity attributes. An alternative solution is based on decomposing ACPs so that the information exposure risk and key management overhead are balanced. The problem is then how to decompose the ACPs such that the Owner has to manage the minimum number of attributes while delegating as much access control enforcement as possible to the Cloud without allowing it to decrypt the data. In what follow we propose such an approach to decompose and we also show that the policy decomposition problem is hard.

5.2.1 Policy Cover

We define the *policy cover problem* as the the optimization problem of finding the minimum number of attribute conditions that "covers" *all* the ACPs in the $\mathcal{ACPB}$. We say that a set of attribute conditions covers the $\mathcal{ACPB}$ if in order to satisfy *any* ACP in the $\mathcal{ACPB}$, it is necessary that at least one of the attribute conditions in the set is satisfied. We call such a set of attribute conditions as the *attribute condition cover*. For example, if $\mathcal{ACPB}$ consists of the three simple ACPs $\{C_1 \wedge C_2, C_2 \wedge C_3, C_4\}$,

the minimum set of attributes that covers $\mathcal{ACPB}$ is $\{C_2, C_4\}$. C_2 should be satisfied in order to satisfy the ACPs $C_1 \wedge C_2$ and $C_2 \wedge C_3$. Notice that satisfying C_2 is not sufficient to satisfy the ACPs. The set is minimum since the set obtained by removing either C_2 or C_4 does not satisfy the cover relationship.

Algorithm 3 GEN-GRAPH

1: $\mathcal{C} = \phi$
2: **for** Each $\mathsf{ACP}_i \in \mathcal{ACPB}$, $i = 1$ to N_p **do**
3: $\quad \mathsf{ACP}'_i \leftarrow$ Convert ACP_i to DNF
4: $\quad$ **for** Each conjunctive term c of ACP'_i **do**
5: $\quad\quad$ Add c to $\mathcal{C}$
6: $\quad$ **end for**
7: **end for**
8: //Represent the conditions as a graph
9: $G = (E, V)$, $E = \phi$, $V = \phi$
10: **for** Each conjunctive term $c_i \in \mathcal{C}$, $i = 1$ to N_c **do**
11: $\quad$ Create vertex v, if $v \notin V$, for each AC in c_i
12: $\quad$ Add an edge e_i between v_i and each vertex already added for c_i
13: **end for**
14: Return G

We define the related decision problem as follows.

Definition 5.2.1 (POLICY-COVER) *Determine whether $\mathcal{ACPB}$ has a cover of k attribute conditions.*

The following theorem states that this problem is NP-complete.

Theorem 5.2.1 *The POLICY-COVER problem is NP-complete.*

Proof We first show that POLICY-COVER $\in$ NP. Suppose that we are given a set of ACPs $\mathcal{ACPB}$ which contains the attribute condition set $\mathcal{AC}$, and integer k.

For simplicity, we assume that each ACP is a conjunction of attribute conditions. However, the proof can be trivially extended to ACPs having any monotonic Boolean expression over attribute conditions. The certificate we choose has a cover of attribute conditions $\mathcal{AC}' \subset \mathcal{AC}$. The verification algorithm affirms that $|\mathcal{AC}'| = k$, and then it checks, for each policy in the $\mathcal{ACPB}$, that at least one attribute condition in $\mathcal{AC}'$ is in the policy. This verification can be performed trivially in polynomial time. Hence, POLICY-DECOM is NP.

Now we prove that the POLICY-COVER problem is NP-hard by showing that the vertex cover problem, which is NP-Complete, is polynomial time reducible to the POLICY-COVER problem. Given an undirected graph $G = (V, E)$ and an integer k, we construct a set of ACPs $\mathcal{ACPB}$ that has a cover set of size k if and only if G has a vertex cover of size k.

Suppose G has a vertex cover $V' \subset V$ with $|V'| = k$. We construct a set of ACPs $\mathcal{ACPB}$ that has a cover of k attribute conditions as follows. For each vertex $v_i \in V$, we assign an attribute condition C_i. For each vertex $v_j \in V'$, we construct an access control policy by obtaining the conjunction of attribute conditions as follows.

- Start with the attribute condition C_j as the ACP P_j
- For each edge (v_j, v_r), add C_r to the ACP as a conjunctive literal (For example, if the edges are $(v_j, v_a), (v_j, v_b)$ and (v_j, v_c), we get $P_j = C_j \wedge C_a \wedge C_b \wedge C_c$)

At the end of the construction we have a set of distinct access control policies $\mathcal{ACPB}$ with size k. We construct the attribute condition set $\mathcal{AC} = \{C_1, C_2, \cdots, C_k\}$ such that C_i corresponds to each vertex in V'. In order to satisfy all access control policies, the attribute conditions in $\mathcal{AC}$ must be satisfied. Hence, $\mathcal{AC}$ is an attribute condition cover of size k for the ACPs $\mathcal{ACPB}$.

Conversely, suppose that $\mathcal{ACPB}$ has an attribute condition cover of size k. We construct G such that each attribute condition corresponds to a vertex in G and an edge between v_i and v_j if they appear in the same access control policy. Let this vertex set be V_1. Then we add the remaining vertices to G corresponding to other

attribute conditions in the access control policies and add the edges similarly. Since the access control policies are distinct there will be at least one edge (v_i, u) for each vertex v_i in attribute condition cover such that $u \notin V_1$. Hence G has a vertex cover of size $V_1 = k$. ■

Since the POLICY-COVER problem is NP-complete, one cannot find a polynomial time algorithm for finding the minimum attribute condition cover. In the following section we present two approximation algorithms for the problem.

The APPROX-POLICY-COVER1 algorithm 4 takes as input the set of ACPs $\mathcal{ACPB}$ and returns a set of attribute conditions whose size is guaranteed to be no more than twice the size of an optimal attribute condition cover. APPROX-POLICY-COVER1 utilizes the GEN-GRAPH algorithm 3 to first represent $\mathcal{ACPB}$ as a graph.

Algorithm 4 APPROX-POLICY-COVER1

```
1: G = GEN-GRAPH(ACPB)
2: ACC = φ
3: for Each disconnected subgraph G_i = (V_i, E_i) of G do
4:    if |V_i| == 1 then
5:       Add AC_i corresponding to the vertex to ACC
6:    else
7:       while E_i = φ do
8:          Select a random edge (u, v) of E_i
9:          Add the attribute conditions AC_u and AC_v corresponding to {u, v} to
            ACC.
10:         Remove from E_i every edge incident on either u or v
11:      end while
12:   end if
13: end for
14: Return ACC
```

We give a high-level overview of the GEN-GRAPH algorithm 3. It takes the $\mathcal{ACPB}$ as the input and converts each ACP into DNF (disjunctive normal form). The unique conjunctive terms are added to the set $\mathcal{C}$. For each attribute condition in each conjunctive term in $\mathcal{C}$, it creates a new vertex in G and adds edges between the vertices corresponding to the same conjunctive term. Depending on the ACPs, the algorithm may create a graph G with multiple disconnected subgraphs.

As shown in the APPROX-POLICY-COVER1 algorithm 4, it takes the $\mathcal{ACPB}$ as the input and outputs a near-optimal attribute condition cover $\mathcal{ACC}$. First the algorithm converts the $\mathcal{ACPB}$ to a graph G as shown in the GEN-GRAPH algorithm 3. Then for each disconnected subgraph G_i of G, it finds the near optimal attribute condition cover and add to the $\mathcal{ACC}$. The attribute condition to be added is related at random by selecting a random edge in G_i. Once an edge is considered, all its incident edges are removed from G_i. The algorithm continues until all edges are removed from each G_i. The running time of the algorithm is $O(V + E)$ using adjacency lists to represent G. It can be shown that the APPROX-POLICY-COVER1 algorithm is a polynomial-time 2-approximation algorithm as follows.

Theorem 5.2.2 *APPROX-POLICY-COVER1 is a polynomial-time 2-approximation algorithm.*

Proof The above running time analysis already shows that the algorithm runs in polynomial time. We prove that the AC cover $\mathcal{ACC}$ returned by the algorithm is at most twice the size of an optimal AC cover $\mathcal{ACC}*$.

Let E'_i denote the set of edges picked at random by the algorithm for each disconnected subgraph G_i. In order to cover the edges in E'_i, any AC cover must include at least one endpoint of each edge in E'_i. Since once an edge is selected, all the incident edges are removed, no two edges in E'_i share an endpoint. Therefore, no two edges in E'_i are covered by the same vertex from $\mathcal{ACC}*$ and we have the following lower bound

on the size of the optimal AC cover. Note that if E_i' is empty, i.e., G_i has only one vertex, the only attribute condition is included in the AC cover.

$$|\mathcal{ACC}*| \geq \sum(|E_i|+1)$$

Each execution of the random edge selection picks an edge for which neither of its endpoints are already in $\mathcal{ACC}$. Thus, it gives an upper bound on the size of the AC cover.

$$|\mathcal{ACC}| \leq 2(\sum|E_i|)+1$$

Combining equations and , we get

$$|\mathcal{ACC}| \leq 2|\mathcal{ACC}*|$$

Hence, we prove the theorem. ■

We now present the idea behind our second approximation algorithm, APPROX-POLICY-COVER2, which uses a heuristic to select the attribute conditions. This algorithm is similar to the APPROX-POLICY-COVER1 algorithm 4 except that instead of randomly selecting the edges to be included in the cover, it selects the vertex of highest degree and removes all of its incident edges.

Example 4

A hospital (Owner) supports fine-grained access control on electronic health records (EHRs) and makes these records available to hospital employees (Usrs) through a public cloud (Cloud). Typical hospital employees includes Usrs playing different roles such as receptionist (rec), cashier (cas), doctor (doc), nurse (nur), pharmacist (pha), and system administrator (sys). An EHR document consists of data items including BillingInfo (BI), ContactInfo (CI), MedicationReport (MR), PhysicalExam (PE), LabReports (LR), Treatment Plan (TP) and so on. In accordance with regulations such as health insurance portability and accountability act (HIPAA), the hospital policies specify which users can access which data item(s). In our example system,

there are four attributes, role (rec, cas, doc, nur, pha, sys), insurance plan, denoted as ip, (ACME, MedA, MedB, MedC), type (assistant, junior, senior) and year of service, denoted as yos, (integer). The following is the re-arranged set of ACPs of the hospital such that each data item has a unique ACP.

(“role = rec” ∨ (“role = nur” ∧ “type ≥ junior”), CI)

(“role = cas” ∨ “role = pha”, BI)

(“role = doc” ∧ “ip = 2-out-4”, CR)

((“role = doc” ∧ “ip = 2-out-4”) ∨ “role = pha”, TR)

((“role = doc” ∧ “ip = 2-out-4”) ∨ (“role = nur” ∧ “yos ≥ 5”) ∨ “role = pha”, MR)

((“role = nur” ∧ “type ≥ junior”) ∨ (“role = dat” ∧ “type ≥ junior”) ∨ (“role = doc” ∧ “yos ≥ 2”), LR)

((“role = nur” ∧ “type = senior”) ∨ (“role = dat” ∧ “yos ≥ 4”), PE)

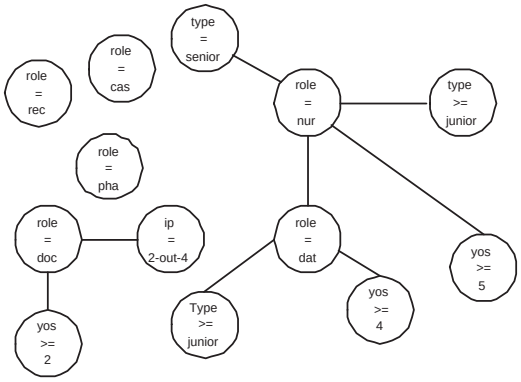

Figure 5.2.: The example graph

Figure 5.2 shows the graph generated by the GEN-GRAPH algorithm for our running example. Notice that there are 5 disconnected graphs. Assume that APPROX-POLICY-COVER2 algorithm is used to construct the AC cover. As mentioned in the approximation algorithm, single vertex graphs are trivially included in the AC cover. The remaining attribute conditions are selected using the greedy heuristic.

That gives us the AC cover $\mathcal{ACC}$ = { "role = rec", "role = cas", "role = pha", "role = doc", "role = nur", "role = dat"}.

5.2.2 Policy Decomposition

The Owner manages only those attribute conditions in $\mathcal{ACC}$. The Cloud handles the remaining set of attribute conditions, $\mathcal{ACB}/\mathcal{ACC}$. The Owner re-writes its ACPs such that they cover $\mathcal{ACC}$. In other words, the Owner enforces the parts of the ACPs related to the ACs in $\mathcal{ACC}$ and Cloud enforces the remaining ACs along with some ACs in $\mathcal{ACC}$. The POLICY-DECOMPOSITION algorithm 5 shows how the ACPs are decomposed into two sub ACPs based on the attribute conditions in $\mathcal{ACC}$.

Algorithm 5 takes the $\mathcal{ACPB}$ and $\mathcal{ACC}$ as input and produces the two sets of ACPs $\mathcal{ACPB}_{\mathsf{Owner}}$ and $\mathcal{ACPB}_{\mathsf{Cloud}}$ that are to be enforced at the Owner and the Cloud respectively. It first converts each policy into DNF and decompose each conjunctive term into two conjunctive terms such that one conjunctive term has only those ACs in $\mathcal{ACC}$ and the other term may or may not have the ACs in $\mathcal{ACC}$. It can be easily shown that the policy decomposition is consistent. That is, the conjunction of corresponding sub ACPs in $\mathcal{ACPB}_{\mathsf{Owner}}$ and $\mathcal{ACPB}_{\mathsf{Cloud}}$ respectively produces an original ACP in $\mathcal{ACPB}$.

Example 5

For our example ACPs, the Owner handles the following sub ACPs.

("role = rec" $\vee$ "role = nur" , CI)

("role = cas" $\vee$ "role = pha", BI)

("role = doc", CR)

("role = doc" $\vee$ "role = pha", TR)

("role = doc" $\vee$ "role = nur" $\vee$ "role = pha", MR)

("role = nur" $\vee$ "role = dat" $\vee$ "role = doc", LR)

("role = nur" $\vee$ "role = dat", PE)

Algorithm 5 POLICY-DECOMPOSITION

1: $\mathcal{ACPB}_{\mathsf{Owner}} = \phi$
2: $\mathcal{ACPB}_{\mathsf{Cloud}} = \phi$
3: **for** Each ACP_i in $\mathcal{ACPB}$ **do**
4: Convert ACP_i to DNF
5: $\mathsf{ACP}_i(\text{owner}) = \phi$
6: $\mathsf{ACP}_i(\text{cloud}) = \phi$
7: **if** Only one conjunctive term **then**
8: Decompose the conjunctive term c into c_1 and c_2 such that ACs in $c_1 \in \mathcal{ACC}$, ACs in $c_2 \notin \mathcal{ACC}$ and $c = c_1 \wedge c_2$
9: $\mathsf{ACP}_i(\text{owner}) = c_1$
10: $\mathsf{ACP}_i(\text{cloud}) = c_2$
11: **else if** At most one term has more than one AC **then**
12: **for** Each single AC term c of ACP'_i **do**
13: $\mathsf{ACP}_i(\text{owner}) \vee= c$
14: $\mathsf{ACP}_i(\text{cloud}) \vee= c$
15: **end for**
16: Decompose the multi AC term c into c_1 and c_2 such that ACs in $c_1 \in \mathcal{ACC}$, ACs in $c_2 \notin \mathcal{ACC}$ and $c = c_1 \wedge c_2$
17: $\mathsf{ACP}_i(\text{owner}) \vee= c_1$
18: $\mathsf{ACP}_i(\text{cloud}) \vee= c_2$
19: **else**
20: **for** Each conjunctive term c of ACP'_i **do**
21: Decompose c into c_1 and c_2 such that ACs in $c_1 \in \mathcal{ACC}$, ACs in $c_2 \notin \mathcal{ACC}$ and $c = c_1 \wedge c_2$
22: $\mathsf{ACP}_i(\text{owner}) \vee= c_1$
23: **end for**
24: $\mathsf{ACP}_i(\text{cloud}) = \mathsf{ACP}'_i$
25: **end if**
26: Add $\mathsf{ACP}_i(\text{owner})$ to $\mathcal{ACPB}_{\mathsf{Owner}}$
27: Add $\mathsf{ACP}_i(\text{cloud})$ to $\mathcal{ACPB}_{\mathsf{Cloud}}$
28: **end for**
29: Return $\mathcal{ACPB}_{\mathsf{Owner}}$ and $\mathcal{ACPB}_{\mathsf{Cloud}}$

As shown in Algorithm 5, the Owner re-writes the ACPs that the Cloud should enforce such that the conjunction of the two decomposed sub ACPs yields an original ACP. In our example, the sub ACPs that the Cloud enforces look like follows.

(“role = rec” $\vee$ “type $\geq$ junior”, CI)

(“role = cas” $\vee$ “role = pha”, BI)

(“ip = 2-out-4”, CR)

(“ip = 2-out-4” $\vee$ “role = pha”, TR)

((“role = doc” $\wedge$ “ip = 2-out-4”) $\vee$ (“role = nur” $\wedge$ “yos $\geq$ 5”) $\vee$ “role = pha”, MR)

((“role = nur” $\wedge$ “type $\geq$ junior”) $\vee$ (“role = dat” $\wedge$ “type $\geq$ junior”) $\vee$ (“role = doc” $\wedge$ “yos $\geq$ 2”), LR)

((“role = nur” $\wedge$ “type = senior”) $\vee$ (“role = dat” $\wedge$ “yos $\geq$ 4”), PE)

5.3 Two Layer Encryption Approach

In this section, we provide a detailed description of the six phases of the TLE approach introduced in Section 5.1. The system consists of the four entities, Owner, Usr, IdP and Cloud. Let the maximum number of users in the system be N, the current number of users be n ($< N$), and the number of attribute conditions N_a.

5.3.1 Identity Token Issuance

IdPs are trusted third parties that issue identity tokens to Usrs based on their identity attributes. It should be noted that IdPs need not be online after they issue identity tokens. An identity token, denoted by $\mathcal{IT}$ has the format { nym, id-tag, c, σ }, where nym is a pseudonym uniquely identifying a Usr in the system, id-tag is the name of the identity attribute, c is the Pedersen commitment for the identity attribute value x and σ is the IdP's digital signature on nym, id-tag and c.

5.3.2 Policy Decomposition

Using the policy decomposition algorithm 5, the Owner decomposes each ACP into at most two sub ACPs such that the Owner enforces the minimum number of attributes to assure confidentiality of data from the Cloud. The algorithm produces two sets of sub ACPs, $\mathcal{ACPB}_{\mathsf{Owner}}$ and $\mathcal{ACPB}_{\mathsf{Cloud}}$. The Owner enforces the confidentiality related sub ACPs in $\mathcal{ACPB}_{\mathsf{Owner}}$ and the Cloud enforces the remaining sub ACPs in $\mathcal{ACPB}_{\mathsf{Cloud}}$.

5.3.3 Identity Token Registration

Usrs register their $\mathcal{IT}$s to obtain secrets in order to later decrypt the data they are allowed to access. Usrs register their $\mathcal{IT}$s related to the attribute conditions in $\mathcal{ACC}$ with the Owner, and the rest of the identity tokens related to the attribute conditions in $\mathcal{ACB}/\mathcal{ACC}$ with the Cloud using the AB-GKM::SecGen algorithm.

When Usrs register with the Owner, the Owner issues them two sets of secrets for the attribute conditions in $\mathcal{ACC}$ that are also present in the sub ACPs in $\mathcal{ACPB}_{\mathsf{Cloud}}$. The Owner keeps one set and gives the other set to the Cloud. Two different sets are used in order to prevent the Cloud from decrypting the Owner encrypted data.

5.3.4 Data Encryption and Upload

The Owner encrypts the data based on the sub ACPs in $\mathcal{ACPB}_{\mathsf{Owner}}$ and uploads them along with the corresponding public information tuples to the Cloud. The Cloud in turn encrypts the data again based on the sub ACPs in $\mathcal{ACPB}_{\mathsf{Cloud}}$. Both parties execute AB-GKM::KeyGen algorithm individually to first generate the symmetric key, the public information tuple PI and access tree $\mathcal{T}$ for each sub ACP. We now give a detailed description of the encryption process.

The Owner arranges the sub ACPs such that each data item has a unique ACP. Note that the same policy may be applicable to multiple data items. Assume that the set of data items $\mathcal{D} = \{d_1, d_2, \cdots, d_m\}$ and the set of sub ACPs $\mathcal{ACPB}_{\mathsf{Owner}} =$

$\{\mathsf{ACP}_1, \mathsf{ACP}_2, \cdots, \mathsf{ACP}_n\}$. The Owner assigns a unique symmetric key, called an ILE key, K_i^{ILE} for each sub $\mathsf{ACP}_i \in \mathcal{ACPB}_{\mathsf{Owner}}$, encrypts all related data with that key and executes the AB-GKM::KeyGen to generate the public PI_i and $\mathcal{T}_i$. The Owner uploads those encrypted data (id, $E_{K_i^{ILE}}(d_i)$, i) along with the indexed public information tuples (i, PI_i, $\mathcal{T}_i$), where $i = 1, 2, \cdots, n$, to the Cloud. The Cloud handles the key management and encryption based access control for the ACPs in $\mathcal{ACPB}_{\mathsf{Cloud}}$. For each sub $\mathsf{ACP}_j \in \mathcal{ACPB}_{\mathsf{Cloud}}$, the Cloud assigns a unique symmetric key K_j^{OLE}, called an OLE key, encrypts each affected data item $E_{K_i^{ILE}}(d_i)$ and produces the tuple (id, $E_{K_j^{OLE}}(E_{K_i^{ILE}}(d_i))$, i, j), where i and j gives the index of the public information generated by the Owner and the Cloud respectively.

5.3.5 Data Downloading and Decryption

Usrs download encrypted data from the Cloud and decrypt twice to access the data. First, the Cloud generated public information tuple is used to derive the OLE key and then the Owner generated public information tuple is used to derive the ILE key using the AB-GKM::KeyDer algorithm. These two keys allow a Usr to decrypt a data item only if the Usr satisfies the original ACP applied to the data item.

For example, in order to access a data item d_i, Usrs download the encrypted data item $E_{K_j^{OLE}}(E_{K_i^{ILE}}(d_i))$ and the corresponding two public information tuples PI_i and PI_j. PI_j is used to derive the key of the outer layer encryption K_j^{OLE} and PI_i used to derive the key of the inner layer encryption K_i^{ILE}. Once those two keys are derived, two decryption operations are performed to access the data item.

5.3.6 Encryption Evolution Management

After the initial encryption is performed, affected data items need to be re-encrypted with a new symmetric key if credentials are added/removed or ACPs are modified. Unlike the SLE approach, when credentials are added or revoked or ACPs are modified, the Owner does not have to involve. The Cloud generates a new sym-

metric key and re-encrypts the affected data items. The Cloud follows the following conditions in order to decide if re-encryption is required.

1. For any ACP, the new group of Usrs is a strict superset of the old group of Usrs, and backward secrecy is enforced.

2. For any ACP, the new group of Usrs is a strict subset of the old group of Usrs, and forward secrecy is enforced for the already encrypted data items.

5.4 Analysis

In this section, we first compare the SLE and the TLE approaches, and then give a high level analysis of the security and the privacy of both approaches.

5.4.1 SLE vs. TLE

Recall that in the SLE approach, the Owner enforces all ACPs by fine-grained encryption. If the system dynamics change, the Owner updates the keys and encryptions. The Cloud merely acts as a storage repository. Such an approach has the advantage of hiding the ACPs from the Cloud. Further, since the Owner performs all access control related encryptions, a Usr colluding with the Cloud is unable to access any data item that is not allowed to access. . However, the SLE approach incurs high overhead. Since the Owner has to perform all re-encryptions when user dynamics or policies change, the Owner has incurs a high overhead in communication and computation. Further, it is unable to perform optimizations such as delayed AB-GKM::ReKey or re-encryption as the Owner has to download, decrypt, re-encrypt and re-upload the data, which could considerably increase the response time if such optimizations are to be performed.

The TLE approach reduces the overhead incurred by the Owner during the initial encryption as well as subsequent re-encryptions. In this approach, the Owner handles only the minimal set of attribute conditions and most of the key management tasks are

performed by the Cloud. Further, when identity attributes are added or removed, or the Owner updates the Cloud's ACPs, the Owner does not have to re-encrypt the data as the Cloud performs the necessary re-encryptions to enforce the ACPs. Therefore, the TLE approach reduces the communication and computation overhead at the Owner. Additionally, the Cloud has the opportunity to perform delayed encryption during certain dynamic scenarios as the Cloud itself manages the OEL keys and encryptions. However, the improvements in the performance comes at the cost of security and privacy. In this approach, the Cloud learns some information about the ACPs.

5.4.2 Security and Privacy

The SLE approach correctly enforces the ACPs through encryption. In the SLE approach, the Owner itself performs the attribute based encryption based on ACPs. The AB-GKM scheme makes sure that only those Usrs who satisfy the ACPs can derive the encryption keys. Therefore, only the authorized Usrs are able to access the data.

The TLE approach correctly enforces the ACPs through two encryptions. Each ACP is decomposed into two ACPs such that the conjunction of them is equivalent to the original ACP. The Owner enforces one part of the decomposed ACPs through attribute based encryption. The Cloud enforces the counterparts of the decomposed ACPs through another attribute based encryption. Usr can access a data item only if it can decrypt both encryptions. As the AB-GKM scheme makes sure that only those Usrs who satisfy these decomposed policies can derive the corresponding keys, a Usr can access a data item by decrypting twice only if it satisfies the two parts of the decomposed ACPs, that is, the original ACPs.

In both approaches, the privacy of the identity attributes of Usrs is assured. Recall that the AB-GKM::SecGen algorithm issues secrets to users based on the identity tokens which hide the identity attributes. Further, at the end of the algorithm neither the Owner nor the Cloud knows if a Usr satisfies a given attribute condition. Therefore,

neither the Owner nor the Cloud learns the identity attributes of Usrs. Note that the privacy does not weaken the security as the AB-GKM::SecGen algorithm makes sure that Usrs can access the issued secrets only if their identity attributes satisfy the attribute conditions.

5.5 Experimental Results

In this section we first present experimental results concerning the policy decomposition algorithms. We then present an experimental comparison between the SLE and TLE approaches.

The experiments were performed on a machine running GNU/Linux kernel version 2.6.32 with an Intel® Core™ 2 Duo CPU T9300 2.50GHz and 4 Gbytes memory. Only one processor was used for computation. Our prototype system is implemented in C/C++. We use V. Shoup's NTL library [37] version 5.4.2 for finite field arithmetic, and SHA-1 and AES-256 implementations of OpenSSL [38] version 1.0.0d for cryptographic hashing and incremental encryption. We use boolstuff library [54] version 0.1.13 to convert policies into DNF. Adjacency list representation is used to construct policy graphs used in the two approximation algorithms for finding a near optimal attribute condition cover.

We utilized the AB-GKM scheme with the subset cover optimization. We used the complete subset algorithm introduced by Naor et. al. [35] as the subset cover. We assumed that 5% of attribute credentials are revoked for the AB-GKM related experiments. All finite field arithmetic operations in our scheme are performed in an 512-bit prime field.

For our experiments, we selected the total number of attribute conditions and the number of attribute conditions per policy based on past case studies [55, 56]. According to the case studies, the number of attribute conditions varies from 50 for a web based conference management system to 1300 for a major European bank. These real systems have upto about 20 attribute conditions per policy. We set the

total attribute condition count between 100-1500 and the the attribute conditions per policy count between 2-20. We generate random Boolean expressions consisting of conjunctions and disjunctions as policies. Each term in the Boolean expression represents a attribute condition.

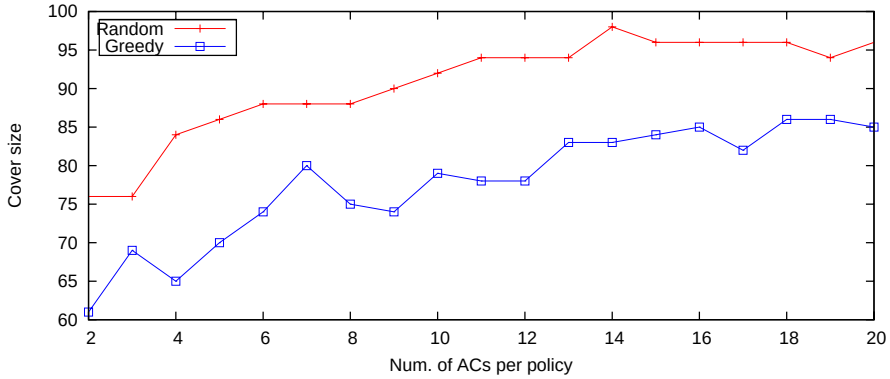

Figure 5.3.: Size of ACCs for 100 attributes

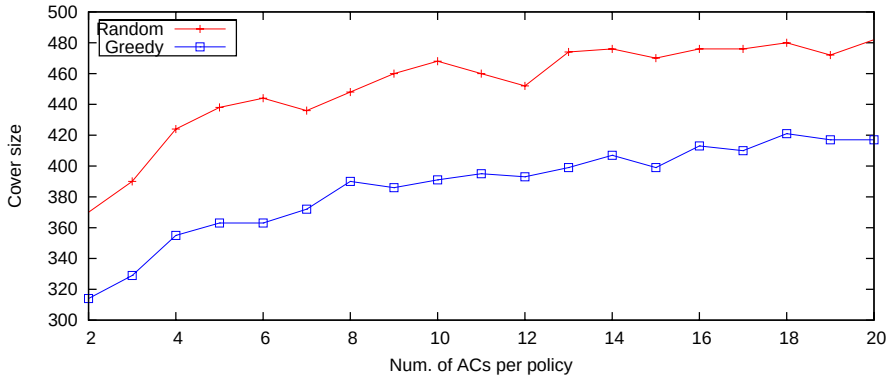

Figure 5.4.: Size of ACCs for 500 attributes

Figures 5.3 5.4 5.5 5.6 show the size of the attribute condition cover, that is, the number of attribute conditions the data owner enforces, for systems having 100, 500, 1000 and 1500 attribute conditions as the number of attribute conditions per policy is increased. In all experiments, the greedy policy cover algorithm performs better.

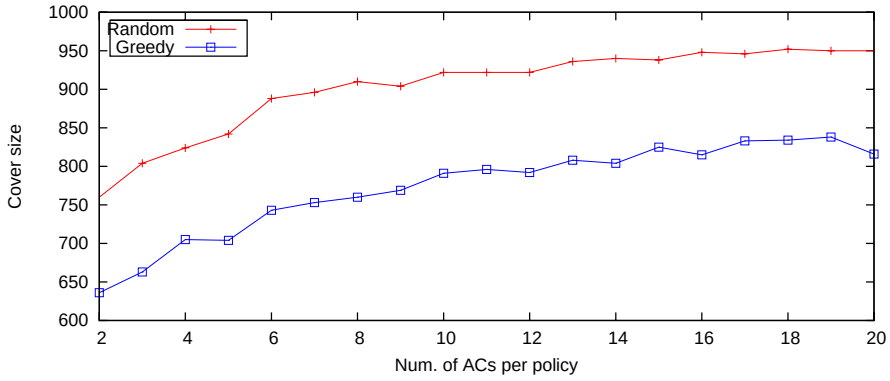

Figure 5.5.: Size of ACCs for 1000 attributes

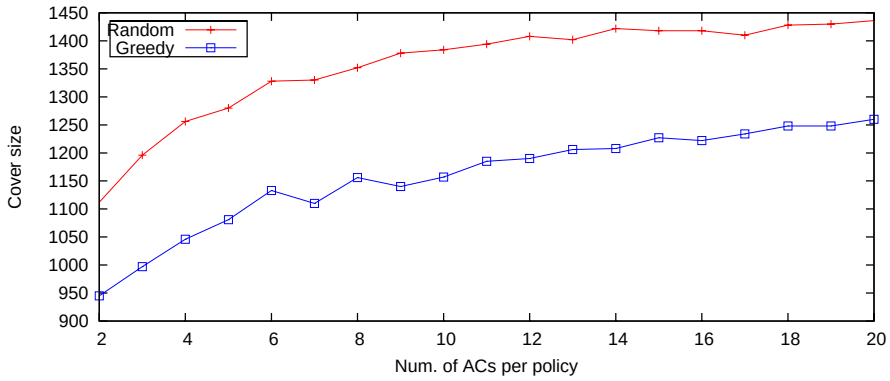

Figure 5.6.: Size of ACCs for 1500 attributes

As the number of attribute conditions per policy increases, the size of the attribute condition cover also increases. This is due to the fact that as the number of attribute conditions per policy increases, the number of distinct disjunctive terms in the DNF increases.

Figures 5.7 5.8 shows the break down of the running time for the complete policy decomposition process for the random and greedy cover algorithms respectively. In this experiment, the number of attribute condition is set to $\{100, 500, 1000\}$ and the maximum number of attribute conditions per policy is set to 5. The total execution time is divided into the execution times of three different components of our scheme.

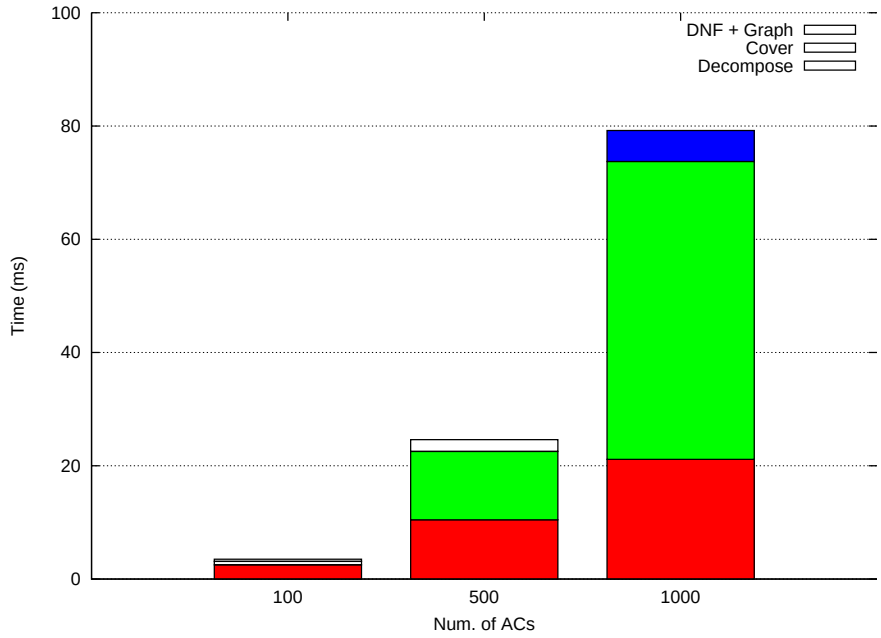

Figure 5.7.: Policy decomposition time breakdown with the random cover algorithm

The"DNF + Graph" time refers to the time required to convert the policies to DNF and construct a in-memory graph of policies using an adjacency list. The "Cover" time refers to the time required to to find the optimal cover and the "Decompose" time refers to time required to to create the updated policies for the data owner and the cloud based on the cover. As can be seen from the graphs, most of the time is spent on finding a near optimal attribute condition cover. It should be noted that the random approximation algorithm runs faster than the greedy algorithm. One reason for this behavior is that each time the latter algorithm selects a vertex it iterates through all the unvisited vertices in the policy graph, whereas the former algorithm simply picks a pair of unvisited vertices at random. Consistent with the worst-cast running times, the"DNF + Graph" and "Decompose" components demonstrate near linear running time, and 'the 'Cover" component shows a non-linear running time.

Figure 5.9 reports the average time spent to execute the AB-GKM::KeyGen with SLE and TLE approaches for different group sizes. We set the number of attribute

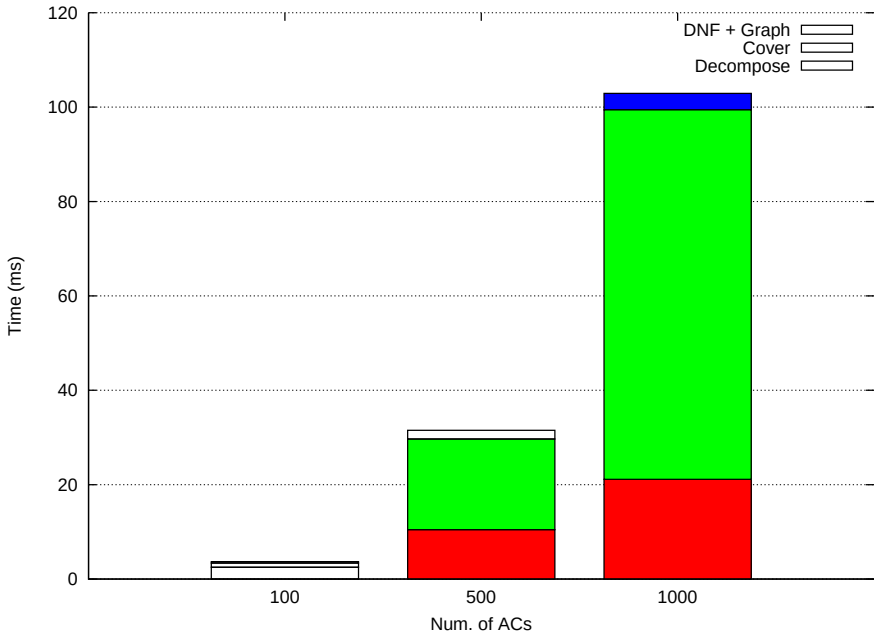

Figure 5.8.: Policy decomposition time breakdown with the greedy cover algorithm

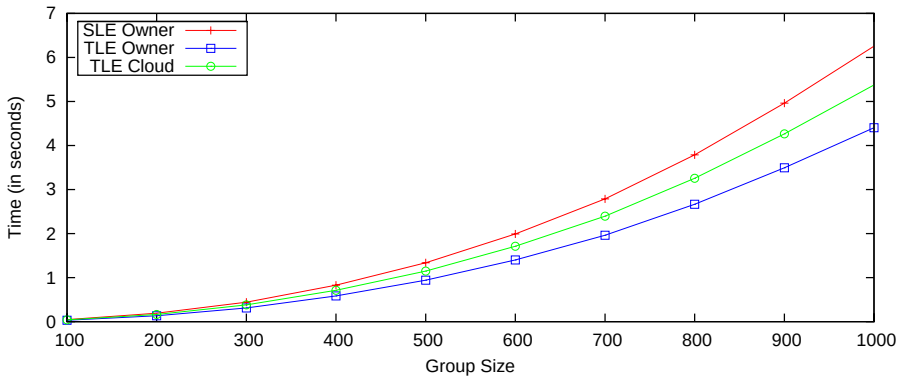

Figure 5.9.: Average time to generate keys for the two approaches

conditions to 1000 and the maximum number of attribute conditions per policy to 5. We utilize the greedy algorithm to find the attribute condition cover. As seen in the diagram, the running time at the Owner in the SLE approach is higher since the Owner has to enforce all the attribute conditions. Since the TLE approach divides

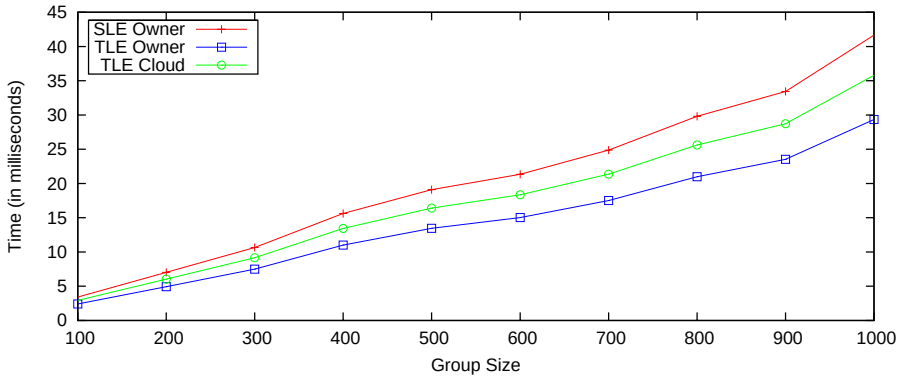

Figure 5.10.: Average time to derive keys for the two approaches

the enforcement cost between the Owner and the Cloud, the running time at the Owner is lower compared to the SLE approach. The running time at the Cloud in the TLE approach is higher than that at the Owner since the Cloud performs fine grained encryption whereas the Owner only performs coarse grained encryption. As shown in Figure 5.10, a similar pattern is observed in the AB-GKM::KeyDer as well.

6 PRIVACY PRESERVING SUBSCRIPTION BASED SYSTEMS

In the last two chapters, our focus was on pull based systems where users pull the content from the third party server. Another popular dissemination model is subscription based publish subscribe systems. The solutions we propose for pull based systems cannot directly be applied to subscription based system as they have the additional requirement of letting the third party server perform content based filtering.

Many systems, including online news delivery, stock quote report dissemination and weather channels, have been or can be modeled as Content-Based Publish-Subscribe (CBPS) systems. Full decoupling of the involved parties, that is, *Content Publishers* (Pubs), *Content Brokers* (Brokers) and *Subscribers* (Subs), in time, space, and synchronization has been the key [57] to seamlessly scale these systems on demand. Hence, CBPS systems have the huge potential to be enabled over cloud computing infrastructures. In a CBPS system, each Sub selectively subscribes to some Brokers to receive different messages. In the most common setting, when Pubs publish messages to some Brokers, these Brokers, in turn, selectively distribute these messages to other Brokers and finally to Subs based on their *subscriptions*, that is, what they subscribed to. These systems, in general, follow a *push based* dissemination approach, that is, whenever new messages arrive, Brokers selectively distribute the messages to Subs. Figure 6.1 shows an example CBPS system.

It is not feasible to have a private Broker network for each CBPS system and most CBPS systems utilize third-party Broker networks which may not be trusted for the confidentiality of the content flowing through them. Because content represents the critical resource in many CBPS systems, its confidentiality from third-party Brokers is important. Consider the popular example of publishing stock market quotes where Subs pay Pub, that is the stock exchange, either for the types of quotes they wish to receive or per usage basis. In such a domain, whenever a new stock quote, referred to

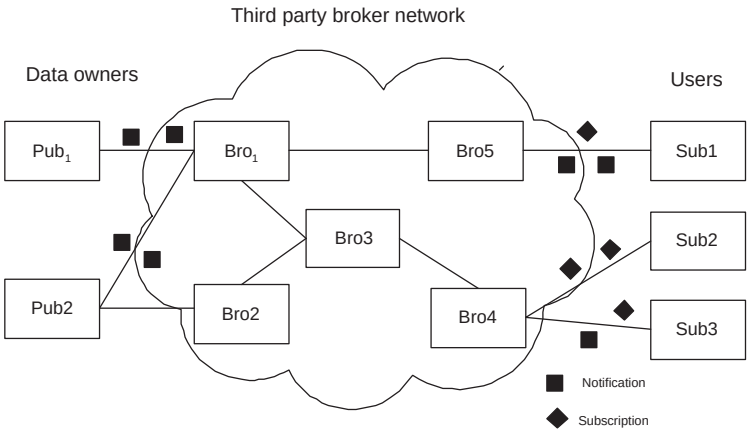

Figure 6.1.: An example CBPS system

in general as a *notification*, is published, Brokers selectively send such a notification only to authorized Subs. Confidentiality is important here because Pubs want to make sure that only paying customers have access to the quotes. We say that a CBPS system provides *publication confidentiality* if Brokers can neither identify the content of the messages published by Pubs nor infer the distribution of *attribute values* of the message [1]. For the stock quote example, in the absence of *publication confidentiality*, Brokers may collect stock quotes, re-sell to others, and/or sell derived market data without any economic incentive to Pubs.

At the same time, the privacy of subscribers is also crucial for many reasons, like business confidentiality or personal privacy. We say that a CBPS system provides *subscription privacy* if Brokers can neither identify what subscriptions Subs made nor relate a set of subscriptions to a specific Sub. Consider again the stock quote example. Suppose for example that Sub subscribes to some Brokers for receiving stock quotes characterized by certain attribute values (e.g. bid price $<$ 2438, 1000 $<$ bid size $<$ 2000, symbol = "MSFT", etc.). In the absence of *subscription privacy*, such a

[1] We assume that a message consists of a set of attribute-value pairs.

subscription can reveal the business strategy of Sub. Further, Brokers may profile *subscriptions* of each Sub and sell them to third parties.

Privacy and confidentiality issues in CBPS have long been identified [6], but little progress has been made to address these issues in a holistic manner. Most of prior work on data confidentiality techniques in the context of CBPS systems is based on the assumption that Brokers are trusted with respect to the privacy of the subscriptions by Subs [7–9]. However, when such an assumption does not hold, both publication confidentiality and subscription privacy are at risk; in the absence of subscription privacy, subscriptions are available in clear text to Brokers. Brokers can infer the content of the notifications by comparing and matching notifications with subscriptions since CBPS systems must allow them to make such decisions to route notifications. As more subscriptions become available to Brokers, the inference is likely to be more accurate. It should also be noted that the above approaches restrict Brokers' ability to make routing decisions based on the content of the messages and thus fail to provide a CBPS system as expressive as a CBPS system that do not address security or privacy issues. Approaches have also been proposed to assure confidentiality/privacy in the presence of untrusted third-party Brokers. These approaches however suffer from one or two major limitations [12–14, 58]: inaccurate content delivery, because of the limited ability of Brokers to make routing decisions based on content; weak security protocols; lack of privacy guarantees. For example, some of these approaches are prone to false positives, that is, sending irrelevant content to Subs.

In this chapter, we propose a novel cryptographic approach along with our AB-GKM scheme to addresses those shortcomings in CBPS systems. To the best of our knowledge, no existing cryptographic solution is able to protect both publication confidentiality and subscription privacy in CBPS systems that address the above shortcomings. A key design goal of our privacy-preserving approach is to design a system which is as expressive as a system that does not consider privacy or security issues. We implement our scheme on top of a popular CBPS system, SIENA [19], and provide several experimental results in order to show our approach is practical.

In summary, our CBPS system exhibits the following properties:

- Notifications and subscriptions are randomized and hidden from Brokers and secure under chosen-ciphertext attacks.
- Both publication confidentiality and subscription privacy are assured as Brokers are able to make routing decisions without decrypting subscriptions and notifications. It is the first system to achieve these properties without sharing keys with Brokers or Subs.
- It supports any type of subscription queries including equality, inequality and range queries at Brokers.
- The computational cost at Brokers are minimized by judiciously distributing the work among Pubs and Subs.

The rest of the chapter is organized as follows. Section 6.1 overviews the CBPS model and the protocols supported by our system. Section 6.2 provides some background knowledge about the main cryptographic primitives used. Section 6.3 provides a detailed description of the proposed protocols. Section 6.4 reports experimental results for the main protocols as well as the system developed on top of SIENA using the main protocols.

6.1 Overview

In this section we give an overview of our proposed scheme by showing the interactions between Pubs, Subs and Brokers, and the trust model. Unless otherwise stated, we describe our approach for one Pub, mainly for brevity. However, our approach can be trivially applied to a system with any number of Pubs. In practice, all the parties in a CBPS system are software programs that act on behalf of real entities like actual organizations or end users, and therefore many of the operations of the protocols we propose are performed transparently to real entities.

Each *notification* is characterized by a set of Attribute-Value Pairs (AVPs). It consists of two parts: the actual message in the encrypted form, which we call the *payload message*, and a set of *blinded AVPs* derived from the payload message. As mentioned earlier, payload message also consists of a set of AVPs. In a blinded AVP, the value is blinded, but the attribute name remains in clear text. The blinding encrypts the value in a special way such that it is computationally infeasible to obtain the value from the blinded values, and that the blinded values are secure under chosen-ciphertext attacks. We provide details on the blind operation in Section 6.3. The payload is encrypted using the AB-GKM scheme based on the acps of the Pub. The AB-GKM scheme makes sure that only those Subs that have valid credentials can access payload messages. The blinded AVPs are placed in the header and the payload message is in the body of the notification. There is a one-to-one mapping between the AVPs in the payload message and the blinded AVPs. Depending on the representation, each attribute name and its corresponding value may be interpreted differently.

In an XML-like syntax, a notification has the following format:

```
<notification>
    <header> -- blinded AVPs -- </header>
    <body> -- enc. payload message -- </body>
</notification>
```

Depending on the representation, each attribute name and its corresponding value may be interpreted differently. For example, the payload could be in a simple property-value format or a complex XML format. If the payload is in XML, attribute names could be the XPaths and values could be the immediate child nodes of XPaths. We use the latter for the examples.

A *subscription* specifies a condition on one of the attributes [2] of the AVPs associated with the notifications. It is an expression of the form ($attr$, $bval_1$, $bval_2$, $bval_3$, op) where $attr$ is the name of the attribute, $bval_1, bval_2, bval_3$ are the blinded values

[2]Note that our approach can easily be extended to subscriptions having multiple attributes.

derived from the actual content v and its additive inverse,[3] and op is a comparison operator, either $\geq$ or $<$. All the other comparison operators are derived from op. Note that our approach supports a wide array of conditions including range queries for numerical attributes and keyword queries for numerical and string attributes.

Example 6

In the stock market quote dissemination system, a payload message, that is, a quote, looks like:

```
<q>
  <symbol>MSFT</symbol>
    <bid>
      <price>2328</price>
      <size>10000</size>
      ...
   </bid>
   <offer>
     <price>2355</price>
     <size>5000</size>
     ...
   </offer>
</q>
```

The set of AVPs, as a collection of pairs,

$$\left\{ \begin{array}{ll} (\text{“/q/symbol”}, \text{“MSFT”}), & (\text{“/q/bid/price”}, 2328), \\ (\text{“/q/bid/size”}, 10000), & (\text{“/q/offer/price”}, 2355), \\ (\text{“/q/offer/size”}, 5000) & \end{array} \right\}$$

from the payload message is blinded and placed in the header of the notification. The notification for the above quote includes these blinded values and the encrypted quote.

[3]The additive inverse of a number $v \in \mathbb{Z}_m$ can be represented by the number $m - v$.

6.1.1 Interactions

We now present an overview of the protocols proposed in our CBPS system. The motivation behind constructing a set of protocols is that they can easily be implemented on top an existing CBPS infrastructure in order to satisfy privacy and security requirements. In summary, `Initialize` protocol initializes the system parameters. `Register` protocol registers Subs with Pubs. `Subscribe` protocol subscribes Subs to Brokers. `Publish` protocol publishes notifications from Pubs to Brokers. `Match` protocol matches notifications with subscriptions at Brokers. `Cover` protocol finds relationships among subscriptions at Brokers. An important property of the two most frequently used protocols, `Match` and `Cover`, is that they are non-interactive. The following gives more details of each protocol.

`Initialize`:
There is a set of system defined public parameters that all Pubs, Brokers and Subs use. In addition to these parameters, Pubs also generate some public and private parameters that are used for subsequent protocols and publish the public parameters. If there are several Pubs, each Pub generates its own public and private parameters.

`Register`:
Subs register themselves with the Pub to obtain a *secret* value and *access tokens*. An *access token* includes Sub's *identity* (id) and allows a Sub to subsequently authenticate itself to the Broker from which it intends to request notifications. An *identity* is a pseudonym that uniquely identifies a Sub in the system. The *secret* value allows a Sub to derive the key using the KeyDer algorithm of AB-GKM and then decrypt the payload of notifications.

`Subscribe`:
In order to assure confidentiality and privacy, unlike in a typical CBPS system, Subs

need to perform an additional communication step with Pub to get the subscription blinded before submitting the subscription to Broker [4].

After authenticating themselves using access tokens to Pubs, Subs receive the content in their subscriptions blinded by the corresponding Pubs. In this step, Subs perform as much computation as it can before sending the subscriptions to Pub so that the overhead on Pubs is minimized. Further, this overhead on Pubs is negligible as subscriptions are fairly stable and the rate of subscriptions is usually way less than that of notifications in a typical CBPS system. Once this step is done, Subs authenticate themselves to Brokers without revealing their identities and present these blinded subscriptions to Brokers. These subscriptions are blinded in such a way that Brokers do not learn the actual subscription criteria, that is, Brokers cannot decrypt the blinded values. However, they can perform `Match` (or `Filter`),[5] and `Cover` protocols based on the blinded subscriptions. Furthermore, no two subscriptions for the same value are distinguishable by Brokers. In order to prevent Brokers from linking different subscriptions from the same Sub, Subs may request for multiple access tokens such that all these access tokens have the same identity but are indistinguishable. For each subscription, Subs may present these different valid access tokens so that Subs' identities are further protected from Brokers.

`Publish`:

Using the counterparts of the secret values used to blind subscriptions, Pubs blind the notifications and publish them to some Brokers. A blinded notification has a set of blinded AVPs and an encrypted payload message. These notifications are blinded in such a way that Brokers do not learn actual values in the messages, but can perform `Match` and `Cover` protocols based on the subscriptions. Further, no two notifications for the same content are distinguishable by Brokers.

[4]Instead of Pub, a trusted third party may be utilized to blind subscriptions in order to reduce the load on Pub.

[5]We use the terms `Match` and `Filter` interchangeably.

`Match`:

For each notification from Pubs, Brokers compare it with Subs' subscriptions. If there is a match, that is, the subscription satisfies the notification, Brokers forward the notification to the correct Subs. The outcome of the `Match` protocol allows Brokers to learn neither the notification nor the publication values. It also prevents Brokers from learning the distribution of the values.

`Cover`:

For each subscription received from Subs, Brokers check if *covering* relationship holds with the existing subscriptions. A subscription S_1 covers another subscription S_2 if all notifications that match S_2 also match S_1. Finding covering relationships among subscriptions allows to reduce the size of the subscription tables maintained by each Broker, and hence improves the efficiency of matching. Like the `Match` protocol, the outcome of the `Cover` protocol does not allow the Brokers to learn the subscription values nor their distribution.

6.1.2 Trust Model

In the system design, we consider threats and assumptions from the point of view of Pubs and Subs with respect to third-party Brokers. We assume that Brokers are honest but curious; they perform PS protocols correctly, but curious to know what Pubs publish and Subs consume. In other words, they are trusted for these PS protocols but not for the content in the notifications and subscriptions nor for the privacy of Subs if they make one or more subscription requests. Further, Brokers may collude. Pubs are trusted to maintain the privacy of Subs. However, our approach can be easily modified to relax this trust assumption. Pubs are also trusted to correctly perform PS protocols and not to collude with any other parties.

6.2 Background

Some of the mathematical notions and the cryptographic building blocks which inspired our approach are described below.

6.2.1 Pedersen Commitment

A cryptographic "commitment" is a piece of information that allows one to commit to a value while keeping it hidden, and preserving the ability to reveal the value at a later time. The *Pedersen commitment* [47] is an unconditionally hiding and computationally binding commitment scheme which is based on the intractability of the discrete logarithm problem.

Pedersen Commitment

Setup A trusted third party T chooses a multiplicatively written finite cyclic group G of large prime order $\mathfrak{p}$ so that the computational Diffie-Hellman problem is hard in G.[6] T chooses two generators g and h of G such that it is hard to find the discrete logarithm of h with respect to g, i.e., an integer x such that $h = g^x$. It is not required that T know the secret number x. T publishes $(G, \mathfrak{p}, g, h)$ as the system parameters.

Commit The domain of committed values is the finite field $\mathbb{F}_\mathfrak{p}$ of $\mathfrak{p}$ elements, which can be represented as the set of integers $\mathbb{F}_\mathfrak{p} = \{0, 1, \ldots, \mathfrak{p} - 1\}$. For a party U to commit a value $\alpha \in \mathbb{F}_\mathfrak{p}$, U chooses $\beta \in \mathbb{F}_\mathfrak{p}$ at random, and computes the commitment $c = g^\alpha h^\beta \in G$.

Open U shows the values α and β to open a commitment c. The verifier checks whether $c = g^\alpha h^\beta$.

[6] For a multiplicatively written cyclic group G of order q, with a generator $g \in G$, the *Computational Diffie-Hellman problem (CDH)* is the following problem: Given g^a and g^b for randomly-chosen secret $a, b \in \{0, \ldots, q-1\}$, compute g^{ab}.

6.2.2 Zero-Knowledge Proof of Knowledge (Schnorr's Scheme)

The *zero-knowledge proof of knowledge (ZKPK)* protocol used in this paper can be viewed a natural extension of Schnorr's scheme [11]. In our proposed approach, we use ZKPK as a privacy-preserving means of subscriber authentication to the brokers.

As in the case of the Pedersen commitment scheme, a trusted party T generates public parameters $G, \mathfrak{p}, g, h$. A Prover which holds private knowledge of values α and β can convince a Verifier that Prover can open the Pedersen commitment $c = g^{\alpha}h^{\beta}$ as follows.

1. Prover randomly chooses $y, s \in \mathbb{F}_{\mathfrak{p}}^{*}$, and sends Verifier the element $d = g^{y}h^{s} \in G$.
2. Verifier picks a random value $e \in \mathbb{F}_{\mathfrak{p}}^{*}$, and sends e as a challenge to Prover.
3. Prover sends $u = y + e\alpha, v = s + e\beta$, both in $\mathbb{F}_{\mathfrak{p}}$, to Verifier.
4. Verifier accepts the proof if and only if $g^{u}h^{v} = d \cdot c^{e}$ in G.

6.2.3 Euler's Totient Function $\phi(\cdot)$ and Euler's Theorem

Let $\mathbb{Z}$ be the set of integers. Let $\mathbb{Z}^{+}$ denote all positive integers. Let $m \in \mathbb{Z}^{+}$. The *Euler's totient function* $\phi(m)$ is defined as the number of integers in $\mathbb{Z}^{+}$ less than or equal to m and relatively prime to m.

Theorem 6.2.1 (Euler's Theorem) *Let $m \in \mathbb{Z}^{+}$. If $\gcd(a, m) = 1$, then $a^{\phi(m)} \equiv 1 \pmod{m}$.*

6.2.4 Composite Square Root Problem

Definition 6.2.1 (Composite square root problem) *Let $n = pq$ be a product of two distinct large primes. The* composite square root problem *the computational problem defined as follows: given $w \in QR$, where $QR = \{y | y = x^{2} \pmod{n}, x \in \mathbb{Z}^{\times}\}$, compute $x \in \{1, 2, \ldots, n-1\}$ such that $w = x^{2} \pmod{n}$.*

It is well known that for each $w \in \mathsf{QR}$, there are four $x \in \{1, 2, \ldots, n-1\}$ such that $x^2 = w \pmod{n}$. If the prime factorization of n is known, then there are efficient algorithms to solve the above problem [59]. However, the problem seems difficult if the factorization of n is hard. In the construction of our CBPS system, we make use of the *composite square root assumption* which is based on this difficulty.

Conjecture 1 (Composite square root assumption) *There exists no polynomial time algorithm to solve the composite square root problem.*

6.2.5 Paillier Homomorphic Cryptosystem

The *Paillier homomorphic cryptosystem* is a public key cryptosystem by Paillier [10] based on the "Composite Residuosity assumption (CRA)." The Paillier cryptosystem is homomorphic in that, by using public key, the encryption of the sum $m_1 + m_2$ of two messages m_1 and m_2 can be computed from the encryption of m_1 and m_2. Our approach and protocols are inspired by how the Paillier cryptosystem works. Hence, we provide some internal details of the cryptosystem below so that readers can follow the rest of the paper.

Key generation

Set $n = pq$, where p and q are two large prime numbers. Set $\lambda = \mathrm{lcm}(p-1, q-1)$, i.e., the least common multiple of $p-1$ and $q-1$. Randomly select a base $g \in \mathbb{Z}/(n^2)^\times$ such that the order of g_p is a multiple of n. Such a g_p can be efficiently found by randomly choosing $g_p \in \mathbb{Z}/(n^2)^\times$, then verifying that

$$\gcd(L(g_p^\lambda \pmod{n^2}), n)) = 1, \text{ where } L(u) = (u-1)/n \tag{6.1}$$

for $u \in S_n = \{u < n^2 | u = 1 \pmod{n}\}$. In this case, set $\mu = \left(L(g_p^\lambda \pmod{n^2})\right)^{-1} \pmod{n}$. The public encryption key is a pair (n, g_p). The private decryption key is (λ, μ), or equivalently (p, q, μ).

Encryption $E(m, r)$

Given plaintext $m \in \{0, 1, \ldots, n-1\}$, select a random $r \in \{1, 2, \ldots, n-1\}$, and encrypt m as $E(m, r) = g_p^m \cdot r^n \pmod{n^2}$. When the value of r is not important to the context, we sometimes simply write a short-hand $E(m)$ instead of $E(m, r)$ for the Paillier ciphertext of m.

Decryption $D(c)$

Given ciphertext $c \in \mathbb{Z}/(n^2)^\times$, decrypt c as

$$D(c) = L(c^\lambda \pmod{n^2}) \cdot \mu \pmod{n}. \tag{6.2}$$

More specifically, the homomorphic properties of Paillier cryptosystem are:

$$\begin{aligned}
&D(E(m_1, r_1)E(m_2, r_2) \pmod{n^2}) = m_1 + m_2 \pmod{n},\\
&D(g^{m_2}E(m_1, r_1) \pmod{n^2}) = m_1 + m_2 \pmod{n},\\
&D(E(m_1, r_1)^k \pmod{n^2}) = km_1 \pmod{n}.
\end{aligned}$$

Also note that the Paillier cryptosystem described above is semantically secure against chosen-plaintext attacks (IND-CPA).

In the construction of our CBPS system, the Paillier homomorphic cryptosystem is used in a way that public and private keys are judiciously distributed among Pubs, Subs, and Brokers such that the confidentiality and privacy are assured based on homomorphic encryption. A detailed description of the construction is presented in Section 6.3.

6.3 Proposed Scheme

In this section, we provide a detailed description of the privacy preserving CBPS system we propose. As introduced in Section 6.1, the system consists of 6 protocols: 1) `Initialize`, 2) `Register`, 3) `Subscribe`, 4) `Publish`, 5) `Match`, and 6) `Cover`.

6.3.1 Initialize

A trusted party, which could be one of the Pubs, runs a Pedersen commitment setup algorithm [47] to generate system wide parameters $(G, \mathfrak{p}, g, h)$. These parameters have the same meaning and purpose as mentioned in Section 6.2. The same party also runs a key generation algorithm similar to Paillier [10] to generate the parameters $(n, p, q, g_p, \lambda, \mu)$. Only Pubs know the parameters (p, q, λ). The parameters (n, g_p, μ) are public. Note that unlike in Paillier, μ is public in our scheme. The system parameter l is the upper bound on the number of bits required to represent any data values published, and we refer to it as *domain size*. For example, if an attribute can take values from 0 up to 500 ($< 2^9$), l should be at least 9 bits long. For reasons that will soon become clear in this section we choose l such that $2^{2l} \ll n$.[7] In addition to these parameters, each Pub has a key pair (K_{pub}, K_{pri}) where K_{pri} is the private key used to sign access tokens of Subs and K_{pub} is the public key used by Brokers to verify authenticity and integrity of them. Each Pub also runs the Setup algorithm of the AB-GKM scheme to initialize the key management system for encrypting payload messages to Subs. Each Pub computes two pairs of secret values (e_m, d_m) and (e_c, d_c) such that $e_m + d_m \equiv 0 \pmod{\phi(n^2)}$, and $e_c + d_c \equiv 0 \pmod{\phi(n^2)}$, where $\phi(\cdot)$ is Euler's totient function and $e_m = e_c$. Note that we have $g^{e_m} g^{d_m} \equiv g^{e_c} g^{d_c} \equiv 1 \pmod{n^2}$. Pub uses e_m to blind Paillier encrypted notifications and d_m, d_c, e_c to blind Paillier encrypted subscriptions.[8] Let s be the largest number $\in \mathbb{Z}$ such that $2^s < n$ and $u \in \mathbb{Z}$ such that $l < u < s - 1$. Finally, each Pub chooses two secret random values $r_m, r_c \in \mathbb{Z}$ such that $1 < r_m, r_c < 2^{u-l}$ and $r_m = r_c$. These values are used to prevent Brokers from learning the distribution of the difference of the values that are being matched. In summary, $(G, \mathfrak{p}, g, h, n, g_p, \mu, K_{pub})$ are the public parameters that all the parties know, $(p, q, \lambda, K_{pri}, r_m, r_c, (e_m, d_m), (e_c, d_c))$ are private parameters of Pubs. Note that in a practical implementation, most of these parameters can be auto-

[7] We use notation $a \ll b$ to denote that "a is sufficiently smaller than b."

[8] The "blind" operation will be introduced in Section 6.3.3.

generated by a computer program which usually only requires Pub to pre-determine l depending on the domain of the content of notifications.

6.3.2 Register

As shown in Figure 6.2, each Sub registers itself with Pub by presenting an id (identity), a pseudonym uniquely identifying Sub. In a real-world system, registration may involve Subs presenting other credentials and/or making payment. Upon successful registration, Sub executes the SecGen algorithm of the AB-GKM scheme to obtain a secret s. We omit the details of the AB-GKM based key management as a detailed application of it is provided in the previous two chapters. During this protocol, each Sub also obtains its initial access token, a Pedersen commitment signed by Pub.

An access token allows Sub to authenticate itself to Broker from which it intends to request notifications as well as to create additional access tokens in consultation with Pub. To create the first access token, Sub encodes its id as an element $\langle \mathsf{id} \rangle \in \mathbb{F}_{\mathfrak{p}}$, chooses a random $a \in \mathbb{F}_{\mathfrak{p}}$, and sends the commitment $com(\langle \mathsf{id} \rangle) = g^{\langle \mathsf{id} \rangle} h^a$ and the values $(\langle \mathsf{id} \rangle, a)$. The Pub signs $com(\langle \mathsf{id} \rangle)$ and sends the digital signature $K_{pri}(com(\langle \mathsf{id} \rangle))$ back to the Sub.

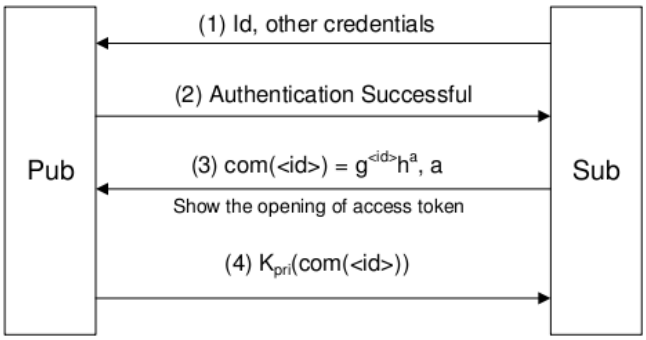

Figure 6.2.: Sub registering with Pub

6.3.3 Subscribe

During this protocol, Subs inform their interests to Brokers as subscriptions. Before subscribing to messages, as Figure 6.3 illustrates, Subs must authenticate themselves to Brokers. Sub gives a zero-knowledge proof of knowledge (ZKPK) of the ability to open the commitment $com(\langle \mathsf{id} \rangle)$ signed by Pub:

$$\text{ZKPK}\{(\langle \mathsf{id} \rangle, a) : com(\langle \mathsf{id} \rangle) = g^{\langle \mathsf{id} \rangle} h^a\}$$

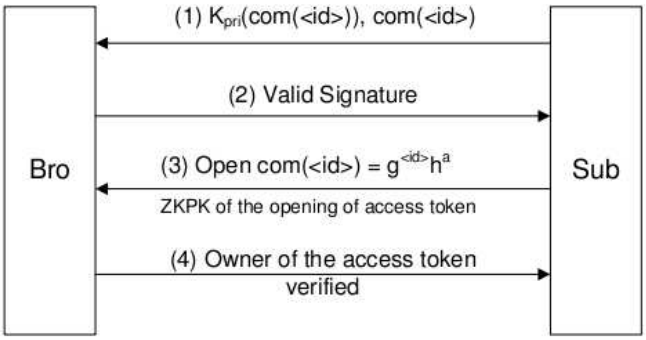

Figure 6.3.: Sub authenticating itself to Broker

Notice that the ZKPK of the commitment opening does not reveal the identity of Sub. Further, Sub may use different access tokens by having different random a values for different subscriptions to prevent Brokers from linking its subscriptions to one access token [9] [10].

If the ZKPK is successful, Sub may submit one or more subscriptions. Recall that subscriptions are blinded by Pub before sending to Broker. The subscription "blinding" functions, $bval_m$, $bval_{c_1}$, $bval_{c_2}$ are defined as follows:

[9] One may use a randomized signature scheme on a committed value [60] to achieve the same objective at the expense of additional computation cost.

[10] Our scheme only provides application level privacy, but not network level privacy. For example, it does not hide IP addresses. In order to provide network level privacy/anonymity, one needs to utilize other orthogonal techniques such as Tor [61]

Let v be the original subscription.

$$E(v) = g_p^v \cdot r_1^n \pmod{n^2}$$

$$bval_m(E(-v)) = g^{d_m} \cdot (E(-v))^{r_m \lambda} \pmod{n^2} \tag{6.3}$$

$$bval_{c_1}(E(-v)) = g^{d_c} \cdot (E(-v))^{r_c \lambda} \pmod{n^2} \tag{6.4}$$

$$bval_{c_2}(E(v)) = g^{e_c} \cdot (E(v))^{r_c \lambda} \cdot (E(r))^{\lambda} \pmod{n^2} \tag{6.5}$$

where $d_m, e_m, r_m, d_c, e_c, r_c$ are generated during `Initialize`, r in Formula 6.5 is a random number such that $r \leq min\{r_c, 2^{(s-1-u)}\}$.

Sub sends $E(v)$ and $E(-v)$, where v is the original subscription for the attribute $attr$, to Pub. Pub sends back the blinded subscription to Sub and Sub sends the tuple $\langle attr,\ bval_{c_1}(E(-v)),\ bval_{c_2}(E(v)),\ bval_m(E(-v)),\ op\rangle$ to Broker. The first two blinded values in the subscription are used by Broker for `Cover` protocol and the third one for `Match` protocol. Note that Sub performs these encryptions to reduced the load on Pubs. It should also be noted that equality filters in our protocols are treated as range filters preventing Brokers from distinguishing equality filters from range filters. For example, in order to subscribe for $v = 5$, Sub subscriber for a range filter where $v \leq 5$ and $v > 4$. Except for range filters, each subscription from the same Sub are treated as disjunctive conditions.

Example 7

Sub wants to get all the notifications with bid price less than 22. The subscription has the format ("/quote/bid/price", 346213, 152311, 453280, $<$) where the second and third parameters are the blind values of 22 and -22, respectively, for `Cover` protocol to use, and the fourth is the blinded value of -22 for `Match` protocol to use.

6.3.4 Publish

Using e_m, the counterpart of d_m which is used to blind subscriptions for `Match` protocol, and other private parameters, Pubs blind the notifications using the function $bval_n$ as defined below.

Let x be one value in the notification.

$$bval_n(x) = g^{e_m} \cdot (E(x))^{r_m\lambda} \cdot E(r)^{\lambda} \pmod{n^2}$$
$$= g^{e_m} \cdot E((r_m x + r)\lambda) \pmod{n^2},$$

where e_m and r_m are generated during `Initialize`, r is selected uniformly at random such that $r \leq min\{r_m, 2^{(s-1-u)}\}$.

Pubs publish the blinded notifications to Brokers. A notification has a set of blinded AVPs and an encrypted payload message. For an illustration purpose, let us assume these AVPs are numbered from 1 to t, where t is the number of attributes of the payload message M being considered. The blinded notification looks like $(\langle attr_1, bval_n(x_1)\rangle, \ldots, \langle attr_t, bval_n(x_t)\rangle)$, where $attr_i$ and x_i are the i^{th} attribute name and value respectively.

6.3.5 Match

For each notification from Pub, Broker compares it with Subs' subscriptions to make routing decisions. We explain the `Match` operation for one attribute in the message, but it can be naturally extended to perform on multiple attributes. If at least one of the attributes in the message matches, we say that the subscription matches the notification, and in this case Broker forwards the notification to the corresponding Subs. For range filters, the conjunction of two corresponding `Match` operations is taken.

Let the blinded values be $bval_n(x)$ and $bval_m(E(-v))$ that Broker has received from Pub and Sub, respectively, for an attribute $attr$ with subscription value being v and notification value being x. Broker computes the following value *diff* as follows.

$$diff = L(bval_n(x) \cdot bval_m(E(-v)) \pmod{n^2}) \cdot \mu \pmod{n},$$

where L, μ are public parameters derived from Paillier. Using the *diff*, Broker makes the matching decision based on Table 6.1.

Table 6.1: Matching decision

diff	Decision
$< n/2$	$x \geq v$
$> n/2$	$x < v$

Before we show that the above computation gives a *diff* equal to $r_m \cdot (x - v) + r$, we describe how `Match` protocol gives the correct matching decision while outputting a (controlled) random *diff* value to Broker. Recall that in `Initialize`, the domain of the input values is set to $0 \sim 2^l$. Therefore, $0 \leq x, v \leq 2^l$. Notice that the difference of any two values x and v is either between $0 \sim 2^l$ if the difference is positive, or between $(n-2^l) \sim n$ if the difference is negative. Also, notice that the range $2^l \sim (n-2^l)$ is not utilized. In order to randomize the difference, we take advantage of this unused range and multiply the actual difference with a random secret value r_m and add another random value r both selected by Pub. The idea behind r_m and r are to first expand $0 \sim 2^l$ range to $0 \sim 2^u$ and $(n - 2^l) \sim n$ to $n - 2^s \sim n - n_m$, and then expand them to $0 \sim n/2$ and $n/2 \sim n$ respectively. Thus the difference is randomized, yet it allows Broker to make correct matching decisions without resulting in false positives or negatives.

During `Match` protocol, Broker does not learn the content under comparison. This is achieved due to the fact that without knowing λ, Broker cannot perform decryption freely, but is forced to engage into the protocol described below. Not knowing the values r_m and r, Broker does not learn the exact difference of the two values under comparison as well.

The following shows the correctness of *diff*. Let

$$y = bval_n(x) \cdot bval_m(E(-v)) \pmod{n^2}.$$

$$
\begin{aligned}
y &= g^{e_m} \cdot (E((r_m x + r)\lambda) \cdot g^{d_m} \cdot (E(-v))^{r_m \lambda} \\
&\quad (\text{mod } n^2) \\
&= g^{e_m + d_m} \cdot \{E(r_m x + r)) \cdot E(-r_m v)\}^{\lambda} \quad (\text{mod } n^2) \\
&= (E(r_m(x - v) + r))^{\lambda} \quad (\text{mod } n^2) \\
\mathit{diff} &= L(y) \cdot \mu \quad (\text{mod } n) = r_m(x - v) + r. \qquad (6.6)
\end{aligned}
$$

6.3.6 Cover

Subscriptions are categorized into groups based on the covering relationships so that Brokers can perform `Match` protocol efficiently. For each subscription received from Subs, Brokers check if covering relationship holds within the existing subscriptions. If it exists, they add the new subscription to the group with the covering subscription, otherwise a new group is created for the new subscription.

Notice that we have not used the blinded values $bval_{c_1}(E(-v))$ and $bval_{c_2}(E(v))$ in subscriptions yet. These two values are used in the `Cover` protocol. In what follows, we explain how the `Cover` protocol works.

Let S_1 and S_2 be two subscriptions for the same *attr* and compatible *op*. Two *op*'s are compatible if either both of them are of the same type. $bval_{c_1}(E(v_1))$ and $bval_{c_2}(E(-v_1))$ refer to the so far unused blinded values of v_1 and of its additive inverse, respectively, of the subscription S_1. The blinded values $bval_{c_1}(E(v_2))$ and $bval_{c_2}(E(-v_2))$ have similar interpretations.

Broker computes one of the following two values in order to decide the covering relationship.

$$
\begin{aligned}
\mathit{diff}_1 &= L(bval_{c_2}(E(v_1)) \cdot bval_{c_1}(E(-v_2)) \\
&\quad (\text{mod } n^2)) \cdot \mu \quad (\text{mod } n) \\
\mathit{diff}_2 &= L(bval_{c_2}(E(v_2)) \cdot bval_{c_1}(E(-v_1)) \\
&\quad (\text{mod } n^2)) \cdot \mu \quad (\text{mod } n) \qquad (6.7)
\end{aligned}
$$

$diff_1$ and $diff_2$ give results $r_c \cdot (v_1 - v_2) + r$ and $r_c \cdot (v_2 - v_1) + r'$ respectively, where r, r' are random numbers. Broker uses the same matching Table 6.1 that is used for making matching decision to make the covering decision. The covering decision for range filters is performed in a similar way, but we omit the details due to lack of space. Similar to `Match`, Brokers do not learn the actual subscription values.

6.3.7 The Distribution of Load

We now briefly explain the rationale behind the distribution of work load among Pubs, Subs and Brokers. If there are $O(N)$ notifications and $O(S)$ subscriptions, in the worst case, Broker needs to perform $O(NS)$ `Match` protocols. Thus, Brokers have to perform significantly more work compared to Pubs and Subs in a typical CBPS system. This is one of the key reasons why the performance of Brokers degrades as the number of notifications and/or subscriptions in the system increases. By optimizing for the frequent case, one can achieve a significant overall system improvement. We followed this well-known design principle to redistribute the load on Brokers partly to Pubs and Subs. Notice that there are no exponentiation operations in both `Match` and `Cover` protocols. Hence, these protocols can be performed very efficiently. This is made possible at the cost of extra work at Pubs and Subs. Since the protocols at Pubs and Subs are executed less frequently compared to those at Brokers, our distribution leads to a better overall system performance. The experimental results show that the protocols at Brokers are very efficient and those at Pubs and Subs also run fast.

6.4 Experimental Results

In this section, we present experimental results for various operations and the two main protocols, `Match` and `Cover`, in our system as well as our privacy preserving CBPS (PP-CBPS) system itself which extends an enhanced SIENA system by implementing privacy preserving matching and covering using our protocols. For the protocol experiments, we have built a prototype system in Java that incorporates

our techniques for privacy preserving `Match` and `Cover` protocols as described in Section 6.3.

The experiments are performed on an Intel® Core™ 2 Duo CPU T9300 2.50GHz machine running GNU/Linux kernel version 2.6.27 with 4 Gbytes memory. We utilize only one processor for computation. The code is built with Java version 1.6.0. along with Bouncy Castle lightweight APIs [62] for most cryptographic operations including the symmetric-key encryption. The Paillier cryptosystem is implemented as in the paper [10], except that we modified the algorithms to fit our scheme. We first look at the experiments mainly on the two important protocols, `Match` and `Cover`, and then describe the system experiments performed on PP-CBPS system.

6.4.1 Protocol Experiments

Table 6.2: Average computation time for general operations

Computation	Time (in ms)
Create access token (Sub)	4.21
Open access token (Pub)	4.17
Sign access token (Pub)	4.10
Verify token signature (Broker)	0.36
ZKP of access token (Sub)	4.18
ZKP of access token (Broker)	6.31
Encrypt payload message (Pub)	34.56
Decrypt payload message (Sub)	0.36

In our experiments we vary values of n in Paillier cryptosystem and the domain size l, and fix the parameters for Pedersen commitment generation, digital signature generation/verification, zero-knowledge proof of knowledge protocol, and symmetric key encryption/decryption. In all our experiments we only measure computational

cost, and assume the communication cost to be negligible. All data obtained by our experiments correspond to the average time taken over 1000 executions of the protocols with varying values for the bit length of n in the Paillier cryptosystem and the domain size l. We first show the computation time for the general operations in order to provide a comparative assessment of our protocols.

We compare our protocol results with the well established computations to show that our approach is efficient and practical.

Table 6.2 shows the average running time for various operations for which we kept the system parameters constant. Access token creation, opening, signing are performed during `Register` protocol and based on Pedersen commitment scheme. Pub signs the access token using SHA-1 and RSA with 1024-bit long private key K_{pri}. Verification of the signature on the access token using the public key K_{pub}, and the ownership proof of the access token via the ZKPK are performed during `Subscribe` protocol. Zero-Knowledge Proof (ZKP) protocols are generally considered time consuming, but in our approach ZKP computation is comparable to other operations in the system, in that it takes merely a few milliseconds. For the experiments, we set the payload size to 4 Kbytes and used AES-128 as the symmetric key algorithm. These performance results demonstrate that the constructs we use and the computations are very efficient.

In the experiment shown in Figure 6.4, we vary the bit length of n in the Paillier cryptosystem. Figure 6.4 shows the time to generate blinded subscriptions and notifications whose values are less than 2^l where l, the domain size, is fixed at 100, a reasonably large value. The time to generate blinded values increases as the bit length of n increases, but even for large bit lengths, it takes only a few milliseconds. The time required to blind subscription is split into two tasks with the Sub performing the encryption and the Pub performing the blinding, but to blind notifications, the Pub performs both operations as one task. We remark that the overall computational cost can be reduced by employing well-known caching techniques.

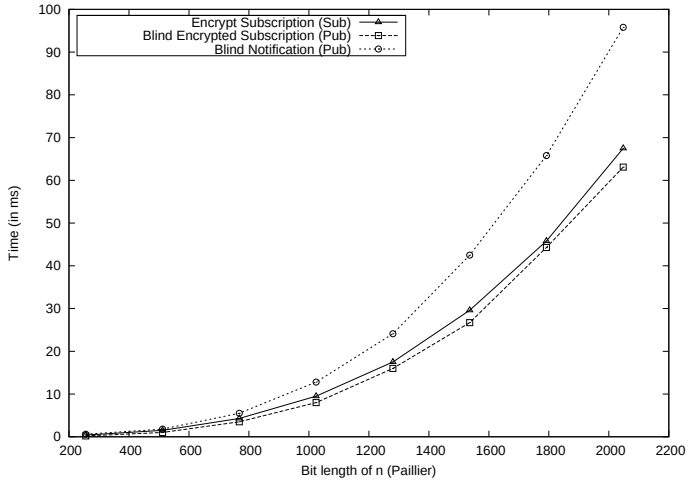

Figure 6.4.: Time to blind subscriptions/notifications for different bit lengths of n

We measure in our experiment the performance impact on blinding when l, the domain size, is changed. We fix n to be of length 1024 bits and measure the time to blind subscriptions and notifications for $l = 10, 20, \cdots, 100$. As shown in Figure 6.5, the domain size does not significantly affect the performance of the blinding operations. Further, as indicated by both Figure 6.4 and Figure 6.5, the time for either component of the subscription blinding is less than that for notification blinding. Since for each subscription, the overhead at the Pub is less compared to the time required to blind a notification, our decision to blind part of the subscription at the Pub is comparable to blinding additional notifications.

In a CBPS, `Match` is the most executed protocol. Hence, it should be very efficient so as not to overload Brokers. For each `Subscribe` protocol, Brokers may need to invoke the `Cover` protocol and, therefore, we want to have a very efficient `Cover` protocol as well. In the following two experiments, we observe the time to perform these protocols.

Figure 6.6 shows the execution time of `Match` and `Cover` protocols as the bit length of n in the Paillier cryptosystem is changed while the domain size l is fixed at 100 bits. The time for both protocols increases approximately linearly with the

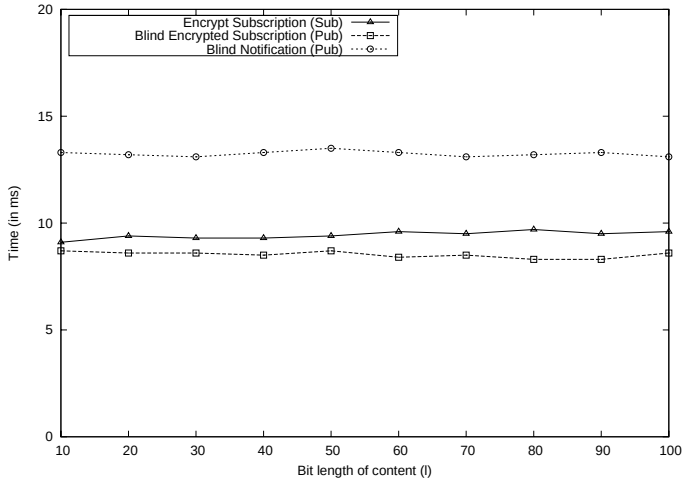

Figure 6.5.: Time to blind subscriptions/notifications for different l

bit length of n. Note that they take only a fraction of a millisecond (less than 100 microseconds) even for large bit lengths of n. This indicates that our `Match` and `Cover` protocols are very efficient for large bit lengths of n.

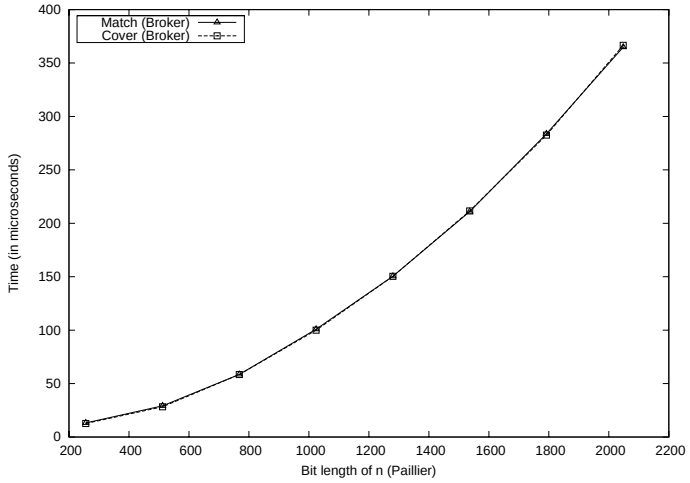

Figure 6.6.: Time to perform match/cover for different bit lengths of n

Figure 6.7 shows the time to execute `Match` and `Cover` protocols as the domain size l is changed while the bit length of n is fixed at 1024. Similar to the blind computations, computational times remain largely unchanged for different l values.

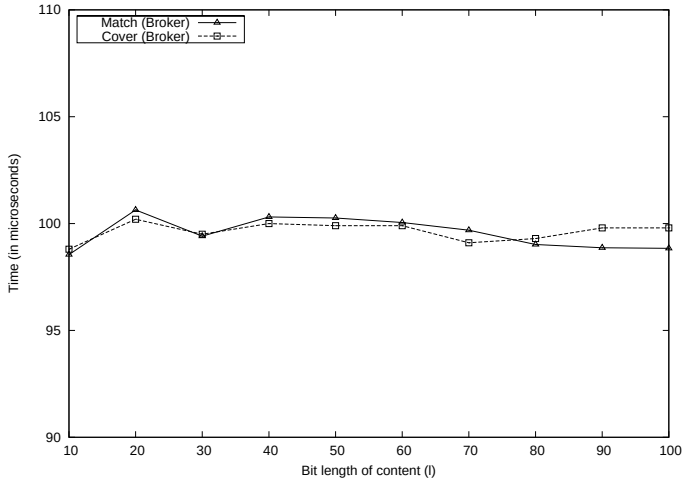

Figure 6.7.: Time to perform match/cover for different l

An observation made through all our protocol experiments is that the domain size l does not significantly affect the computational time of the key protocols `Publish`, `Subscribe`, `Match` and `Cover`, but the bit length n of the Paillier cryptosystem does. However, even for large bit lengths of n, our protocols take only a few microseconds or milliseconds and thus they are very efficient and practical.

6.4.2 System Experiments

In this section, we provide the experiments performed on our PP-CBPS system. PP-CBPS is constructed by a freely available popular wide-area event notification implementation SIENA. SIENA provides a pluggable-architecture that allows to incorporate our protocols to provide `Match` and `Cover` operations. All the testing data are generated uniformly at random. In all the experiments, the average time to match a notification with a subscription is measured where 1000 notifications are generated

each time and the system groups the subscriptions according to the covering relationships at the time of subscription. It should be noted that the matching time does not include the time to create notifications and subscriptions which is measured in our protocol experiments in Section 6.4.1.

Figure 6.8 shows the time to perform equality filtering in PP-CBPS (secure matching) and SIENA (plain matching) for different number of subscriptions in the system. Notifications and subscriptions are drown uniformly from 10 bit random integers. We use a small domain size to demonstrate the effect of covering on the overall system with and without security. As can be seen, PP-CBPS performs the matching within 10x of that of SIENA and is still quite efficient to match thousands of subscriptions within 10 ms. In both cases, the increase in matching time with the number of subscriptions is sub-linear since the covering operation groups the similar subscriptions together, reducing the number of `Match` protocols needs to be executed.

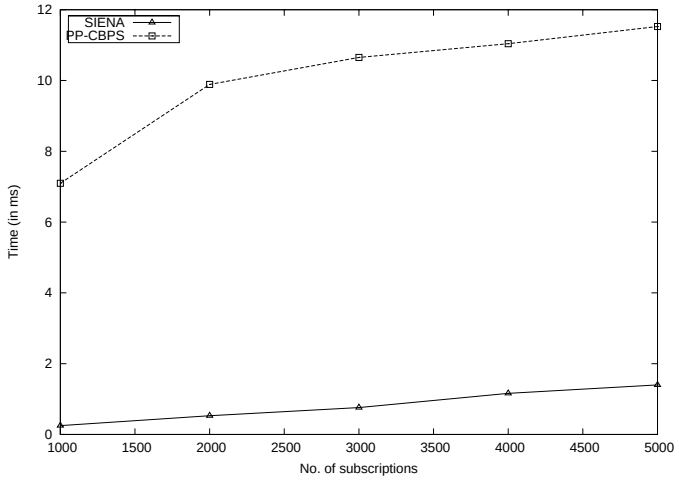

Figure 6.8.: Equality filtering time

Figure 6.9 shows the time to perform equality filtering in PP-CBPS for two different domain sizes, 10 and 25 bits, of notifications and subscriptions for different number of subscriptions in the system. It should be noted that SIENA currently does not support domain sizes larger than 27 bits, but our protocols can work under much

larger domains. As can be seen, the matching is more efficient with smaller domains. This is due to the fact that smaller domains create more covering relationships than larger domains and, hence, less matching protocols need to be executed to match a notification against all the subscriptions. Further, observe that the rate of increase of the overall matching cost decreases as the number of subscriptions increases. This, again, is due to the covering protocol.

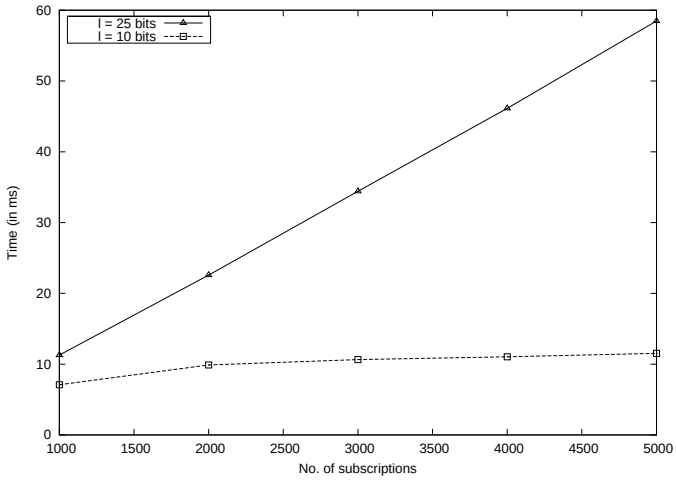

Figure 6.9.: Equality filtering time for different domain sizes

Figure 6.10 shows the time to perform inequality filtering in PP-CBPS for two different domain sizes, 10 and 25 bits, of notifications and subscriptions for different number of subscriptions in the system. We observe results similar to that of equality filtering in Figure 6.9. However, notice that the inequality filtering is much more efficient than equality filtering for the same domain size. This is due to the fact that inequality subscriptions create more covering relationships than equality subscriptions requiring much less matching operations.

Even though, according to the protocol experiments in Section 6.4.1, the time to perform individual `Match` or `Cover` operations remains largely constant for different domain sizes, the overall system performs better with smaller domain sizes. As the domain size is reduced, there is a higher probability of having subscriptions satis-

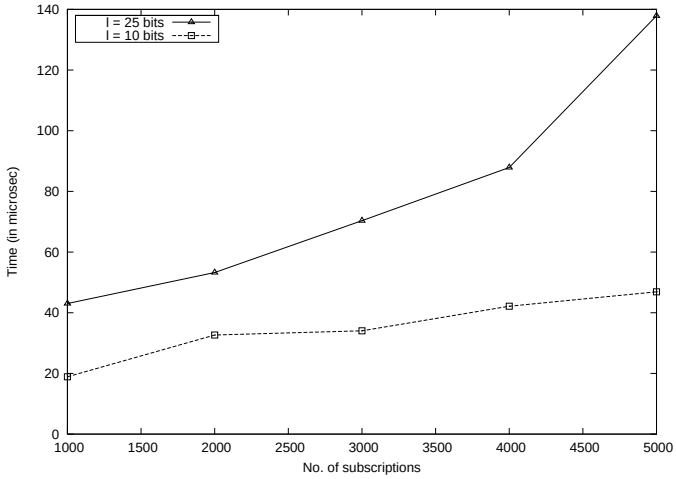

Figure 6.10.: Inequality filtering time for different domain sizes

fying covering relationships. Hence, the number of matching operations need to be performed reduces considerably leading to a better performance.

7 SURVEY OF RELATED WORK

Approaches closely related to our work have been investigated in different areas: group key management, functional encryption, selective publication of documents, secure data outsourcing, secret sharing schemes, proxy re-encryption systems, searchable encryption, secure multiparty computation, and private information retrieval. We compare our work with these areas below.

7.1 Group Key Management (GKM)

GKM is a widely investigated topic in the context of group-oriented multicast applications [15,28]. Early work on GKM relied on a key server to share a secret with users to distribute keys to decrypt documents [22,23]. Such approaches suffer from the drawback of sending $O(n)$ rekey information, where n is the number of users, in the event of join or leave to provide forward and backward secrecy. Hierarchical key management schemes [24,25], where the key server hierarchically establishes secure channels with different sub-groups instead of with individual users, were introduced to reduce this overhead. However, they only reduce the size of the rekey information to $O(\log n)$, and furthermore each user needs to manage at worst $O(\log n)$ hierarchically organized redundant keys. Similar to the spirit of our approach, there have been efforts to make rekey a one-off process [27,28]. It should be noted that the secure lock approach [26] based on the Chinese Remainder Theorem (CRT) is not a true broadcast key management scheme. Even though the session key can be updated with a single broadcast, the scheme still incurs $O(n)$ communication cost for rekeying. To the best of our knowledge, the approach based on *"n out of m"* secret sharing [29,30] proposed by Berkovits [27] is the first true broadcast scheme. The paper presents two variants. In both variants, each of the n users are given a secret share and

another $n + r$ (where $r > 0$) shares are given to all the users in the system. In other words, it creates a $n + r + 1$ out of $2n + r + 1$ secret sharing scheme. A valid user who has $n + r + 1$ shares can recover the secret, but others cannot. In the first variant, each user evaluates $n+r+1$ equations [29] whereas, in the second variant, the common $n + r$ shares are pre-evaluated and given only the results to reduce the load on users [30]. Both variants are correct, but it is not clear what security penalties proposed variants have due to certain assumptions made about the properties of secret shares. A recent research effort introduces a related BGKM approach based on access control polynomials [28]. This approach encodes secrets given to users at registration phase in a special polynomial of order at least n in such a way that users can derive the secret key from this polynomial. The special polynomials used in this approach represent only a small subset of domain of all the polynomials of order n, and the security of the approach is neither fully analyzed nor proven. Further, it appears that the security of the scheme weakens as n increases.

7.2 Functional Encryption

Functional encryption [63] is a popular public key cryptographic construct used to support fine-grained encryption on data. Functional encryption allows to encode an arbitrary complex access control policy with the encrypted message and allow to decrypt the message only for those satisfying the policy encoded. There are two subclasses of functional encryption: predicate encryption with public index [16, 64, 65] and predicate encryption without public index [66, 67].

In predicate encryption with public index schemes, the policy under which the encryption is performed is public. Unlike the public key cryptosystems, public is not a random string but some publicly known values that binds to users. The simplest scheme is called identity based encryption (IBE) where user identity (e.g. email address) is used as the public key. The idea of IBE was proposed by Shamir [68], but the first practical constructs proposed by Boneh and Cocks [64, 65]. Attribute

based encryption (ABE) is a more expressive predicate encryption with public index scheme. The concept of ABE, introduced by Sahai and Waters [16], can be considered as a generalization of IBE. In ABE, the public keys of a user is described by a set of identity attributes the user has. ABE has two popular variations: Key Policy ABE (KP-ABE) where encrypted documents are associated with attributes and user keys with policies [17]; Ciphertext Policy ABE (CP-ABE) where user keys are associated with attributes and encrypted documents with policies [18]. In either cases the cost of key management is minimized by using attributes that can be associated with users. Further, an ABE based approach supports expressive ACPs. However, such an approach suffers from some major drawbacks. Whenever the group dynamic changes, the rekeying operation requires to update the private keys given to existing members in order to provide backward/forward secrecy. This in turn requires establishing private communication channels with each group member which is not desirable in a large group setting.

In predicate encryption without pubic index schemes, the policy under which the encryption is performed is hidden from users. In other words, such schemes preserves the privacy of the access control policies. Anonymous IBE [69, 70], Hidden Vector Encryption [66], and Inner product predicate [67] are all fall under such schemes. Even though they preserve the privacy of the policy, they have limited expressibility compared to the former schemes and also suffer from the same limitations as the former approach. Our AB-GKM schemes address this limitation.

7.3 Selective Publishing of Documents

The database and security communities have carried out extensive research concerning techniques for the selective dissemination of documents based on access control policies [71–73]. These approaches fall in the following two categories.

1. Encryption of different subdocuments with different keys, which are provided to users at the registration phase, and broadcasting the encrypted subdocuments to all users [71, 72].
2. Selective multicast of different subdocuments to different user groups [73], where all subdocuments are encrypted with one symmetric encryption key.

The latter approaches assume that the users are honest and do not try to access the subdocuments to which they do not have access authorization. Therefore, these approaches provide neither backward nor forward key secrecy. In the former approaches, users are able to decrypt the subdocuments for which they have the keys. However, such approaches require all [71] or some [72] keys be distributed in advance during user registration phase. This requirement makes it difficult to assure forward and backward key secrecy when user groups are dynamic with frequent join and leave operations. Further, the rekey process is not transparent, thus shifting the burden of acquiring new keys on existing users when others leave or join. Having identified these problems, our preliminary work [20], proposes an approach to make rekey transparent to users by not distributing actual keys during the registration phase. However, the security of the approach is not analyzed and it cannot handle large user groups.

7.4 Secure Data Outsourcing

With the increasing utilization of cloud computing services, there has been a real need to access control the encrypted data stored in an untrusted third party. Our work falls into this category. There has been some recent research efforts [74, 75] to construct privacy preserving access control systems by combining oblivious transfer and anonymous credentials. The goal of such work is similar to ours but we identify the following limitations. Each transfer protocol allows one to access only one record from the database, whereas our approach does not have any limitation on the number of records that can be accessed at once since we separate the access control from the authorization. Another drawback is that the size of the encrypted database is not

constant with respect to the original database size. Redundant encryption of the same record is required to support ACPs involving disjunctions. However, our approach encrypts each data item only once as we have made the encryption independent of ACPs. Yu et al. [76] proposed an approach based on ABE utilizing PRE (Proxy Re-Encryption) to handle the revocation problem of ABE. While it solves the revocation problem to some extent, it does not preserve the privacy of the identity attributes as in our approach.

7.5 Secret Sharing Schemes

Secret sharing schemes split a shared secret among a group of users by giving secret shares to users and allow them to combine their secrets in a specific way and obtain the shared secret. Shamir [29] proposed the first secret sharing scheme, (n, k)-threshold scheme, where k users out of n can construct a unique polynomial $f(x)$ of degree $k-1$ and recover the shared secret $f(0)$. Since the definition of such scheme, several extensions have been proposed [30, 77, 78]. A major difference between GKM protocols and secret sharing schemes is that the former are designed to allow any individual group member to obtain a shared secret by itself, and no persistent secure communication channel is assumed between valid group members, whereas the latter are to prevent a single group member from gaining the secret alone, and require a secure communication channel, when group members combine the secret shares, to protect the shared secret from being learned by parties outside the group.

7.6 Proxy Re-Encryption Systems

In a proxy re-encryption system one party A delegates its decryption rights to another party B via a third party called a "proxy." More specifically, the proxy transforms a ciphertext computed under party A's public key into a different ciphertext which can be decrypted by party B with B's private key. In such a system neither the proxy nor party B alone can obtain the plaintext. A direct application of

the proxy re-encryption system does not solve the problem of CBPS: with the proxy as the Broker, it does not by default have the capability of selectively making content-based routing decisions. However, it might still be possible to use proxy re-encryption as a building block in the construction of a CBPS system for data confidentiality.

7.7 Searchable Encryption

Search in encrypted data is a privacy-preserving technique used in the *outsourced storage model* where a user's data are stored on a third-party server and encrypted using the user's public key. The user can use a query in the form of an encrypted token to retrieve relevant data from the server, whereas the server does not learn any more information about the query other than whether the returned data matches the search criteria. There have been efforts to support simple equality queries [79, 80] and more recently complex ones involving conjunctions and disjunctions of range queries [81]. These approaches cannot be applied directly to the CBPS model.

7.8 Secure Multiparty Computation (SMC)

SMC allows a set of participants to compute the value of a public function using their private values as input, but without revealing their individual private values to other participants. The problem was initially introduced by Yao.Since then improvements have been proposed to the initial problem [82,83]. SMC solutions rely on some form of zero-knowledge proof of knowledge (ZKPK) or oblivious transfer protocols which are in general interactive. Interactive protocols are not suitable for the CBPS model. Hence SMC solutions do not work for the CBPS model. Further, these solutions usually have a higher computational and/or communication cost which may not be acceptable for a CBPS system.

7.9 Private Information Retrieval (PIR)

A PIR scheme allows a client to retrieve an item from a database server without revealing which item is retrieved. Approaches of PIR assume either the server is computationally bounded, where the problem reduces to oblivious transfer, or there are multiple non-cooperating servers each having the same copy. Having only two communication parties, PIR schemes are not directly applicable to the Pub-Sub-Broker architecture of the CBPS model. Moreover, similar to SMC solutions, PIR schemes in general have a higher communication complexity which may not be acceptable for a CBPS system.

8 SUMMARY

In this dissertation, we defended our thesis that with novel group key management and cryptographic techniques we can construct privacy preserving fine grained access control on third party data management systems while assuring the confidentiality of data and preserving the privacy of users. We proposed solutions under two of the most popular dissemination models: pull based service model and subscription based publish-subscribe model. Having identified the drawbacks and issues in the existing key management systems for supporting privacy preserving attribute based access control, we first proposed a novel key management scheme called AB-GKM. Using the AB-GKM scheme along with existing and new cryptographic constructs, we constructed privacy preserving access control on both pull and subscription based models based on encryption.

While this dissertation provides an extensive investigation of privacy preserving access control for pull and subscription based dissemination systems, there are a number of problems and challenges that needs to be solved. We briefly look at some of them below:

Privacy preserving in the relational model:
Under the relational model, generally referred to as *Database-as-a-service* (DBaaS), the third party server provides a relational database to store data. With the popularity of third party services such as Amazon RDS and Microsoft SQL Azure there is a timely need to assure the confidentiality of sensitive data and the privacy of users while supporting relational functions. The challenge is to use encryption that enforces acps as well as allows to perform relational queries on encrypted data.

Content based access control in the pull model:

In the pull based model, we investigated mechanisms supporting only content independent ACPs. More expressive systems support access control based on both identity and content attributes. An example policy may look like "A doctor can access the data belonging to her patients only". Additional mechanisms are required to support content based ACPs while assuring the confidentiality of sensitive data and the privacy of users.

Providing accountability while preserving privacy:

Another important issue is how to build accountability in to third party dissemination systems while preserving the privacy. The problem is challenging as it involves the conflicting goals of privacy and traceability. In order to balance the privacy and accountability, we need new traitor tracing schemes. The solution to the problem should preserve the privacy of benign users (i.e. writes cannot be traced to the user who made them) as long as they follow the third party service provider's terms of use. However users should become traceable (i.e. an illegal write can be traced to a user) if they deviate from those terms of use. Previous research addressing this problem is very limited [84, 85]. Further, these approaches rely on a trusted third-party (TTP) which escrows the identity of the user to the service provider. For example, each user write is accompanied with the identity encrypted with TTP's public key. If the service provider finds an illegal write, it asks the TTP to escrow the identity by decrypting the message. In such a setting, users need to trust the TTP to reveal their identity to service provider only if their writes violate the terms of use and need to trust the service provider not to make false identity escrow requests to the TTP. Having a TTP (or a set of TTPs) is the ideal model and it is well known that relying on this ideal model is vulnerable if the above trust assumptions cease to hold (for example, one of the parties is controlled by an adversary). Answers need to be found to the questions "How to identify a breach of terms of use and encode it as a well-defined rule?" and "How to preserve the privacy of good users while providing accountability?".

Exploiting the relationship among acps/attribute conditions:
In many systems, acps and attribute conditions exhibit partial order relationships. For example, hierarchical policies are used in many domains. The most common example of such hierarchies is Role Based Access Control (RBAC) models [86]. Our AB-GKM scheme does not consider relationship among acps or attribute conditions. Due to the non-linear cost associated with KeyGen algorithm of AB-GKM, one can improve the efficiency of KeyGen by breaking the problem into a set of smaller problems and using the relationship among ACPs to derive keys. It is challenging to exploit the relationships among ACPs while preserving the privacy of users.

Privacy preserving access control on big data systems:
Big Data technologies such as Apache Hadoop are increasingly being used to store and/or analyze sensitive data. In order to comply with various regulations and organizational policies, such data needs to be stored encrypted and the access to them needs to be controlled based on the identity attributes of users. However, most of the existing third party systems utilizing traditional key management schemes provide either no or limited assurance of confidentiality and privacy. The challenge is to handle large volume of data and many users in an efficient manner while assuring the confidentiality of data and preserving the privacy of users who use such services.

LIST OF REFERENCES

LIST OF REFERENCES

[1] Liberty Alliance. `http://www.projectliberty.org/` [Last accessed: July 18, 2012].

[2] OpenID. `http://openid.net/` [Last accessed: July 18, 2012].

[3] Microsoft Windows CardSpace. `http://windows.microsoft.com/en-us/windows-vista/Windows-CardSpace` [Last accessed: July 18, 2012].

[4] Higgins Open Source Identity Framework. `http://www.eclipse.org/higgins/` [Last accessed: July 18, 2012].

[5] R. Richardson. CSI Computer Crime and Security Survey. Technical report, Computer Security Institute, 2008.

[6] W. Chenxi, A. Carzaniga, D. Evans, and A.L. Wolf. Security issues and requirements for internet-scale publish-subscribe systems. In *HICSS 2002: Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, pages 3940–3947, Jan 2002.

[7] E. Bertino, B. Carminati, E. Ferrari, B. Thuraisingham, and A. Gupta. Selective and authentic third-party distribution of XML documents. *IEEE Transactions on Knowledge and Data Engineering*, 16(10):1263–1278, Oct. 2004.

[8] M. Srivatsa and L. Liu. Securing publish-subscribe overlay services with eventguard. In *CCS 2005: Proceedings of the 12th ACM conference on Computer and Communications Security*, pages 289–298, 2005.

[9] M. Nabeel and E. Bertino. Secure delta-publishing of XML content. In *ICDE, 2008. Proceedings of the IEEE 24th International Conference on Data Engineering*, pages 1361–1363, Apr 2008.

[10] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT 1999: Proceeding of the 18th International Conference on the Theory and Application of Cryptographic Techniques*, pages 223–238, 1999.

[11] C.P. Schnorr. Efficient identification and signatures for smart cards. In *Proceedings of the 8th CRYPTO Conference on Advances in Cryptology*, pages 239–252, 1989.

[12] C. Raiciu and D. S. Rosenblum. Enabling confidentiality in content-based publish/subscribe infrastructures. In *Proceedings of the Securecomm and Workshops*, pages 1–11, 2006.

[13] M. Srivatsa and L. Liu. Secure event dissemination in publish-subscribe networks. In *ICDCS 2007: Proceedings of the 27th International Conference on Distributed Computing Systems*, pages 22–33, 2007.

[14] K. Minami, A. J. Lee, M. Winslett, and N. Borisov. Secure aggregation in a publish-subscribe system. In *WPES 2008: Proceedings of the 7th ACM workshop on Privacy in the electronic society*, pages 95–104, 2008.

[15] Y. Challal and H. Seba. Group key management protocols: A novel taxonomy. *International Journal of Information Technology*, 2(2):105–118, 2006.

[16] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *EUROCRYPT 2005: Procedings of the 25th Annual International Cryptology Conference on Advances in Cryptology*, pages 457–473, 2005.

[17] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *CCS 2006: Proceedings of the 13th ACM Conference on Computer and Communications Security*, pages 89–98, 2006.

[18] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *SP 2007: Proceedings of the 28th IEEE Symposium on Security and Privacy*, pages 321–334, 2007.

[19] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Transaction on Computer Systems*, 19(3):332–383, 2001.

[20] N. Shang, M. Nabeel, F. Paci, and E. Bertino. A privacy-preserving approach to policy-based content dissemination. In *ICDE 2010: Proceedings of the 2010 IEEE 26th International Conference on Data Engineering*, 2010.

[21] M. Nabeel, N. Shang, and E. Bertino. Privacy preserving policy based content sharing in public clouds. *IEEE Transactions on Knowledge and Data Engineering*, 2012.

[22] H. Harney and C. Muckenhirn. Group key management protocol (GKMP) specification. Technical report, Network Working Group, United States, 1997.

[23] H. Chu, L. Qiao, K. Nahrstedt, H. Wang, and R. Jain. A secure multicast protocol with copyright protection. *SIGCOMM Computer Communication Review*, 32(2):42–60, 2002.

[24] C.K. Wong and S.S. Lam. Keystone: A group key management service. In *ICT 2000: Proceedings of the International Conference on Telecommunications*, 2000.

[25] A.T. Sherman and D.A. McGrew. Key establishment in large dynamic groups using one-way function trees. *IEEE Transactions on Software Engineering*, 29(5):444–458, May 2003.

[26] G. Chiou and W. Chen. Secure broadcasting using the secure lock. *Software Engineering, IEEE Transactions on*, 15(8):929–934, Aug 1989.

[27] S. Berkovits. How to broadcast a secret. In *EUROCRYPT 1991: Proceedings of the 10th annual international conference on Advances in Cryptology*, pages 535–541, 1991.

[28] X. Zou, Y. Dai, and E. Bertino. A practical and flexible key management mechanism for trusted collaborative computing. In *INFOCOM 2008: The 27th Conference on Computer Communications*, pages 538–546, 2008.

[29] A. Shamir. How to share a secret. *ACM Communications*, 22(11):612–613, 1979.

[30] E. F. Brickell. Some ideal secret sharing schemes. In *EUROCRYPT 1989: Proceedings of the workshop on the theory and application of cryptographic techniques on Advances in cryptology*, pages 468–475, 1990.

[31] O. Goldreich. *Foundations of cryptography: Basic tools*. Cambridge University Press, New York, NY, USA, 2000.

[32] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS 1993: Proceedings of the 1st ACM conference on Computer and communications security*, pages 62–73, 1993.

[33] S Goldwasser, S Micali, and C Rackoff. The knowledge complexity of interactive proof-systems. In *STOC 1985: Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 291–304, 1985.

[34] D. Dummit and R. Foote. Gaussian-Jordan elimination. In *Abstract Algebra*, page 404. Wiley, 2nd edition, 1999.

[35] D. Naor, M. Naor, and J. B. Lotspiech. Revocation and tracing schemes for stateless receivers. In *CRYPTO 2001: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pages 41–62, 2001.

[36] D. Halevy and A. Shamir. The LSD broadcast encryption scheme. In *CRYPTO 2001: Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, pages 47–60, 2002.

[37] V. Shoup. NTL library for doing number theory. `http://www.shoup.net/ntl/` [Last accessed: July 18, 2012].

[38] OpenSSL the open source toolkit for SSL/TLS. `http://www.openssl.org/` [Last accessed: July 18, 2012].

[39] N. Shang, M. Nabeel, E. Bertino, and X. Zou. Broadcast group key management with access control vectors. Technical report, Department of Computer Science, Apr 2010.

[40] M. Nabeel and E. Bertino. Attribute based group key management. Technical Report CERIAS TR 2010, Purdue University, 2010.

[41] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters. Secure attribute-based systems. In *CCS 2006: Proceedings of the 13th ACM Conference on Computer and Communications Security*, pages 99–112, 2006.

[42] XML in clinical research and healthcare industries. `http://xml.coverpages.org/healthcare.html` [Last accessed: July 18, 2012].

[43] M. Eichelberg, T. Aden, J. Riesmeier, A. Dogac, and G. B. Laleci. A survey and analysis of electronic healthcare record standards. *ACM Computer Survey*, 37(4):277–315, 2005.

[44] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext policy attribute based encryption library. `http://http://acsc.cs.utexas.edu/cpabe/` [Last accessed: July 18, 2012].

[45] B. Lynn. Pairing based cryptography library. `http://crypto.stanford.edu/pbc/` [Last accessed: July 18, 2012].

[46] J. Li and N. Li. OACerts: Oblivious attribute certificates. *IEEE Transactions on Dependable and Secure Computing*, 3(4):340–352, 2006.

[47] T.P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO 1991: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, pages 129–140, 1992.

[48] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal on Uncertainity Fuzziness Knowledge-Based Systems*, 10(5):557–570, 2002.

[49] Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Incremental cryptography: The case of hashing and signing. In *Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology*, pages 216–233, 1994.

[50] Enrico Buonanno, Jonathan Katz, and Moti Yung. Incremental unforgeable encryption. In *FSE 2001: Revised Papers from the 8th International Workshop on Fast Software Encryption*, pages 109–124, 2001.

[51] N. Shang. G2HEC: A Genus 2 Crypto C++ Library. `http://www.math.purdue.edu/~nshang/libg2hec.html` [Last accessed: July 18, 2012].

[52] F. Paci, N. Shang, E. Bertino, K. Steuer Jr., and J. Woo. Secure transactions' receipts management on mobile devices. In *Symposium on Identity and Trust on the Internet (IDtrust Symposiums)*, Apr 2009.

[53] P. Gaudry and É. Schost. Construction of secure random curves of genus 2 over prime fields. In *EUROCRYPT 2004: Advances in Cryptology*, pages 239–256, 2004.

[54] Boolstuff, A boolean expression tree toolkit. `http://sarrazip.com/dev/boolstuff.html` [Last accessed: July 18, 2012].

[55] A. Schaad, J. Moffett, and J. Jacob. The role-based access control system of a european bank: a case study and discussion. In *SACMAT 2001: Proceedings of the sixth ACM symposium on Access control models and technologies*, pages 3–9, 2001.

[56] K. Fisler, S. Krishnamurthi, L. A. Meyerovich, and M. C. Tschantz. Verification and change-impact analysis of access-control policies. In *ICSE 2005: Proceedings of the 27th international conference on Software engineering*, pages 196–205, 2005.

[57] P. Eugster, P.A. Felber, R. Guerraoui, and A. Kermarrec. The many faces of publish/subscribe. *ACM Computing Survey*, 35(2):114–131, 2003.

[58] S. Choi, G. Ghinita, and E. Bertino. A privacy-enhancing content-based publish/subscribe system using scalar product preserving transformations. In *DEXA 2010: Proceedings of the 21st Conference on Database and Expert Systems Applications*, 2010.

[59] H. Cohen. *A course in computational algebraic number theory*, chapter 1.5, pages 31–36. Springer-Verlag, 1993.

[60] J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Proceedings of the 23rd CRYPTO Conference on Advances in Cryptology*, pages 56–72, 2004.

[61] Roger D., Nick M., and Paul S. Tor: The second-generation onion router. In *USENIX 2004: In Proceedings of the 13th Usenix Security Symposium*, 2004.

[62] Bouncycastle. Bouncy Castle Crypto APIs. `http://www.bouncycastle.org/` [Last accessed: July 18, 2012].

[63] D. Boneh, A. Sahai, and B. Waters. Functional encryption: definitions and challenges. In *TCC 2011: Proceedings of the 8th conference on Theory of cryptography*, pages 253–273, 2011.

[64] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO 2001: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pages 213–229, 2001.

[65] C. Cocks. An identity based encryption scheme based on quadratic residues. In *Proceedings of the 8th IMA International Conference on Cryptography and Coding*, pages 360–363, 2001.

[66] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC 2007: Proceedings of the 4th conference on Theory of cryptography*, pages 535–554, 2007.

[67] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT 2008: Proceedings of the theory and applications of cryptographic techniques 27th annual international conference on Advances in cryptology*, pages 146–162, 2008.

[68] A. Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 47–53, 1985.

[69] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. *Jurnal of Cryptology*, 21(3):350–391, March 2008.

[70] C. Gu, Y. Zhu, and H. Pan. Information security and cryptology. In Dingyi Pei, Moti Yung, Dongdai Lin, and Chuankun Wu, editors, *Inscrypt*, chapter Efficient Public Key Encryption with Keyword Search Schemes from Pairings, pages 372–383. 2008.

[71] E. Bertino and E. Ferrari. Secure and selective dissemination of XML documents. *ACM Transaction Information System Security*, 5(3):290–331, 2002.

[72] G. Miklau and D. Suciu. Controlling access to published data using cryptography. In *VLDB '2003: Proceedings of the 29th international conference on Very large data bases*, pages 898–909. VLDB Endowment, 2003.

[73] A. Kundu and E. Bertino. Structural signatures for tree data structures. *Proceeding of VLDB Endowment*, 1(1):138–150, 2008.

[74] S. Coull, M. Green, and S. Hohenberger. Controlling access to an oblivious database using stateful anonymous credentials. In *Irvine: Proceedings of the 12th International Conference on Practice and Theory in Public Key Cryptography*, pages 501–520, 2009.

[75] J. Camenisch, M. Dubovitskaya, and G. Neven. Oblivious transfer with access control. In *CCS 2009: Proceedings of the 16th ACM conference on Computer and communications security*, pages 131–140, 2009.

[76] S. Yu, C. Wang, K. Ren, and W. Lou. Attribute based data sharing with attribute revocation. In *ASIACCS 2010: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, pages 261–270, 2010.

[77] J.C. Benaloh and J. Leichter. Generalized secret sharing and monotone functions. In *CRYPTO 1988: Proceedings of the 8th Annual International Cryptology Conference on Advances in Cryptology*, pages 27–35, 1990.

[78] T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO 1991: Proceeding of 1991 CRYPTO Conference on Advances in Cryptology*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140, 1992.

[79] D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *SP 2000: Proceedings of the 2000 IEEE Symposium on Security and Privacy*, pages 44–55, 2000.

[80] D. Boneh, G. Crescenzo, R. Ostrovsky, and G. Persiano. Public-key encryption with keyword search. In *EUROCRYPT 2004: Proceedings of the 2004 EUROCRYPT on Advances in Cryptology*, 2004.

[81] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. *Theory of Cryptography*, pages 535–554, May 2007.

[82] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *EUROCRYPT 2004: Proceeding of the 2004 EUROCRYPT Conference on Advances in Cryptology*, 2004.

[83] I. Damgård, M. Geisler, and M. Kroigard. Homomorphic encryption and secure comparison. *International Journal on Applied Cryptology*, 1(1):22–31, 2008.

[84] L. Buttyán and J. Hubaux. Accountable anonymous access to services in mobile communication systems. In *SRDS 1999: Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems*, pages 384–394, 1999.

[85] M. Backes, J. Camenisch, and D. Sommer. Anonymous yet accountable access control. In *WPES 2005: Proceedings of the 4th ACM Workshop on Privacy in the Electronic Society*, 2005.

[86] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.

VITA

VITA

Contact Information

Department of Computer Science
Purdue University
305 N. University St., W. Lafayette, Indiana 47907

Voice: 765-337-2645 (Mobile)
Email: nabeel(at)cs.purdue.edu
http://www.cs.purdue.edu/ nabeel

Research/Development Interests

Data privacy, Context-aware security, distributed systems & security, database systems & security, information security and applied cryptography in general

Education

Purdue University, West Lafayette, IN, Aug. 2008 - Aug. 2012
Ph.D. in Computer Science
Advisor: Elisa Bertino
Dissertation: Privacy Preserving Access Control for Third-Party Data Management Systems

Purdue University, West Lafayette, IN, Aug. 2006 - May 2008
M.S. in Computer Science, GPA: 3.8/4.0
Advisor: Elisa Bertino

University of Moratuwa, Moratuwa, Sri Lanka, Feb. 2000 - Mar. 2004
B.Sc. with Honors in Computer Science & Engineering, GPA: 4.0/4.0, Rank:1/500

Honors and Awards

- Recipient of **Purdue Research Foundation grant**. 2011 - 2012
- Recipient of **Purdue Cyber Center research grant**. 2010 - 2011
- **Fulbright Fellow** at Purdue University. 2006 - 2008
- **PHP PECL Axis2/C Committer**, (Later the project was moved to wso2.org). 2006
- **Apache Committer** for the Axis2/C project. 2006
- **UNESCO Team Gold Medal Award** for the Highest Class Average in B.Sc. Engineering. 2004
- **TP De S Munasinghe Award** for the Highest Class Average in B.Sc. Computer Sci. & Eng. 2004
- **Silver Medal**, National Best Quality Software Competition, Sri Lanka. 2005
- **Gold Medal** Award for the All-Island Highest Aggregate (rank = 1) in Mathematics Stream, G.C.E A/L, Sri Lanka. 1998

Publications

Conference Publications

1. Mohamed Nabeel, Elisa Bertino, Privacy Preserving Delegated Access Control in the Data-as-a-Service Model. In *IEEE International Conference on Information Reuse and Integration (IRI), 2012.*
2. Mohamed Nabeel, Ning Shang, Elisa Bertino, Efficient Privacy-Preserving Publish Subscribe Systems. In *ACM Symposium on Access Control Models and Technologies (SACMAT), 2012.*
3. Mohamed Nabeel, David Stork, Oblivious Tree-based Classification in the Cloud. *Under Review.*

4. Mohamed Nabeel, Elisa Bertino, Murat Kantarcioglu, Bhavani Thuraisingham, Towards Privacy Preserving Access Control in the Cloud. In *IEEE International Conference on Collaborative Computing (CollaborateCom), 2011.*
5. Mohamed Nabeel, Elisa Bertino, Towards Attribute Based Group Key Management. In *ACM Conference on Computer and Communication Security (CCS), 2011* (Poster paper).
6. Mohamed Nabeel, Ning Shang, John Zage, Elisa Bertino, Mask: A System for Privacy-Preserving Policy-Based Access to Published Content. In *ACM International Conference on Management of Data (SIGMOD), 2010* (Demo paper).
7. Ning Shang, Mohamed Nabeel, Federica Paci, Elisa Bertino, A Privacy-Preserving Approach to Policy-Based Content Dissemination. *International Conference on Data Engineering (ICDE), 2010.*
8. Mohamed Nabeel, Elisa Bertino, Secure Delta-Publishing of XML Content. In *International Conference on Data Engineering (ICDE), 2008* (Poster paper).

Journal Publications

1. Mohamed Nabeel, Elisa Bertino, Privacy Preserving Delegated Access Control in the Cloud. Under Review In *IEEE Transaction on Knowledge and Data Engineering (TKDE).*
2. Mohamed Nabeel, Elisa Bertino, Attribute Based Group Key Management. Under Review In *IEEE Transactions on Dependable and Secure Computing (TDSC).*
3. Mohamed Nabeel, Elisa Bertino, Privacy Preserving Policy Based Content Sharing in Public Clouds. Under Review In *IEEE Transaction on Knowledge and Data Engineering (TKDE).*

Projects

Secure Advanced Metering Infrastructure Project Current

An industry collaborated project to secure the communication links in Advanced Metering Infrastructure (AMI).

CloudMask Project Current

A research project to build a privacy preserving cloud based storage/data service that protects the privacy of the users who access the service as well as the data stored in the cloud.

Ionomics Atlas 2011

Ionomics Atlas is a research project that provides a Google map based interface to find relationship among ionomic, genetic and environmental information for Arabidopsis Thaliana plant population. It is available to the public at `http://ibnkhaldun.cs.purdue.edu:8348/ionomicsatlas/`.

Mask Project 2010

A research project to build the first system addressing the seemingly-unsolvable problem of how to selectively share contents among a group of users based on access control policies expressed as conditions against the identity attributes of these users while at the same time assuring the privacy of these identity attributes from the content publisher. (C/C++/Java/Abstract Algebra)

Cancer Care Engineering Project 2010

A research project to model cancer-care systems, build educational tools and an interactive community. Have been involved in the project as a research assistant to build certain components of the project.

Smart Pump Informatics 2009

A research project to mine patterns in sensitive information collected from infusion devices (smart pumps) installed in different hospitals. Involved in the project as a summer intern 2009 to build certain components.

An Efficient Group Key Management (GKM) Scheme 2009

Designed and Implemented a new GKM scheme which is efficient and secure under frequent join and leave operations. C/C++, NTL library for implementing a novel GKM scheme, OpenSSL for cryptographic functions. Developed as part of a research paper for ICDE 2010.

A Scalable Routing Protocol to Distribute Hierarchically Organized Data 2008 - 2009

Designed and implemented a complete system which introduces the novel concept of hierarchically organized routing tables. Java, XML and related technologies, overlay networks. Developed as part of a research paper for DocEng 2009.

Secure Delta-Publishing of XML Documents 2008

Designed and implemented a complete Publish-Subscribe system to incrementally disseminate XML documents while preserving confidentiality and integrity. Java, XML and related technologies including XML encryption and digital signatures, overlay networks. Developed as part of the ICDE 2008 conference paper.

Apache Axis2/C 2006 - 2007

A high-performance open source Web Services middleware in C. Was part of the team in 2006 (Earned the Apache committership for my work). C, Web Services standards, middleware.

WSO2 WSF/PHP 2006

A high-performance open source Web Services middleware for PHP built on Apache Axis2/C in C. Initiated the project in early 2006 and was part of the team in 2006. C, PHP, Web Services standards, middleware, extension development for PHP.

Electronic Trading System 2004 - 2005

Responsible for design and development of several components of a trading system which is deployed in multiple high-profile stock exchanges. C/C++, various data specification used to disseminate trades and quotes, trading business logic.

PHPlus Web Application Development Framework 2004

A PHP based web application development framework which allow to design and develop application logic in parallel. Developed as an undergraduate research project and was part of the team of 4 in 2004. C/C++, PHP, XML, HTML, framework development.

Work Experience

Purdue University, West Lafayette, IN, USA.

Research Assistant Aug. 2011 - Present

Have been involved in projects on privacy preserving group key management, secure and privacy preserving cloud storage services, and privacy preserving publish subscribe systems.

Rosen Center for Advanced Computing, West Lafayette, IN, USA.

Research & Development Intern. May 2011 - Aug. 2011

Involved in devising policies to make a healthcare project HIPAA complaint and implementing the policies for the project. Also involved in research projects to analyze efficiency of the electric vehicles in Indiana and analyze recent earthquakes in Chile.

Cyber Center, West Lafayette, IN, USA.

Graduate Research Assistant Aug. 2010 - May 2011

Designed and developed a web-based system to find correlations among ionomic, genetic and environmental information of plant populations. The system is available at http://ibnkhaldun.cs.purdue.edu:8348/ionomicsatlas/.

Ricoh Innovations Inc., Menlo Park, CA, USA.

Research & Development Intern. May 2010 - Aug. 2010

Designed and developed techniques and complete systems to obliviously perform classification of data on an untrusted remote third-party server.

Rosen Center for Advanced Computing, West Lafayette, IN, USA.

Graduate Research Assistant Aug. 2009 - May 2010

Involved in a health-care research project called *ccehub.org*, the goal of which is to model cancer-care systems, build educational tools and an interactive community.

Rosen Center for Advanced Computing, West Lafayette, IN, USA.

Research & Development Intern. May 2009 - Aug. 2009

Involved in a health-care research project called *Smart Pump Informatics* to mine patterns in sensitive information collected from infusion devices (smart pumps) installed in different hospitals in Indiana.

Purdue University, West Lafayette, IN, USA.

Teaching Assistant Aug. 2008 - May 2009

Conducted labs, designed assignments and graded assignments for the courses CS 426 (Computer Security), CS 251 (Data Structures & Algorithms), and CS 541 (Database Management Systems) under different instructors.

Purdue University, West Lafayette, IN, USA.

Research Assistant May 2008 - Aug. 2008

Conducted research to find an efficient and scalable approach to selectively disseminate portions of XML documents to different users confirming to access control policies. Developed a prototype to demonstrate the approach.

WSO2 Inc., Colombo, Sri Lanka.

Senior Software Engineer Jan. 2006 - Jul. 2006

Actively participated in the development of popular open source Apache Axis2/C Web Services engine. Earned Apache committership for my work. Initiated the project of PHP Web Services (WSF/PHP).

Millennium Information Technologies Inc., Malabe, Sri Lanka.

Software Engineer Mar 2004 - Dec. 2005

Actively participated in the design and development of back-end software for international capital markets. Was mainly responsible for designing and developing external feed gateways which need to handle high volume of data and very high data rates. Guided several employees in this area.

Colombo University, Colombo, Sri Lanka.

Part-time Instructor Mar 2004 - Dec. 2005

Taught undergraduate level courses Network & System Administration and Object Oriented Programming for an external bachelor's program by Colombo university for several groups of students.

CodegenIT Inc., Colombo, Sri Lanka.

Software Engineering Intern Jan. 2003 - Jun. 2003

Actively participated in the design and development of back-end software for travel and hospitality industry.

Organizations and Clubs

- **Fulbright Association**, Purdue University. Aug. 2006 - Present
  Secretary/Web Master Aug. 2007 - May 2008
  Treasurer/Web Master Aug. 2006 - May 2007
- **Graduate Student Board (GSB)**, Purdue University.
  First year representative Aug. 2006 - May 2007
- **Web Team**, University of Moratuwa, Sri Lanka.
  Web Developer Jan. 2002 - Dec. 2002
- **Computer Society**, University of Moratuwa , Sri Lanka.
  Committee member Jan. 2003 - Dec. 2003

Professional Activities

- ACM, student member
- IEEE, student member
- CODASPY poster track committee member
- Conference and Jouranl reviewer
  - ACM Symposium on Access Control Models and Technologies (SACMAT)
  - International Conference on Distributed Computing Systems (ICDCS)
  - Very Large Data Bases (VLDB)
  - International Conference on Data Engineering (ICDE)
  - Annual International Conference on Financial Cryptography and Data Security (FC)
  - ACM Symposium on Information, Computer and Communications Security (ASIACCS)
  - Annual Computer Security Applications Conference (ACSAC)
  - Extending Database Technology (EDBT)

- ACM Conference on Data and Application Security and Privacy (CODASPY)
- IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY)
- IEEE Transaction on Knowledge and Data Engineering (TKDE)
- IEEE Transactions on Dependable and Secure Computing (TDSC)
- IEEE Transactions on Information Forensics and Security
- International Journal of Information Security