A Survey on Group Signature Schemes

Subhra Mishra
Tilak Rajan Sahoo

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela – 769 008, India

A Survey on Group Signature Schemes

Dissertation submitted in

May 2014

to the department of

Computer Science and Engineering

of

National Institute of Technology Rourkela

in partial fulfillment of the requirements

for the degree of

Bachelor of Technology

by

Subhra Mishra

(Roll 110CS0590)

Tilak Rajan Sahoo

(Roll 110CS0148)

under the supervision of

Dr. Sujata Mohanty

Department of Computer Science and Engineering

National Institute of Technology Rourkela

Rourkela – 769 008, India

dedicated to our Parents and Sisters...

Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, India. www.nitrkl.ac.in

Dr. Sujata Mohanty
Asst. Professor

Certificate

This is to certify that the work in the thesis entitled *A Survey on Group Signature Schemes* by *Subhra Mishra and Tilak Rajan Sahoo*, bearing roll numbers 110CS0590 and 110CS0148 respectively, is a record of an original research work carried out by them under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of *Bachelor of Technology* in *Computer Science and Engineering*. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Dr. Sujata Mohanty

Declaration of Authorship

We, Subhra Mishra and Tilak Rajan Sahoo, declare that this thesis titled, "A Survey on Group Signature Schemes" and the work presented in it are my own. I confirm that:

- This work was done completely by us while in candidature for a B-Tech degree at this Institute.
- Where any portion of this thesis has previously been submitted for a degree or any other qualification at this Institute or any other University, this has been clearly stated in the references.
- Where I have consulted the published work of others, this is always clearly mentioned.
- Where I have quoted from the work of others, the source is also mentioned. This thesis is completely my own work with the exception of such quotations. If ever any dispute occurs, my supervisor is not responsible for that.
- I have acknowledged each and every main source of help.

Signed: ________________________________

Date: ________________________________

Acknowledgement

Apart from the efforts of me, the success of any project depends largely on theencouragement and guidelines of many others.I offer my enduring gratitude to thefaculty, my fellow students, who have inspired me to continue my work in this field.My first and sincere appreciation goes to *Dr. Sujata Mohanty*, my supervisor for all I have learnt and the continuous help and support in all the stages of this project. I would also like to thank her for being an open person to ideas and for encouraging and helping me to shape my interest and ideas. For all I learnt from her, and for providing the vision for the success of project.

I would like to thank my family, relatives and friends for always believing in me, for their continuous love and support in my decisions and especially toourbeloved sisters, Sipra Mishra and Tamanna Sahoo, who have always been in supporting rolesto me and without whom I could not have made it here. In the end, I would like to extend my thanks to our Head of Department, *Prof.S.K.Rath* for valuable advice and encouragement with providing me the resources that I used for this research.The guidance and support received from all the members who contributed and who are contributing to this project, was vital for the success of the project. I am grateful for their consistent support and help.

Subhra Mishra

Tilak Rajan Sahoo

Abstract

Group Signature, extension of digital signature, allows members of a group to sign messages on behalf of the group, such that the resulting signature does not reveal the identity of the signer. Any client can verify the authenticity of the document by using the public key parameters of the group. In case of dispute, only a designated group manager, because of his special property, is able to open signatures, and thus reveal the signer's identity. Its applications are widespread, especially in e-commerce such as e-cash, e-voting and e-auction.

This thesis incorporates the detailed study of various group signature schemes, their cryptographic concepts and the main contributions in this field. We implemented a popular group signature scheme based upon elliptic curve cryptosystems. Moreover, the group signature is dynamic i.e. remains valid, if some members leave the group or some new members join the group. Full traceability feature is also included in the implemented scheme. For enhanced security the the scheme implements distributed roles of the group manager. We also analysed various security features, formal models, challenges and cryptanalysis of some significant contributions in this area.

Keywords: anonymity; eliptic curve; digital signature; unforgeability; discrete logarithm

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1. Introduction

A vital role in our lives is nowadays played by cryptography, mainly in information technologies, sometimes we even do not realize how relevant. We use it every time we communicate with our bank, we require secrecy during browsing the web, while sending an email to and there are also many other situations where we want to keep our data secret[38]. Sometimes, we just want to hide our and addressee identity. Also group signatures have inconsiderable importance, as it was mentioned, mainly in information technologies.

A digital signature is a computational technique in cryptography in order to facilitate the authorization of a digital message or document. A valid digital signature proves the receiver that the message was sent in by an valid sender, also the sender cannot lie about not having sent the message and that the information was not tampered in its course. This techinique of cryptography is basically applied in software industry, financial transactions, and in cases of legal disputes where there is a necessity to track frauds or counterfeiting of information[38].

Extending the idea of digital signature schemes into groups, a new signature scheme i.e. group signature scheme, first introduced by Chaum and Heyst in 1991, provides authority to any group member to sign messages anonymously on behalf of the group [1]. A client can verify the authenticity of the signature by using only the group's public key parameters. It must be computationally hard to identity of the group member so that he cannot be linked from a signed message or his signature. However, in the case of a legal dispute, the identity of a signer or member can be revealed by a designated entity i.e. the group manager. The major feature of group signature is the security of the information or the data that makes it more important as well as attractive for many real time applications, such as e-commerce, e-auction and e-voting, where the priority is privacy and anonymity of signer which is very much high and important for any organization.

As mentioned above, group signature scheme was first introduced by David Chaum and Eugene van Heyst [1]. They presented that time a new type of signature for a group of persons (of course, they do not have to be humans necessarily, but for example computers in the network, smart/sim cards etc.) satisfying the following properties [1]:

(1) none but members of the group only can sign messages;

(2) the receiver of message can verify that the message received by him is signed by a valid authorized group member, but he can not discover identity of group member making the signature;

(3) if necessary in case of legal dispute, the anonymity of the group member who signed the message is revealed.

Thus, as seen from the first and second property, one of the major points of the group signature scheme is to provide anonymity to its signers (i.e. group members). Every group member has a private secret key which enables him to sign messages, but resultant signature maintains the secrecy of the identity of the signer. The third property tells, there is an higher entity (generally called group manager) who has the power and resources to track the signing member, or reveal the signer's identity by using a special algorithm. Revocation of members is supported by some systems as well [3], i.e., where group member can be revoked (or disabled) without putting any affect the ability to sign of unrevoked members (in this thesis, this case is treated).

Chaum and van Heyst described a "classical" example as an application of group signature thet we can mention[1]. Let a company which has a number of computers be considered, each of which is connected to the company's local network. Company has also departments, all of which have their own printers (connected to the network). Only members of the department are allowed to use their department's printer. Therefore, before printing, printer must be convinced that the user is from its department. Also, company (or users) require privacy, so member's name must not be revealed. But in case, if it is necessary like in a legal dispute, for example if someone prints too much, senior authority in charge must be able to discover or reveal his identity.

A number of group signature schemes followed and have been proposed after the initial scheme proposed by D. Chaum and van Heyst . A dynamic group signature scheme was architectured by Chen and Pedersen, that allows new members to join the group anytime. Usage of group signatures in e-bidding[2] was also suggested by them. The first group signature scheme that could be used for large groups was put forward by Camenisch and Stadler as in that scheme the public key of the group and the signatures have lengths that do not depend on the size of the group [4]. Later, a scheme to support efficient revocation of group members was proposed by Kim et al.Some obstacles hindering the way of real world applications of group signatures were brought forward by Ateniese and Tsudik, like coalition attacks and member deletion[13].

In the literature review, it was observed that currently the available group signature schemes can be differentiated into two types, firstly registration by a public-key, and secondly a certificate-based type. In the first protocol, keys are constructed by using only groups of known order. However, in above mentioned schemes, both the keys of the group and the size of signature are not independant of the number of group members. This a serious problem considering groups with huge number of members. In the second protocol, a certificate of membership is given to each member of the group, and the group signature is based on secret public key of membership certificate. Therefore, neither the group public key nor the size of the signatures of members depend on the number of members in the group.

1.1 Motivation

A we know of digital signature and facilities it has provided regarding information security, so extending the idea of digital signature to group where we can parallely authorize multiple information or documents and save time. Group Signatures have vital role in day to day corporate organizations' e-commerce applications. Increase in demand for a more secure and lesser complex Group Signature scheme has always been there. The scheme implemented by us provides these features. The use of elliptic curve cryptography increases the security the scheme by providing desired security level that is achieved by significantly smaller keys in elliptic curve system than in its counterpart- RSA system. Another significant advantage being in general, the algorithms used for encryption and decryption in ECC schemes are faster and can be run on machines that are less efficient.

1.2 Basic Concepts and Requirements of Group Signature

A technique of authorizing the documents, messages or relevant information anonymously on behalf of group by any member belonging to it is termed as a group signature scheme, where the group consists of a manager and valid members shown in Figure 1.1. The integrity of sign is verified by a trusted verifier, where he is aware of the sign's correctness but not the identity of the signing member. This concept of group signature put forward by Chaum and Heyst which allows any member of a group to authorize message on behalf of the group. According to their scheme, the following policies must be included in any group signature scheme:

- Group members are only role persons to authorize the messages by signing them.
- The validation of the signature should be verified without the identity of the signer being revealed
- If a situation of necessity arises, the signature can be opened to reveal the anonymity of signer.

In group signature schemes, the only person capable of addition of new members and revoking of the existing members from the group is the group manager. In case of legal disputes, if any, the responsibility of revealing the identity of the member or signer is of group manager's. Also we have to take care that all channels used during the communication are not synchronous which says the sender after putting a message though the channel does not need to wait for the receiver to receive the message off the channel. The channel communication between the sender and the receiver is assumed to be anonymous. Basic terms used in the group signature schemes are group public key which the verifier uses to check the validation of the signature, group's secret key which is used by a member of group to generate his signature and the group manager's secret key that is used to track the identity of the signing member. A standardized group signature scheme contains the following five phases[18]:

1 setup phase: group manager computes the public key and the secret key in this phase by implementing the algorithm for group key generation. He inputs a security parameter to the algorithm and it returns the group public key and also the secret key of group manager. The secret key is kept with him and the group public key is circulated among the members.

2. Join phase: an interactive protocol is established in this phase between the group manager and the to-be-member after which the user becomes a valid group member. A secret key is chosen by the Group member using which another parameter is generated by the member. This generated parameter is sent to the group manager. Then using his own secret key the group manager generates the group member's signing key and returns it to newly joined group member.

3. Sign phase: This is the signing phase in which an protocol is established between the group member and the verifier where he has to verify a group signature whether it is generated by a valid group member or not. Group member uses the signing key pairs to sign the message. The generated group member signature of knowledge is sent by the member to the verifier for verification.
4. Verify phase: This phase implements a deterministic algorithm using given group public key and the signed message to verify the validity of the group signature. Signer sends his signature to the verifier, i.e. the signature generated by the signature of knowledge. The message is accepted if true value is returned by the verification phase else the message is rejected if false value is returned by the verification phase.
5. Open phase: This phase implements a deterministic algorithm to reveal the identity of the signer, by taking input a signed message and the secret key of group manager. The signature is taken as input by the group manager and using the private parameters outputs the identity of the signer as return value. This open algorithm is implemented when a incident of a legal dispute arises.

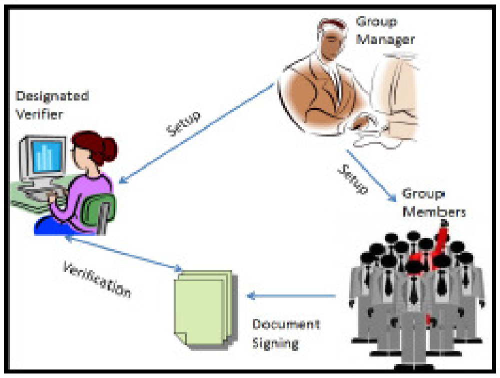


Figure 1.1: Layout of standard group signature system

1.3 Properties

- Anonymity: Given a sign which is valid must be difficult for anyone to discover the identity of the signer computationally. As the constant differs every time, the same member generates different signature for every new message to be signed. The group manager only can determine the identity of the signing member using his secret key. For a non-member it is almost not possible to discover the secret parameters of the signing group member as the knowledge of the secret key of the group manager is required and so without the secret key of the group manager it is almost impossible to determine the secret parameters of the signer and hence an outsider cannot cannot determined the identity of the signer. In this property we conclude that if neither group manager's secret key nor group member's secret key is exposed then it is infeasible to reveal the signer of a authorized valid signature.
- Unforgeability: Only a valid authorized member belonging to the group can produce a valid signature i.e. a valid member only can produce a signature on behlf of his group.
- Unlinkability: This property states that deciding if two valid signatures were generated by the same group member is difficult. According to this

property one cannot conclude that both signatures are from the same member or not if he’s provided with two signatures.

- Traceability: Using only open algorithm and the group manager's secret key, the group manager can track the identity of the signing member if given any valid signature. Like in case of any legal dispute or emergencies, any signer’s identity can be traced by the group manager only. It is not possible for an outsider to track the signer because open algorithm, which used to trace a signing group member, requires the knowledge of the secret key of the group manager.
- Exculpability: The group members even alongwith the group manager are not able to sign a document on behalf of any other group member. The knowledge of the secret parameters of the group member is required to generate a valid signature. And every member has his own unique secret key that are used to generate the signature. Even a group manager cannot sign on behalf of any group member because the group manager does not have the members’ secret keys.

1.4 Elliptic curve cryptography

Elliptic curve cryptography (ECC) was first introduced in 1985 independently by N. Koblitz and V. Miller[18]. They proposed to use elliptic curves in cryptographic schemes. In 1990’s, elliptic curve schemes went through commercial acceptance. EC cryptography schemes are public-key mechanisms which are able to give the same facilities as the schemes of RSA or ElGamal. But the security of ECC is based on a hardness of another problem, known as the elliptic curve discrete logarithm problem (ECDLP). The best algorithms to solve ECDLP have full exponential-time (unlike RSA’s algorithms which have the subexponential-time). Thus, required security level can by achieved with significantly smaller keys in elliptic curve system than in its rival- RSA system.

For example, in general it is accepted that a 160-bit elliptic curve key can give the security of same level as a 1024-bit RSA key. One more advantage of ECC schemes over others is that, encryption and decryption algorithms in ECC schemes are faster as well are able to run on machines which are less efficient.

1.5 Problem Statement

The objective of thesis is to study and review existing group signature schemes, elliptic curve cryptography concepts, finally implementing a group signature scheme based on following assumptions:

- Group signature scheme based upon hard computational assumptions, such as, elliptic curve cryptography (ECC)
- Group signature scheme should be unaffected by joining or leaving of any member.
- Group signature scheme must satisfy all basic security requiremenst like anonymity, traceability, and unlinkability.

1.6 Organization of Thesis

This thesis is organized as follows, Chapter 2 gives a brief introduction of preliminaries of cryptography. An existing scheme is described in Chapter 3. alongwith its security analysis and corectness evaluation. Discussion about implementation and result is depicted in Chapter 4. Finally conclusion in Chapter 5.

Chapter 2

Literature Survey

Chapter 2

Literature Survey

In this chapter, we have reviewed the literature related to various group signature schemes and their security features. First, we have given a brief overview of cryptography concepts then preliminaries related to elliptic curve cryptography[36], hash functions[31], random number generations[32], and prime number with primality test[35]. Later, we have reviewed some popular group signature schemes based on their security features.

2.1 Cryptography Concepts and Signature Requirements

Cryptography could be characterized as securing data by transforming it into an unreadable structure, called cipher text. Just those who have a secret key can decipher the message into plain text[38]. Encoded messages can sometimes be broken by cryptanalysis, also called code breaking, in spite of the fact that present day cryptography techniques are virtually unbreakable. Cryptography system might be extensively classified into symmetric-key system which uses a single key that both the sender and recipient have, and asymmetric-key system which uses two keys, a public key known to everybody and a private key which just the recipient of messages uses, yet in case of the signature, it needs a public key system where the signer signs with private key and the verifier verfies with the signer's public key.

2.1.1 Elliptic Curve Cryptography

First, we will do an assessment of some primitives of finite fields. Mostly, finite field F is an algebraic structure $(F,+,\cdot)$ that contains a set F and two operations (+) and (·), holding the following properties:

(1) F is an additive group with reference to operation (+);

(2) F \ *{0}* is an multiplicative group with reference to operation (·);

(3) for all $a, b, c \in$ F supports

$a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ and $(a + b) \cdot c = (a \cdot c) + (b \cdot c)$.

Two types of finite fields are employed in elliptic curve cryptography thus there are two types of elliptic curves as follows:

(1) Elliptic curves over a field Fp = $\{0, 1, \ldots, p-1\}$ of prime characterization p;

(2) Elliptic curves over a field $F_{2^m} = \{0, 1, \ldots, 2^m - 1\}$ of characterization 2.

Only first type of fields employed for elliptic curve cryptography will be used in this paper. Now, we can define an elliptic curve over a finite field.

Definition 2. *An elliptic curve E over a finite field* F*p is elucidated by an equation of the form*

$E : y^2 = x^3 + ax + b \pmod{p}$,.. (1)

where a, b $\in$ F*p and* $\Delta = 4a^3 + 27b^2 \neq 0 \pmod{p}$.

An equation (1) is called simplified Weierstrass equation (general form of Weierstrass equation is required for curves over a field F_{2^m}), Δ is known as *discriminant* of elliptic curve E and the condition that $\Delta \neq 0$ makes certain that curve is "smooth", or that is, there are no points at which the curve has more than one discrete tangent lines. A pair (x, y), where $x, y \in$ Fp, is a *point* on the curve if (x, y) meets equation (1). The *point at infinity* represented by ∞, is also assumed to be on the curve. The set of all points on E is given by E(Fp).

Strictly,

$$E(F_p)=\{(x,y)\in F_p^{\,2};\, y^2=x^3+ax+b\}\cup\{\infty\}.$$

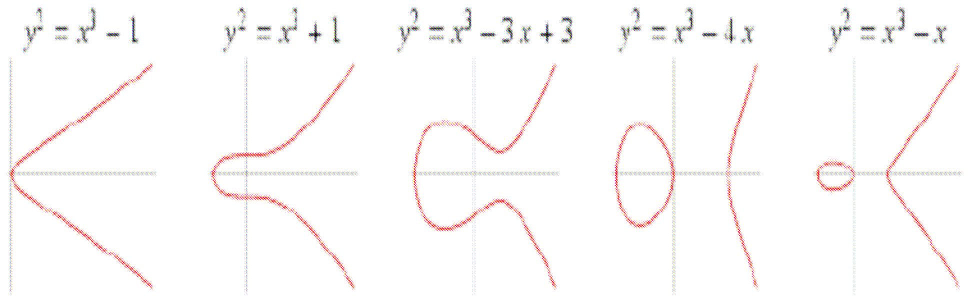
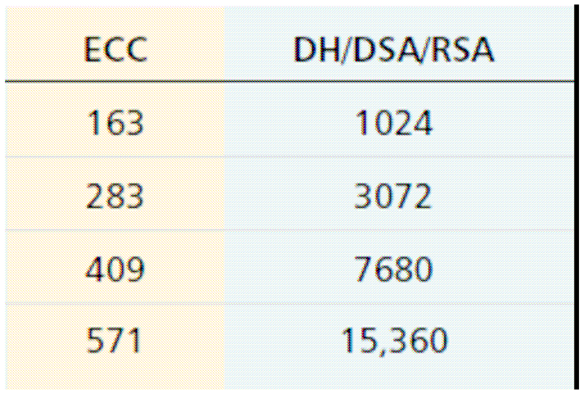


Fig. 2.1 Examples of elliptic curves in R2.

Elliptic curve cryptography schemes might be utilized as public-key mechanisms that give the same functionality as RSA or Elgamal schemes. 160-bit elliptic curve key gives the same level of security as a 1024-bit RSA key. Calculations for encryption and unscrambling for ECC schemes are quicker and could be run on less productive machines.

For the same level of security for every best presently known attacks, elliptic curve-based frameworks might be implemented with much littler parameters, prompting huge performance advantages. The performance advantages of elliptic curves in the SSL/TLS protocol have been analyzed in profundity.



ECC	DH/DSA/RSA
163	1024
283	3072
409	7680
571	15,360

Table 2.1. *Key sizes for equal security levels(in bits).*

2.1.2 Cryptographic Hash Function

A cryptographic hash function is hash function which transforms random block of information and provides a fixed size string where each data is mapped such that any modification would vary the value of hash with very high probability[31]. The information to be concealed is known to be the message and the hash value obtained is known as the message digest or digest. Ideally the hash function must satisfy certain qualities, firstly should be staright-forward to process the hash value for given message and in the meantime must be unfeasible to produce a message with a random hash and likewise be safe against modifying a message without the hash. We may come upon a long list of cryptographic hash functions, whereas many have been found to be liable and should not be used. Considering the integrity of information we may use hash function like SHA 1, MD2, MD4 and MD5 where each scheme can be employed to provide a digest of respective bits determined by the requirement of message or information integrity.

2.1.3 Random Number Generator

A random number generator is a computational device built to create a sequence of numbers that doesn't have any pattern, i.e. have all the earmarks of being random[32]. The numerous usages of randomness have come about to the development of numerous diverse methods for producing random information. Random number generators are extremely productive in developing Monte Carlo-method simulations, as debugging is elevated by the possibility to run the same arrangement of random numbers again and again by beginning from the same random seed. They are additionally utilized in cryptography - so long as the seed is secret. Sender and receiver can handle the same set of numbers consequently to use as keys. There are two essential methods used to create random numbers. One ascertains some physical occasion that is required to be random and then adjusts for conceivable predispositions in the estimation process. Alternate makes utilization of computational calculations which can handle long chains of obviously random effects, which are indeed totaly dictated by a shorter introductory quality, called a seed or key. The recent sort is frequently known as pseudorandom number generators.

2.1.4 Prime Numbers and Primality Test

A primality test is an algorithm for figuring out whether an input number is prime. Throughout different fields of mathematics, it is utilized for cryptography[35]. Dissimilar to integer factorization, primality tests does not regularly give prime factors, it just states if the input number is prime or not. Factorization is thought to be a computationally troublesome issue, inasmuch as primality testing is almost simple. Primality tests could be characterized in two sorts: deterministic and probabilistic.

Deterministic Algorithm: A deterministic primality testing calculation acknowledges an integer and dependably yields a prime or a composite. Deterministic tests focus with absolute certainty whether a number is prime. As of not long ago, all deterministic calculations were so lacking at discovering bigger primes that they were viewed as infeasible. In 2002, Agrawal, Kayal and Saxena affirmed that that they had discovered an algorithm for primality testing with polynomial timecomplexity of O((log 12n)) .

Probabilistic Algorithm: Probabilistic tests can conceivably (in spite of the fact that with little likelihood) falsely recognize a composite number as prime .However, they are when all is said in done much speedier than deterministic tests. Numbers that have passed a probabilistic prime test are consequently appropriately alluded to as possible primes until their primality could be showed deterministically.

2.2 Classification of group signature schemes

The existing group signature schemes/protocols can be one of the four types, namely Static Group signature, secondly Dynamic group signature with revocation, third is group signature scheme with verifiable opening and the final group signature scheme where group manager can be distributed among different roles. The basic functionality remains similar following the standardized group signature scheme considering secret and public keys' generation, the group key generation and, designated verification as well as opening of group signature.

2.2.1 Static Group Signature

Static group signatures contains four polynomial time algorithms[27] namely key generation in which the system generates the public key of the group, alongwith generating the secret key for signing of messages, signature generating algorithm which takes input the secret key and the information for signing and returns the signed message, signature verifcation algorithm which accepts input the public key of group, the signature with the document and returns the value as true or false, finally the opening algorithm which takes secret key of group manager, signed document and the signature to reveal the identity of user who signed. In general, a static group signature computes all the parameters initially with the members of the group and also revocation of existing member is possible only by removing of member. But addition of any new member is not allowed.

2.2.2 Dynamic Group Signature

Dynamic group signature [22] as suggested by the name is the non-deterministic and randomness nature of scheme. The dynamic group signature contais five polynomial algorithms known as signature key parameter generation in which public parameters as well as secret parameters of a member is determined by generating a new list which keeps tracking the registration of group member, joining phase in which two algorithms are computed, firstly the member is registered into the registration list and then secondly the members' parameters for signature are generated, signature generating algorithm which takes input the secret key and the information for signing and returns the signed message, signature verifcation algorithm which accepts input the public key of group, the signature with the document and returns the value as true or false thereby the validity of signature. Finally the opening algorithm which takes secret key of group manager, signed document and the signature to reveal the identity of user who signed the message. The difference between the static and dynamic group signature is the addition of join phase which provides the full revocation as well as addition of new group members.

2.2.3 Group Signature with Verifiable Opening

The main requirement of any digital signature is to protect the signers' identity, still allowing the manager to reveal the anonymity in case of disputes by using the opening algorithm procedure. The group signature with verifable opening has five polynomial algorithm like the dynamic group signature. But the difference here is in the open phase, in which the algorithm is divided into two parts i.e. opening algorithm and the judging algorithm. The basic requirement of group signature scheme does not provide resources to the manager to accuse a member falsely of signature , thus assuring the integrity of the decision of manager, he has to provide additional proof against the member. The opening algorithm takes secret key of group manager, signed document and the signature to reveal the identity of user who signed the message then in judge phase, algorithm input accepts the proof and signature, revealing the validity of signature of manager by proving it a verified open phase.

2.2.4 Group Signature with Distributed Roles

Every Group signature scheme includes the group manager, who is responsible for many roles in the signature protocol. The manager has two major tasks i.e. the group membership and the signature opening, these two tasks can be divided into two different entitties namely, the issuer and the opener as distributed manager's role. This group signature scheme contains the basic polynomial algorithms except differing in key generation in which the algorithm provides secret key to issuer and secret key to opener alongwith group's public parameters, the issuer runs the join procedure in which the registration list is updated after every successful join operation and the opening procedure in which the other role i.e. opener opebns the signature in case of disputes. These can be made into the opening group signature with verification by including the proof and validity of judge. Another approach can be requirement of a third trusted party(TTP) to generate in advance both the private keys and then give the keys to the issuer and the opener respectively using secure channels.

2.3 Application of Group Signatures

2.3.1 e-Voting

E-voting also called as electronic voting is a method to cast votes and electronically count the votes. E-voting is supervised physically by government representatives of and electoral authorities which are independent where group signature can be very useful. Voting is done within sole influence of the voters, and since trusted party is needed to govern the voting scheme which is government authorities where a vital role is played by the authorization, group signature can be best applicable.

2.3.2 Sales and Auction System

Electronic commerce, also known as e-commerce, is a type of business in which buying and selling of services and products is carries out over electronic systems like the Internet and other digital networks. It consists of the data exchanging thereby facilitating the financing and aspects of payment of business transactions. Group signature is also an efficient and effective way of assuring security during communication within or with an organization.

2.3.3 Corporate Organisation

Every well developed organisation consist of many roles working for the achievement of particular aim that consists of sensible data to be shared among themselves, thus group signature can be an efficient way to validate the information among everyone thereby saving the valuable time by implementing a reliable approach.

2.4 Literature review of Group signature schemes

2.4.1 Group Signature based on DLP

Chaum and Heyst presented the group signature scheme based on DLP. In 1997, Park, Kim and Won proposed an ID-based group signature [6]. The standard responsibility of their scheme is that signer's public key is identification (ID) that does not need to be verified, so there is no constraining motivation to set up a trusted center to verify a giant number of public keys. By the by, an ID-based group signature must ustilize a set of group member identities in the signing phase. Exactly when the group member changes, the group signature is latent and plus the length of its signature increases with the measure of members. In 1998, Lee and Chang proposed a capable group signature based on the discrete logarithm[18]. The scheme was more powerful the extent that computational, communication and storage costs are concerned, while allowing the group to be changed without having the members picking the new keys. Regardless, when the signer has been perceived, the authority must redistribute the keys of this signer and send the keys to him/her. In 1999, Tseng and Jan intended to upgrade the formerly expressed issue to propose an improved group signature that is based on the Lee-Chang scheme[8]. In that year, Sun exhibited in that the Tseng-Jan scheme is still not unlinkable. After that, Tseng-Jan [9] proposed to improve their scheme. In 2000, Li et al.[2] demonstrated that two schemes of the Tseng-Sun's paper, which are called Tj1 and Tj2 in Li et al's paper, both could be attacked. The threshold group signature is a key kind of signature. Various threshold group signatures are proposed however numerous accomplished conspiracy attack and are insecure.

2.4.2 Group Signature with anonymity and separability

We have group signature based on strong separability Shundong Xia, where author proposed secure scheme based on discrete logarithm problem, such that group manager might be part into membership manager and revocation manager. Earlier proposed group signature scheme were not having identity regarding the public keys, thus requiring the manger to maintain data to map the identity information .

The scheme recommended that past schemes may have frail manifestation of separability if proper communication is not accessible between revocation and membership manager hence defending strong separability.

In the paper Fucai Zhou, 2008 anonymity of signature was contrasted with the group signature where they examined a critical problem, that is the signatures are produced on behalf of group or group member and concluding that the signature ought to be produced on behalf of group and additionally pointed the conict of authenticated content[26]. In 2009, another enhanced group signature was presented by Cheng Lee et al. where the problem of unlinkability and unforgeability was upgraded based on the discrete logarithm.

2.4.3 Group Signature based on Threshold Scheme

The group signature based on threshold scheme might be characterized as group oriented (t, n) traceable signers and group oriented anonymous signers. The signature was ended up being under forgery attack in paper proposed by Z.c. Li, 2001.threshold based signature was under revision by numerous authors and additionally being utilized within proxy and blind signaturesIn the paper Yuan-Lung Yu, 2005 the author consolidates the short mystery key characteristic for the elliptic curve cryptosystem and the (t, n) threshold strategy to make a signature scheme with simultaneous signing. The perceiving characteristic of the proposed scheme is that the threshold value indicates the minimum number of members required to process a valid group signature. All message recipients then can affirm the signature. Numerous threshold group signature schemes have been proposed, yet most of them encounter the sick impacts of conspiracy attack and are insecure. In this paper Fengyin Li, 2007, taking into account the discrete logarithm problem, an ensured threshold group signature scheme is proposed. The scheme is threshold-signing, and also threshold-verifying. In the paper, Fucai Zhou showed the essential of real group signature and gave an alternate scheme to comprehend a true group signature, which is focused around pivot threshold scheme[7].in 2011, Improvement of threshold group signature scheme was exhibited by Tong lu and Baoyuankang where the scheme proposes to be more secure as giving the strong unforgeability focused around discrete logarithm problem.

2.4.4 Short Group Signature

The Group signature schemes are revised concerning numerous security factors where size of the signature was recognized to be principle issue by a few authors as contrasted with the complexities of signature generation schemes. In 2004 Dan Boneh a short signature scheme was proposed where they gave a plan that has pretty nearly the size of RSA signature standard with same security. The plan was focused around bilinear groups with Strong Diffie Hellman assumptions (SDH). Many schemes were produced that might be productive and short in size yet acknowledging the security of the signature. In 2006, the author acknowledged the formal security model which has been proposed by Bellare, Shi and Zang, including both dynamic groups,concurrent join and proposed amazingly dynamic short signature scheme with solid security under random oracle assumption[23]. The signature scheme was focused around Strong Diffie Hellman assumptions (SDH) and external Diffie Hellman assumptions. Starting late a paper on Short group signature with control linkability (Jung Yeon Hwang, Chung, Cho, & Nyang, 2011) focuses at giving dynamic membership where the controllable connection capacity enables an entity who has linking key to check if two signatures are from the same signer while protecting anonymity. The plan is sufficiently powerful and suitable for real-time applications even with restricted resources, for instance, vehicular adhoc network and Trusted Platform Module in the meantime plan supporting controllable connection capacity gives a signature that is shorter than the standard normal group signature.

2.5 Chapter Summary

The audit of group signature provides for us the thought of improvement of signature scheme with different security features to be appropriate in real time application, yet because of the complexities and active attacks analyzed, has fizzled the scheme to be completely relevant. The most unpredictable attack is the colluding attack where the signature schemes proposed, recognizes just the features that are foreseeable. Accordingly assessing the schemes,we implement one of the recent schemes proposed by Krystian Baniak, in the year 2011, using elliptic curve cryptosystem.

Chapter 3

Group Signature Scheme based on Eliptic Curve Cryptography

3.1 Mechanism

This group signature architecture proposed by Kristian Baniak[37] is a dynamic group signature scheme with application of revocable anonimity and distributed roles of the manager. The set of phases for this scheme consists of the following procedures: Issue/Join, Open, Revoke, Sign and finally Verify. Another key facility it provides for the scheme construction is the division of group manager's responsibilities into the different modules. In addition the set of group signature scheme phases is partitioned into the public and protected classes thereby putting a limit to the number of the oracles to increase security access.

The key elements of Krystian's group signature scheme are the following:

Group Issuing Manager: (IM) – one who is responsible for the adding of group members alongwith the maintenance of the secret database of member certificates. The Group Manager has a *isk* key used for provisioning new members and implements group signature scheme phases like *Join/Issue*.

Group Opening Manager: (GOM) – implemented on the Members List.

• Open, used by the Opening manager GOM has a key *omk* that is used to open a signature and reveal the identity of the signing member. Open, actually performs a revocation of subscriber's identity.

Group Revocation Manager: (GRM) – It is used to revocate a member in the case of treachery. GRM uses the Remove procedure computed on the Group Member List.

• Revoke, used to disable the group member, when necessary.

Group Member: – any member, implementing the Sign procedure.

The members of the group signature scheme, use the Sign procedure to authenticate messages and documents. Generally, the signature is constructed over the digest of the exchanged message.

GMRL is a list of revoked group members that contains the following records: (*TKi; Tr*), where the token *TKi* is the revocation token of an member *agi* of index *i*, is created during the Join phase and *Tr* is the time of occurrence revocation.

• a message digest function *Hash*

Procedure 3.1 Group Signature Scheme Setup

```
$R_1 \leftarrow \{0,1\}^{\rho_1}$;   $R_2 \leftarrow \{0,1\}^{\rho_2}$;   $R_3 \leftarrow \{0,1\}^{k}$;
$(pk_e, sk_e) \leftarrow K_e(1^k, r_e)$,   $r_e$ is a random value
$(pk_s, sk_s) \leftarrow K_s(1^k)$;
$gpk \leftarrow (1^k, R_1, R_2, R_3, pk_e, pk_s)$,   group public key
$gmsk \leftarrow (sk_s)$,   group manager key
$gomk \leftarrow SSEC(sk_e, r_e)$,   GOM private key protected with the secret sharing scheme
$cert_{GPK} \leftarrow (gpk, T_c, E(K_{AD}^{-1} : Hash(gpk, T_c)))$,   group manager certificate
$cert_{GMRL} \leftarrow (R_3, T_c, E(K_{AD}^{-1} : Hash(R_3, T_c)))$,   GMRL certificate
$GMRL \leftarrow \emptyset$
```

The notation used in the cryptographic attributes descriptions is as follows:

(pk_i, sk_i) :	public and private key of a given agent collector instance
$cert_i$:	certificate of agent i generated during the provisioning
agc_{id} :	agent collector's id, reference to the Agent Collector table
T_c :	table entry creation time stamp
T_r :	revocation entry creation time stamp *entry.time_revoked*
RT_i :	revocation token generated for a group member in `Join`
TK_i :	revocation token generated for a group member in `Sign`
$Hash(TK_i)$	digest of a revocation token
$rsig_i$:	revocation token signed by the agent i
pk_e :	public key of the group opening manager
(sk_e, r_e)	private key of the group opening manager

Procedure 3.2 Join & Issue Group Procedure

The agent collector generates a key pair $(pki; ski) \leftarrow Ks(1k)$ and signs it to produce

$sigi \leftarrow E(ski : pki)$. It also creates a revocation token $RTi \leftarrow (R3; Ta)$ and creates its

signature $rsigi \leftarrow E(ski : R3; Ta)$. Both items are sent to the Central Repository that is the Group Manager:

$AGi \rightarrow CR : E(KCR : (pki; sigi;RTi; rsigi); Ta;Na;Kr)$

The Agent Directory verifies the signatures *sigi* and *sigR* before continuing the procedure. When signatures match it generates the certificate:

$certi \leftarrow Sign(sks : (i; pki))$ and formulates the response [**4**]. The response from the central repository agent is encrypted with the challenge *Kr* proposed by the agent collector:

$CR \rightarrow AGi : E(Kr : (i; pki; sigi; certi);Na; Tc;E(K$□□$1CR : Hash((i; pki; sigi; certi);Na; Tc)))$

$CR \rightarrow AD : E(KCD : agent = (i; pki; sigi; certi; agcid; Tc; TKi);E(K$□□$1CR : H(agent)))$

The Agent Directory receives the new agent collector registration information and populates *Agents*[*: : :*] the table with the new entry.

Procedure 3.3 Sign procedure $GSig(gpk, m)$

$H_k(m)$ is a k-bit message digest function run on the original message m being signed.

```
    $TK_i \leftarrow Encrypt(pk_e : R_3, T_a)$
    $R_{GMRL} \leftarrow E(K_{AD} : TK_i, T_s, N_a)$
    $h_m \leftarrow H_k(m, R_{GMRL}, T_s)$
    $sgn \leftarrow Sign(sk_i : h_m); \quad r \leftarrow \{0,1\}^k$
    $c \leftarrow Encrypt(pk_e : (i, pk_i, cert_i, sgn), r)$
    $\pi_1 \leftarrow P_1(R_1, (pk_e, pk_s, h_m, c), (i, pk_i, cert_i, s, r))$
    return $(\pi_1, c, T_s, R_{GMRL})$
```

Procedure 3.4 Revocation status check procedure $GMRL(gpk, T_s, R_{GMRL}))$

This procedure uses the object notation for the entries of the GMRL CRL table.

```
    $(TK_i, T_a, N_a) \leftarrow E(K_{AD}^{-1} : R_{GMRL})$
    If $T_a \neq T_s$ return false
    Unless $ValidTokens.has(Hash(TK_i))$ then return false
    Foreach $entry$ in $GMRL[]$ do
      If $entry.token = TK_i$ then
        If $T_a >= entry.time\_revoked$ then
          return false
        End
      End
    End
    return true
```

Procedure 3.5 Open procedure $Open(gpk, gomk, m, \pi_1, c, T_s, R_{GMRL})$

```
$h_m \leftarrow H_k(m, T_s, R_{GMRL})$
$sk_e \leftarrow Compose(gomk)$
$(i, pk, cert, s) \leftarrow Decrypt(sk_e : c)$
If $Agents[i] \neq NULL$ or $pk_i = pk$ then
   $(pk_i, sig_i) \leftarrow Agents[i]$
Else return false
If $V_1(R_1, (pk_e, pk_s, h_m, c), \pi_1) = 0$ then return false
$\pi_2 \leftarrow P_2(R_2, (pk_e, c, i, pk, cert, s), (sk_e, r_e))$
return $(i, \pi_2, pk_i, sig_i, cert, c, s)$
```

Procedure 3.7 Remove procedure $Remove(gomk, i, T_r)$

CR receives a command to disable agent i from the management station. CR fetches the revocation token from the agent directory agents database $Agents[i]$ and new entry is inserted into GMRL table. The token is protected with the *gomk* so it has to be decrypted first. It is necessary to retrieve the secret key sk_e that is protected by the secure secret sharing scheme $SSEC(N, K)$, where K out of N part–key holders are required to commit the procedure.:

```
$T_c \leftarrow Time.now()$
$sk_e \leftarrow Compose(gomk)$
$TK_i \leftarrow Decrypt(sk_e : Agents[i].TK_i)$
$entry \leftarrow (TK_i, Tc, T_r)$
$GMRL[\ldots] \leftarrow entry$
```

Procedure 3.8 Verify procedure $Verify(gpk, m, \pi_1, c, T_s, R_{GMRL})$

$H_k(m)$ is a k-bit message digest function run on the original message m that has been signed by a group member.

```
$(R_1, pk_e, pk_s) \leftarrow gpk$
$h_m \leftarrow H_k(m, R_{GMRL}, T_s)$
If $V_1(R_1, (pk_e, pk_s, h_m, c), \pi_1)$ then
    return $GMRL(gpk, T_s, R_{GMRL})$
Else return false
```

3.9 Security Analysis & Correctness

1. **Unforgeability**

- Only members who successfully ‘join’ the group can sign a message and create a valid signature.

2. **Traceability and Anonymity**

- Only the Opening Manager, given a signature ‘Signature’, can discover (by the help of ‘Osk’-> Opening Manager Secret Key) identity of the group member who created ‘Signature’ (traceability) and an entity not holding an ‘Osk’ is unable to extract identity of the group member who created a signature (anonymity)

3. **Correctness**

- Valid signatures of the group members always verify correctly.
- This requires that for all (Mpk, Msk) aloted to a member, every signature created by a group member verify as valid, except when the user is revocated.

$$\mathit{Verify}(\mathrm{E_0, E_1, E_2, V_0, V_1, V_2, ACOM, BCOM, Vmpk, Vrev, M}) = \text{valid} \Longleftrightarrow (\mathrm{Vmpk, Vrev}) \notin \text{RevocationList}$$

- Revocation List:- List of individual (Mpk, Rpk) of each revocated member

4. **Unlinkability of the signatures**

- A signature is given by:-

$$Signature(\mathrm{E}_{0,}\mathrm{E}_1, \mathrm{E}_2, \mathrm{ACOM}, \mathrm{BCOM}, \mathrm{c}, \mathrm{taux}, \mathrm{taus}, \mathrm{taue'}, \mathrm{taut}, \mathrm{tayE}, \mathrm{index})$$

- where, most of the parameters are calculated from random values, hence, it is infeasible to decide whether two signatures have been created by the same group member or not.

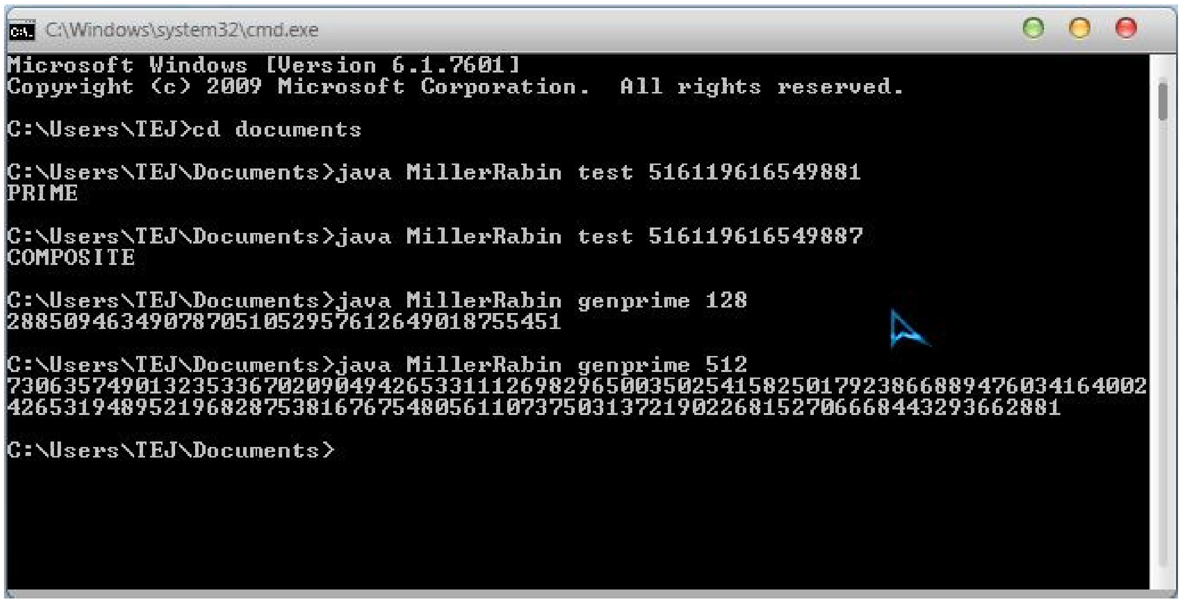
5. **No Framing**

- Even if all group members (and all the group managers) collude, they cannot forge a signature for a non-participating member as they cannot access Msk(x) of the group member (Secret Key of the group member).

Chapter 4

Implementation and Results

4.1 Primality Test: Miller Rabin

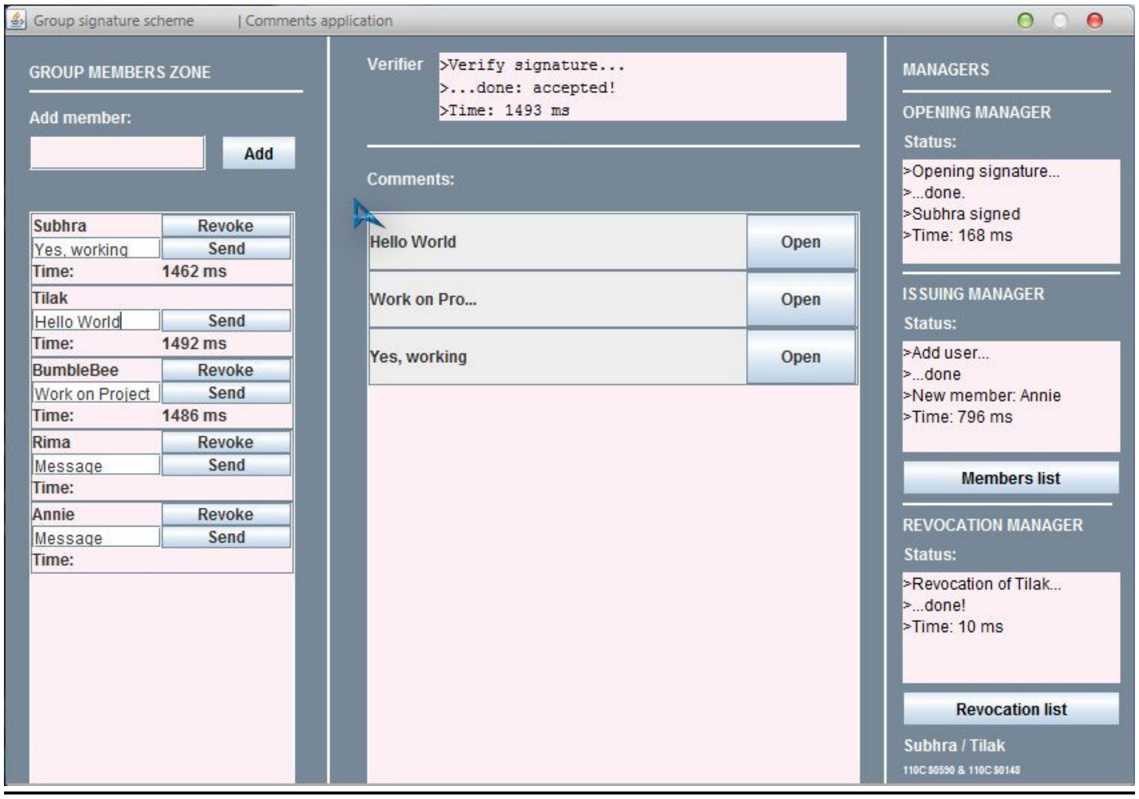


```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\TEJ>cd documents

C:\Users\TEJ\Documents>java MillerRabin test 516119616549881
PRIME

C:\Users\TEJ\Documents>java MillerRabin test 516119616549887
COMPOSITE

C:\Users\TEJ\Documents>java MillerRabin genprime 128
288509463490787051052957612649018755451

C:\Users\TEJ\Documents>java MillerRabin genprime 512
73063574901323533670209049426533111269829650035025415825017923866889476034164002
42653194895219682875381676754805611073750313721902268152706668443293662881

C:\Users\TEJ\Documents>
```

Fig 4.1 Primality Testing implementation

Test Prime Number:	Generate Prime number:
i/p: 516119616549881	i/p(no. of bits): 128
o/p: PRIME	o/p: 28850946349078705105295….55451
i/p: 516119616549887	i/p: 512
o/p: COMPOSITE	o/p: 7306357490132353367….
	....022681527066844329…881

4.2 Group Signature Scheme Implementation:



Subhra	Revoke	
Yes, working	Send	
Time:	1462 ms	
Tilak		
Hello World	Send	
Time:	1492 ms	
BumbleBee	Revoke	
Work on Project	Send	
Time:	1486 ms	
Rima	Revoke	
Message	Send	
Time:		
Annie	Revoke	
Message	Send	
Time:		

```
>Verify signature...
>...done: accepted!
>Time: 1493 ms
```

Comments:

Hello World	Open
Work on Pro...	Open
Yes, working	Open

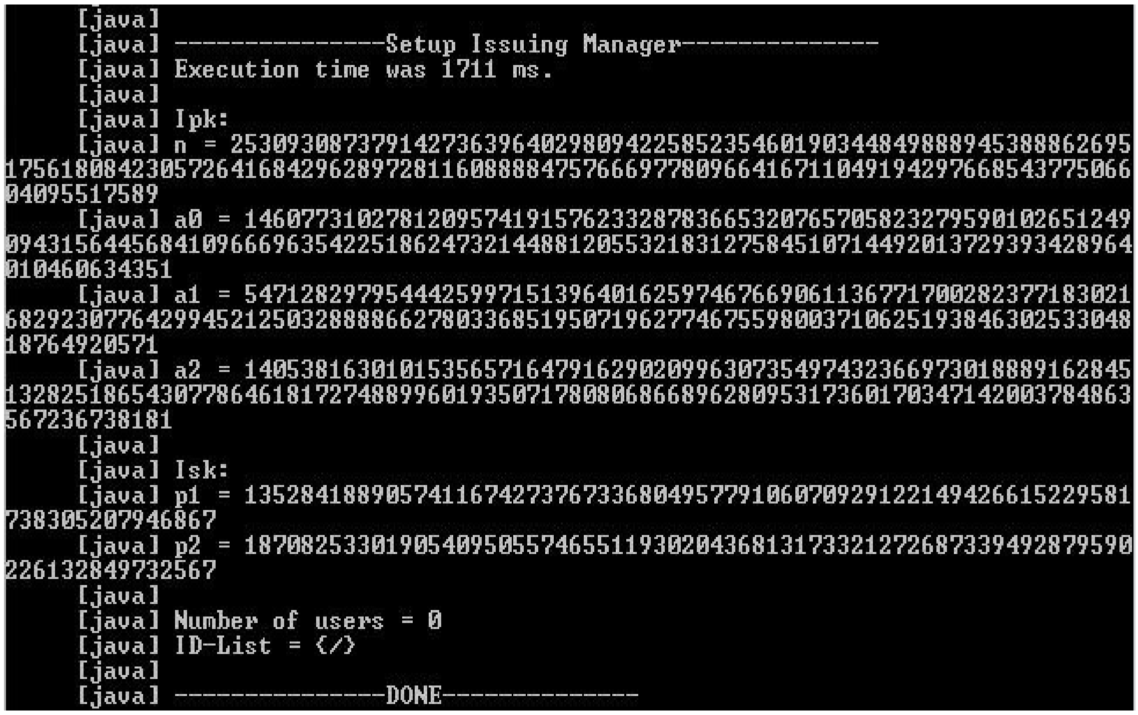
```
>Opening signature...
>...done.
>Subhra signed
>Time: 168 ms
```

```
>Add user...
>...done
>New member: Annie
>Time: 796 ms
```

```
>Revocation of Tilak...
>...done!
>Time: 10 ms
```

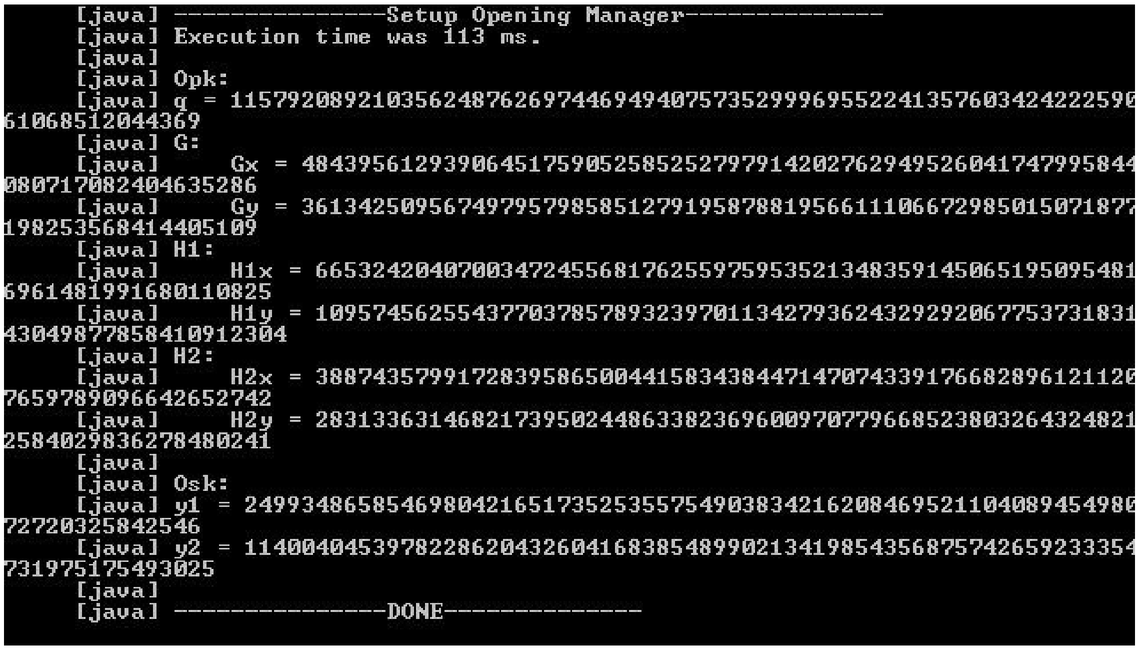
Fig 4.2 *Application GUI view*

4.3 Setup phase



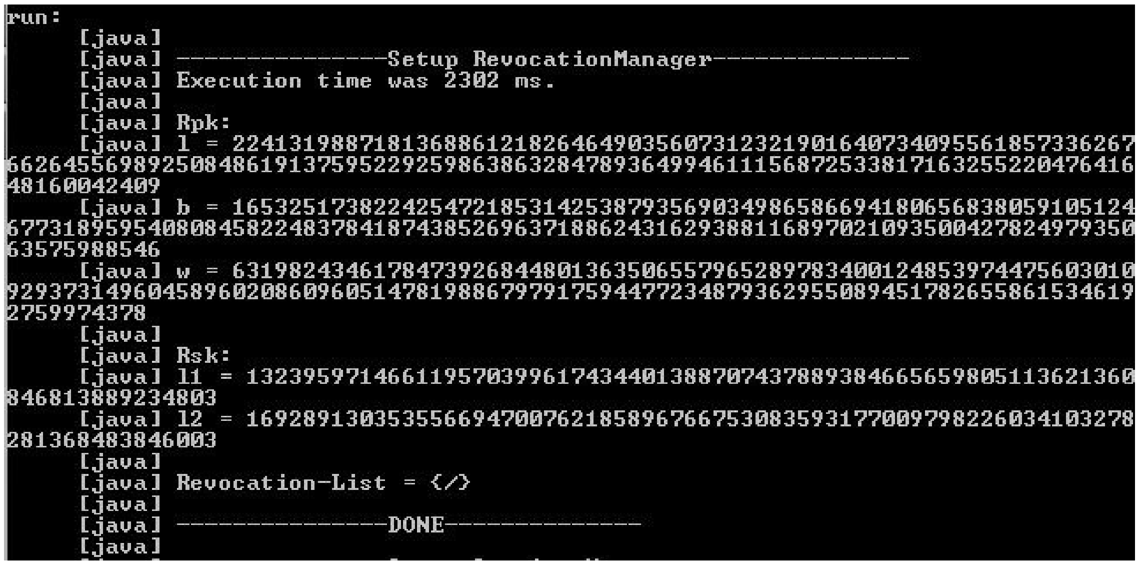
```
      [java]
      [java] ----------------Setup Issuing Manager---------------
      [java] Execution time was 1711 ms.
      [java]
      [java] Ipk:
      [java] n = 2530930873791427363964029809422585235460190344849888945388862695
1756180842305726416842962897281160888847576669778096641671104919429766854377506604095517589
      [java] a0 = 14607731027812095741915762332878366532076570582327959010265124909431564456841096669635422518624732144881205532183127584510714492013729393428964010460634351
      [java] a1 = 5471282979544425997151396401625974676690611367717002823771830216829230776429945212503288886627803368519507196277467559800371062519384630253304818764920571
      [java] a2 = 14053816301015356571647916290209963073549743236697301888916284513282518654307786461817274889960193507178080686689628095317360170347142003784863567236738181
      [java]
      [java] Isk:
      [java] p1 = 135284188905741167427376733680495779106070929122149426615229581738305207946867
      [java] p2 = 187082533019054095055746551193020436813173321272687339492879590226132849732567
      [java]
      [java] Number of users = 0
      [java] ID-List = {/}
      [java]
      [java] ----------------DONE---------------
```

4.3 *Setup Issuing Manager*



```
      [java] ----------------Setup Opening Manager---------------
      [java] Execution time was 113 ms.
      [java]
      [java] Opk:
      [java] q = 1157920892103562487626974469494075735299969552241357603424222590
61068512044369
      [java] G:
      [java]     Gx = 484395612939064517590525852527979142027629495260417479958440
80717082404635286
      [java]     Gy = 361342509567497957985851279195878819566111066729850150718771
9825356841440510 9
      [java]     H1:
      [java]     H1x = 665324204070034724556817625597595352134835914506519509548196
96148199168011 0825
      [java]     H1y = 109574562554377037857893239701134279362432929206775373183143
0498778584109123 04
      [java] H2:
      [java]     H2x = 388743579917283958650044158343844714707433917668289612112076
5978909664265 2742
      [java]     H2y = 283133631468217395024486338236960097077966852380326432482125
840298362784 80241
      [java]
      [java] Osk:
      [java] y1  = 2499348658546980421651735253557549038342162084695211040894549807
2720325842546
      [java] y2  = 1140040453978228620432604168385489902134198543568757426592333547
31975175493025
      [java]
      [java] ----------------DONE---------------
```

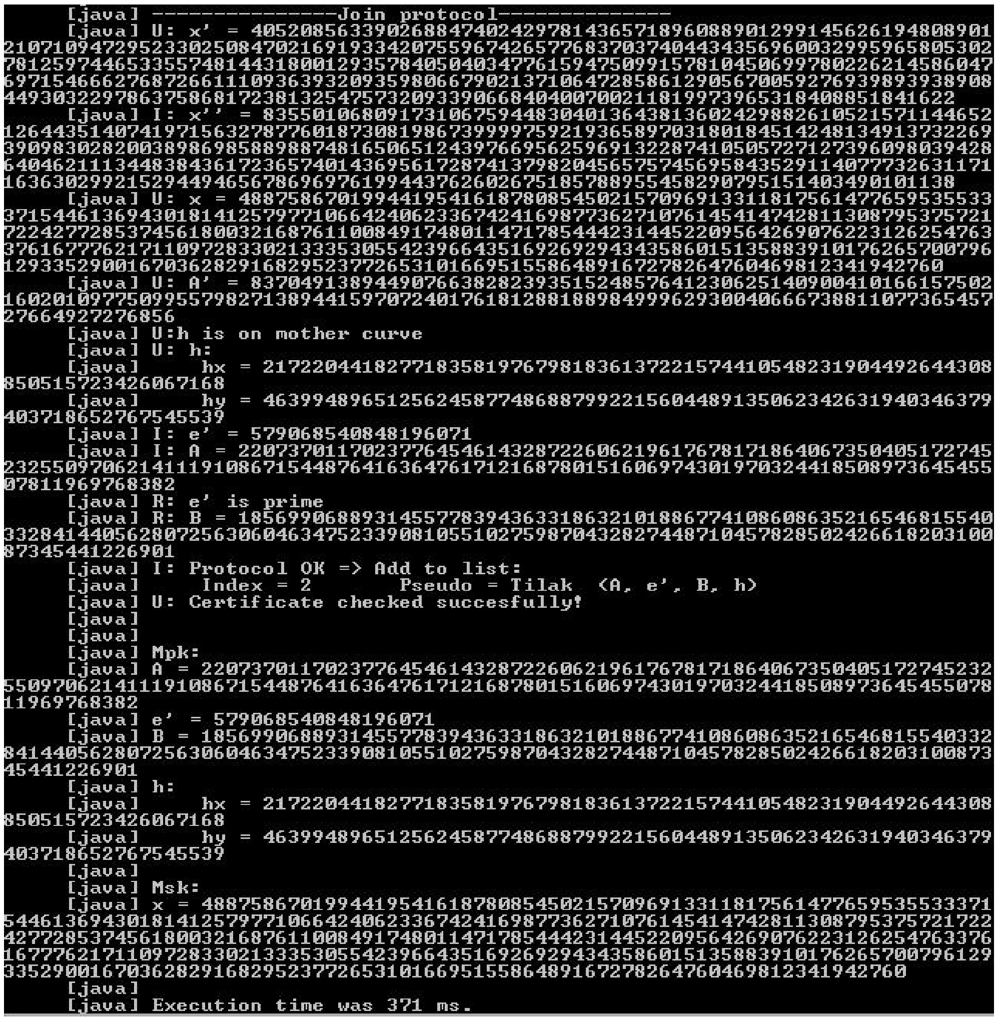
4.4 Setup Opening Manager



```
run:
      [java]
      [java] ----------------Setup RevocationManager---------------
      [java] Execution time was 2302 ms.
      [java]
      [java] Rpk:
      [java] l = 22413198871813688612182646490356073123219016407340955618573362 67
66264556989250848619137595229259863863284789364994611156872533817163255220476416
48160042409
      [java] b = 16532517382242547218531425387935690349865866941806568380591051 24
67731895954080845822483784187438526963718862431629388116897021093500427824979350
63575988546
      [java] w = 63198243461784739268448013635065579652897834001248539744756030 10
92937314960458960208609605147819886797917594477234879362955089451782655861534619
2759974378
      [java]
      [java] Rsk:
      [java] l1  = 1323959714661195703996174344013887074378893846656598051136213608
46813889234803
      [java] l2  = 1692891303535566947007621858967667530835931770097982260341032782
81368483846003
      [java]
      [java] Revocation-List = {/}
      [java]
      [java] ----------------DONE---------------
      [java]
```

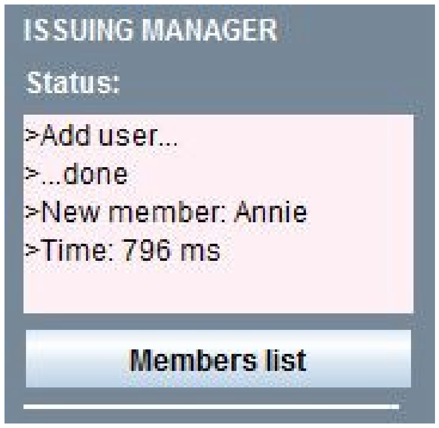
4.5 Setup Revocation Manager

4.4 Join Phase:



```
      [java] ----------------Join protocol---------------
      [java] U: x' = 405208563390268847402429781436571896088901299145626194808901
21071094729523302508470216919334207559674265776837037404434356960032995965805302
78125974465335574814431800129357840504034776159475099157810450699780226214586047
69715466627687266111093639320935980667902137106472858612905670059276939893938908
44930322978637586817238132547573209339066840400700211819973965318408851841622
      [java] I: x'' = 83550106809173106759448304013643813602429882610521571144652
12644351407419715632787760187308198673999975921936589703180184514248134913732269
39098302820038986985889887481650651243976695625969132287410505727127396098039428
64046211134483843617236574014369561728741379820456575745695843529114077732631171
16363029921529449465678696976199443762602675185788955458290795151403490101138
      [java] U: x = 4887586701994419541618780854502157096913311817561477659535533
37154461369430181412579771066424062336742416987736271076145414742811308795375721
72242772853745618003216876110084917480114717854442314452209564269076223126254763
37616777621711097283302133353055423966435169269294343586015135883910176265700796
1293352900167036282916829523772653101669515586489167278264760469812341942760
      [java] U: A' = 837049138944907663828239351524857641230625140900410166157502
16020109775099557982713894415970724017618128818898499962930040666738811077365457
27664927276856
      [java] U:h is on mother curve
      [java] U: h:
      [java]      hx = 21722044182771835819767981836137221574410548231904492644308
850515723426067168
      [java]      hy = 46399489651256245877486887992215604489135062342631940346379
403718652767545539
      [java] I: e' = 579068540848196071
      [java] I: A = 2207370117023776454614328722606219617678171864067350405172745
23255097062141119108671544876416364761712168780151606974301970324418508973645455
07811969768382
      [java] R: e' is prime
      [java] R: B = 1856990688931455778394363318632101886774108608635216546815540
33284144056280725630604634752339081055102759870432827448710457828502426618203100
87345441226901
      [java] I: Protocol OK => Add to list:
      [java]      Index = 2        Pseudo = Tilak  <A, e', B, h>
      [java] U: Certificate checked succesfully!
      [java]
      [java]
      [java] Mpk:
      [java] A = 2207370117023776454614328722606219617678171864067350405172745232
55097062141119108671544876416364761712168780151606974301970324418508973645455078
11969768382
      [java] e' = 579068540848196071
      [java] B = 1856990688931455778394363318632101886774108608635216546815540332
84144056280725630604634752339081055102759870432827448710457828502426618203100873
45441226901
      [java] h:
      [java]      hx = 21722044182771835819767981836137221574410548231904492644308
850515723426067168
      [java]      hy = 46399489651256245877486887992215604489135062342631940346379
403718652767545539
      [java]
      [java] Msk:
      [java] x = 4887586701994419541618780854502157096913311817561477659535533371
54461369430181412579771066424062336742416987736271076145414742811308795375721722
42772853745618003216876110084917480114717854442314452209564269076223126254763376
167777621711097283302133353055423966435169269294343586015135883910176265700796129
3352900167036282916829523772653101669515586489167278264760469812341942760
      [java]
      [java] Execution time was 371 ms.
```

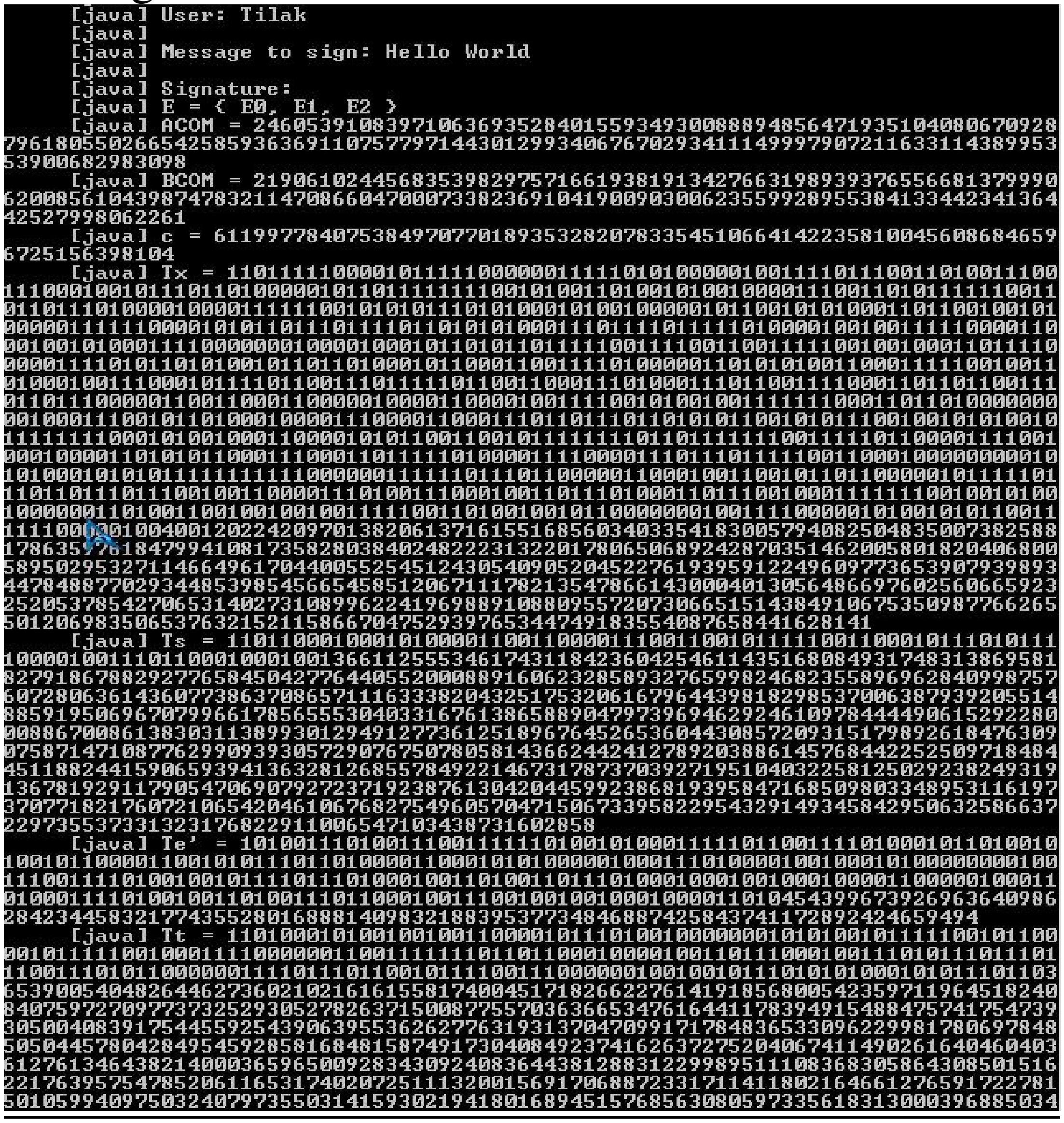
4.6 Join Phase



```
>Add user...
>...done
>New member: Annie
>Time: 796 ms
```

4.7 Issuing Manager GUI view

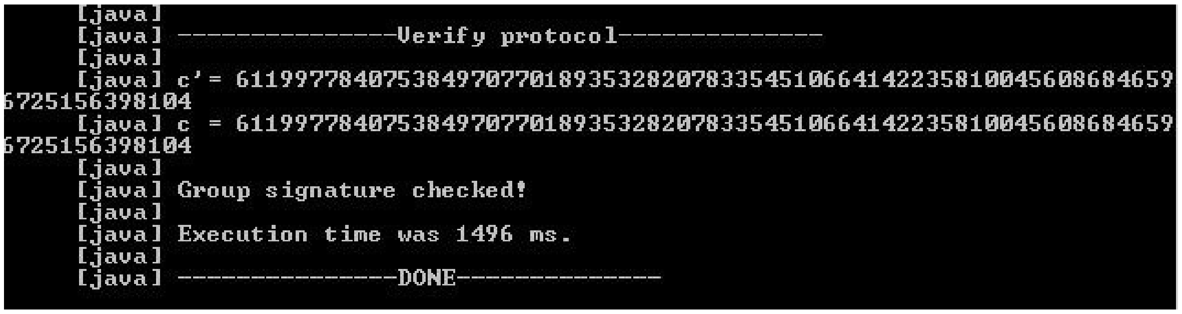
4.5 Sign:



```
     [java] User: Tilak
     [java]
     [java] Message to sign: Hello World
     [java]
     [java] Signature:
     [java] E = { E0, E1, E2 }
     [java] ACOM = 246053910839710636935284015593493008889485647193510408067092 8
79618055026654258593636911075779714430129934067670293411149997907211633114389953
53900682983098
     [java] BCOM = 21906102445683539829757166193819134276631989393765566813799 90
62008561043987478321147086604700073382369104190090300623559928955384133442341364
42527998062261
     [java] c = 61199778407538497077018935328207833545106641422358100456086846 59
6725156398104
     [java] Tx = 11011111000010111111000000111110101000000100111101110011010011100
11100010010111011010000001011011111111001010011010010100100001110011010111111 0011
01101110100001000011111100101010111010100010100100000101100101010001101100100101
00000111111000010101101110111101101010100011101111011111010000100100111110000110
00100101000111100000000100001000101101011011111001111001100111110010010001101111 0
00001111010110101001011011010001011000110011110100000110101010011000111110010011
01000100111000101111011001110111110110011000111010001110110011110001101101100111
01101110000001100110001100000010000011000010011110010100100111111100011011010000000
00100011100101101000100001110000110001110110111011010101100101011100100101010010
11111111000101001000110000101011001100101111111101101111111001111110110000111 1001
00010000110101011000111000110111110100000111100001110111011111001100010000000000 10
10100010101011111111111000000111111011101100000110001001100101101100000101111101
11011011101110010011000011101001110001001011101000110111001000111111100100101 00
10000001101001100100100100111110011010010010110000000010011110000010100101011001 1
11110000010040012022420970138206137161551685603403354183005794082504835007382588
17863597618479941081735828038402482223132201780650689242870351462005801820406800
58950295327114664961704400552545124305409052045227619395912249609773653907939893
44784887702934485398545665458512067111782135478661430004013056486697602560665923
25205378542706531402731089962241969889108809557207306651514384910675350987766265
501206983506537632152115866704752939765344749183554087658441628141
     [java] Ts = 11011000100010100000110011000001110011001011111001100010111010111
10000100111011000100010013661125553461743118423604254611435168084931748313869581
82791867882927765845042776440552000889160623285893276599824682355896962840998757
60728063614360773863708657111633382043251753206167964439818298537006387939205514
88591950696707996617856555304033167613865889047973969462924610978444490615292280
00886700861383031138993012949127736125189676452653604430857209315179892618476309
07587147108776299093930572907675078058143662442412789203886145768442252509718484
45118824415906593941363281268557849221467317873703927195104032258125029238249319
13678192911790547069079272371923876130420445992386819395847168509803348953116197
37077182176072106542046106768275496057047150673395822954329149345842950632586637
229735537331323176822911006547103438731602858
     [java] Te' = 1010011101001110011111101001010001111101100111101000101101001 0
10010110000011001010111011010000011000101010000001000111010000100100010100000000 0100
11100111101001001011110111010001001101001101110100010001001000100000110000001000 11
01000111101001001101001110110001001110010010010001000011010454399673926963640986
2842344583217743552801688814098321883953773484688742584374117289242465 9494
     [java] Tt = 11010001010010010011000010111010010000000010101001011111001011 00
00101111100100011110000000110011111110110110001000010011011100010011101011101 1101
11001110101100000011110111011001011110011100000010010010111010101000101011101103
65390054048264462736021021616155817400451718266227614191856800542359711964518240
84075972709773732529305278263715008775570363665347616441178394915488475741754739
30500408391754455925439063955362627763193137047099171784836533096229981780697848
50504457804284954592858168481587491730408492374162637275204067411490261640460403
61276134643821400036596500928343092408364438128831229989511108368305864308501516
22176395754785206116531740207251113200156917068872331711411802164661276591722781
50105994097503240797355031415930219418016894515768563080597335618313000396885034
```

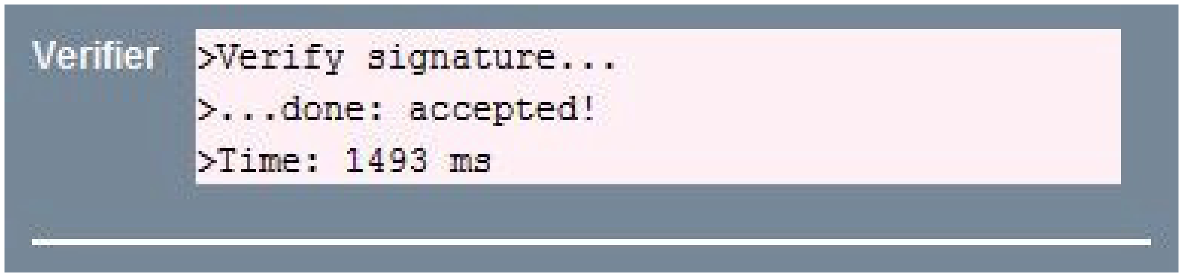
4.8 Signature Protocol

4.6 Verify: *Success*



```
     [java]
     [java] ---------------Verify protocol--------------
     [java]
     [java] c'= 6119977840753849707701893532820783354510664142235810045608684659
6725156398104
     [java] c = 6119977840753849707701893532820783354510664142235810045608684659
6725156398104
     [java]
     [java] Group signature checked!
     [java]
     [java] Execution time was 1496 ms.
     [java]
     [java] ---------------DONE--------------
```

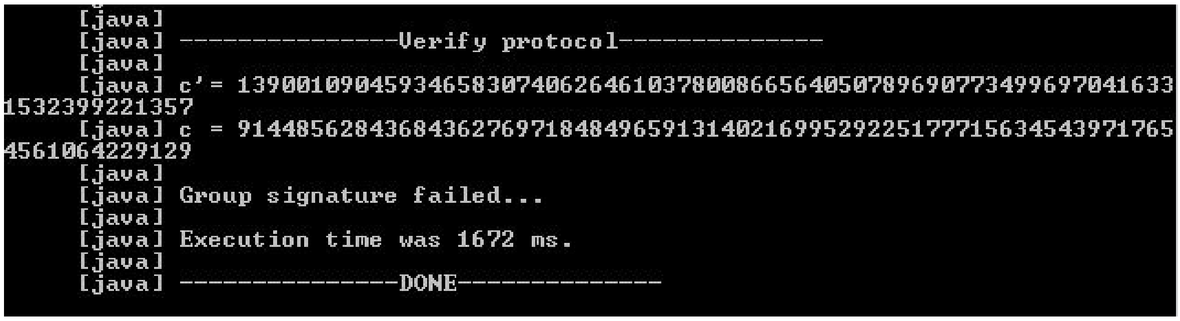
4.9 Verify Protocol: Success



```
>Verify signature...
>...done: accepted!
>Time: 1493 ms
```

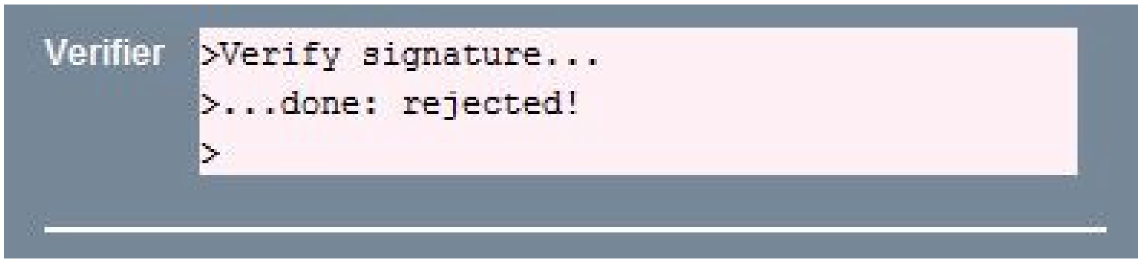
4.10 Verify Protocol: Success GUI view

4.7 Verify: *Fail*



```
     [java]
     [java] ---------------Verify protocol--------------
     [java]
     [java] c'= 1390010904593465830740626461037800866564050789690773499697041633
1532399221357
     [java] c = 9144856284368436276971848496591314021699529225177715634543971765
4561064229129
     [java]
     [java] Group signature failed...
     [java]
     [java] Execution time was 1672 ms.
     [java]
     [java] ---------------DONE--------------
```

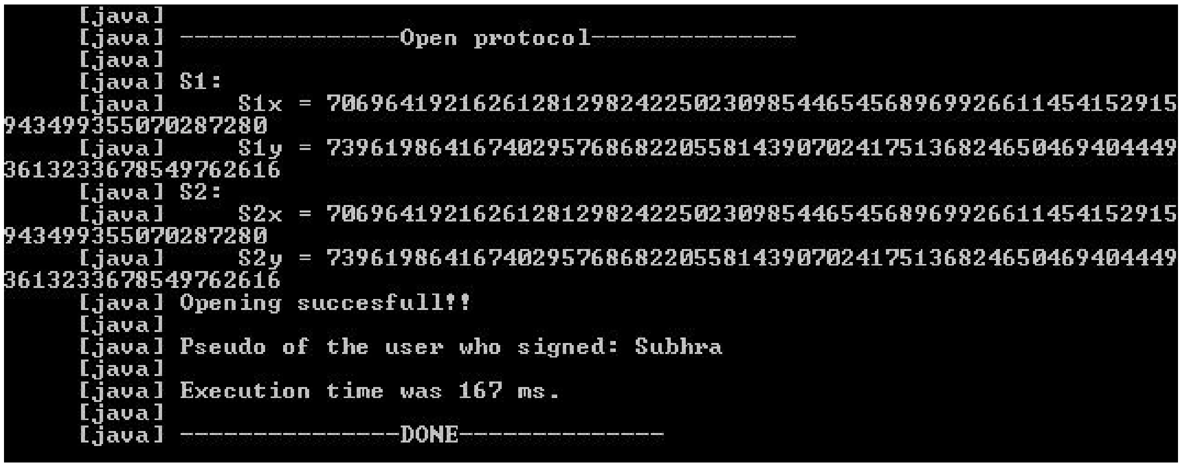
4.11 Verify Protocol: Failure



```
>Verify signature...
>...done: rejected!
>
```

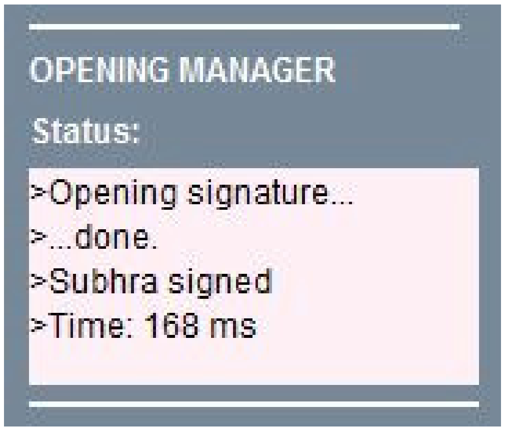
4.12 Verify Protovol: Failure GUI view

4.8 Open:



```
      [java]
      [java] ---------------Open protocol--------------
      [java]
      [java] S1:
      [java]     S1x = 7069641921626128129824225023098544654568969926611454152915
9434993550702872800
      [java]     S1y = 7396198641674029576868220558143907024175136824650469404449
3613233678549762616
      [java] S2:
      [java]     S2x = 7069641921626128129824225023098544654568969926611454152915
9434993550702872800
      [java]     S2y = 7396198641674029576868220558143907024175136824650469404449
3613233678549762616
      [java] Opening succesfull!!
      [java]
      [java] Pseudo of the user who signed: Subhra
      [java]
      [java] Execution time was 167 ms.
      [java]
      [java] ---------------DONE--------------
```

4.13 Open Protocol



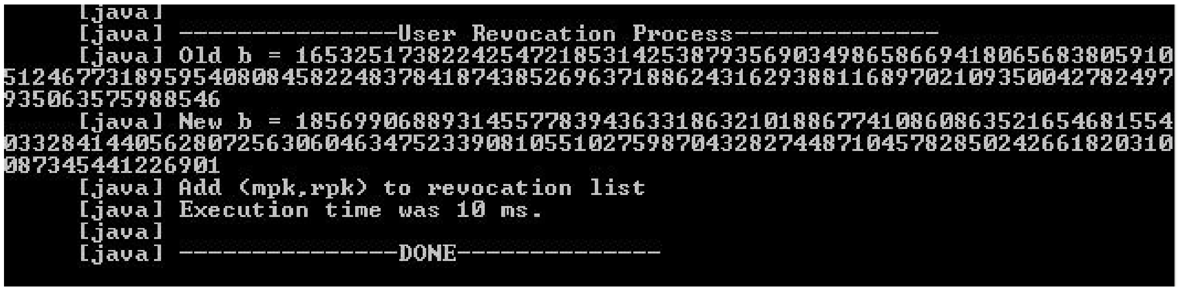
OPENING MANAGER

Status:

```
>Opening signature...
>...done.
>Subhra signed
>Time: 168 ms
```

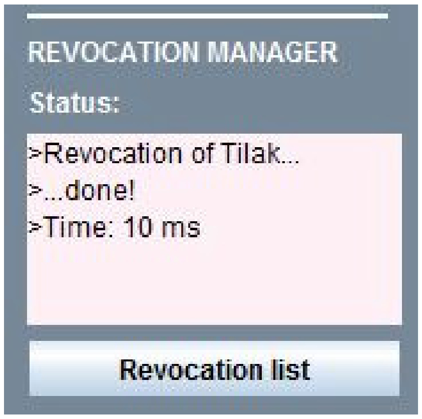
4.14 Open Protocol GUI view

4.9 Revocation:



```
     [java]
     [java] ----------------User Revocation Process---------------
     [java] Old b = 165325173822425472185314253879356903498658669418065683805910
51246773189595408084582248378418743852696371886243162938811689702109350042782497
935063575988546
     [java] New b = 185699068893145577839436331863210188677410860863521654681554
03328414405628072563060463475233908105510275987043282744871045782850242661820310
087345441226901
     [java] Add (mpk,rpk) to revocation list
     [java] Execution time was 10 ms.
     [java]
     [java] ----------------DONE---------------
```

4.15 Revocation Phase



```
>Revocation of Tilak...
>...done!
>Time: 10 ms
```

4.16 Revocation Phase GUI view

Chapter 5

Conclusion and Future Work

5.1 Conclusion and Future Work:

The Group Signature scheme proposed by Krystian Baniak in 2011[3] was successfully implemented using elliptic curve cryptography and tested to work. The scheme satisfies the standard security features of basic group signature scheme like anonymity, unforgeability, and unlinkability. Also the implemented scheme is member independent such that any member leaving or joining would not affect the signature generation scheme. This system is mostly concerned with maximal subscriber privacy and a verifiable evidence source. This scheme can also be applied in e-voting system, e-cash system and e-commerce applications. Further research is, however, needed for ensuring resistance to Revocation List Oracle corruption and ability for an adversary to infer on agent's identity knowing the date and time of given agent revocation.

Bibliography

[1] J. Y. Hwang, S. Lee, B. . Chung, H. S. Cho, and D. Nyang. Group signatures with controllable linkability for dynamic membership. Information Sciences, 222:761-778, 2013.

[2] N. Lee, T. Hwang, and C. Li. (t, n) threshold untraceable signatures.Journal of Information Science and Engineering, 16(6):835-846, 2000.

[3] J. J. Chen and Y. Liu. A traceable group signature scheme. Mathematical and Computer Modelling, 31(2-3):147-160, 2000.

[4] J. Zhang, J. Zou, and Y. Wang. An improved group signature scheme. In Lecture Notes in Computer Science, volume 3592, pages 185-194, 2005.

[5] T. Isshiki, K. Mori, K. Sako, I. Teranishi, and S. Yonezawa. Using group signatures for identity management and its implementation. In Proceedings of the Second ACM Workshop on Digital Identity Management, DIM 2006. Co-located with the 13th ACM Conference on Computer and Communications Security, CCS'06, pages 73-78, 2006.

[6] Y. He. New dynamic group signature scheme. Wuhan University Journal of Natural Sciences, 11(6):1693-1696, 2006.

[7] L. Fengyin, Y. Jiguo, and J. Hongwei. A new threshold group signature scheme based on discrete logarithm problem. In Proceedings - SNPD 2007: Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, volume 3, pages 1176-1181, 2007.

[8] Y. Geng, G. Shao, M. Zheng, and G. Cui. An improved e_cient group signature scheme for large groups. HuazhongKejiDaxueXuebao (ZiranKexue Ban)/Journal of Huazhong University of Science and Technology (Natural Science Edition), 37(7):66-69, 2009.

[9] H. Park, S. Lim, I. Yie, K. Kim, and J. Song. Strong unforgeability in group signature schemes. Computer Standards and Interfaces, 31(4):856-862, 2009.

[10] H. . Liu, W. . Xie, J. . Yu, and P. Zhang. E_ciency identity-based threshold group signature scheme. TongxinXuebao/Journal on Communications, 30(5):122-127, 2009.

[11] D. Chaum and E. van Heyst. Group signatures. Lecture Notes On Computer Science, 547(8):257-265, 1991.

[12] L. Chen and T. P. Pedersen. New group signature schemes. In A. De Santis, editor, Advances in Cryptology- EUROCRYPT'94, pages 171-181. Springer, Berlin,, 1994.

[13] Giuseppe Ateniese and Gene Tsudik. Some open issues and new directions in group signatures. pages 196-211. Springer-Verlag, 1999.

[14] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups. In Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology, CRYPTO '97, pages 410-424. Springer-Verlag, 1997.

[15] Mihir Bellare and Sara K. Miner. A forward-secure digital signature scheme. pages 431-448. Springer-Verlag, 1999.

[16] Hyun-Jeong Kim, Jong In Lim, and Dong Hoon Lee. Efficient and secure member deletion in group signature schemes. In Proceedings of the Third International Conference on Information Security and Cryptology, ICISC '00, pages 150-161. Springer-Verlag, 2001.

[17] Li-Hua Li, Chi-Yu Liu, and Min-Shiang Hwang. Cryptanalysis of an efficient secure group signature scheme. SIGOPS Oper. Syst. Rev., 38(4):66-69, October 2004.

[18] W.B. Lee and C.C. Chang. Efficient group signature scheme based on the discrete logarithm. volume 145, pages 15-18. IEE, 1998.

[19] S.J. Park S.J. Kim and D.H Won. Convertible group signatures. volume 1163, pages 311-321. Springer-Verlag, 1996.

[20] J. Camenisch. E_cient and generalized group signatures. volume 1233, pages 465-479. Springer-Verlag, 1997.

[21] Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In ACM Conference on Computer and Communications Security, pages 168-177, 2004.

[22] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In CT-RSA, pages 136{153, 2005.

[23] Cecile Delerablee and David Pointcheval. Dynamic fully anonymous short group signatures. In VIETCRYPT, pages 193-210, 2006.

[24] Fengyin Li, Jiguo Yu, and Hongwei Ju. A new threshold group signature scheme based on discrete logarithm problem. In Proceedings of the Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing - Volume 03, SNPD '07, pages 1176-1182, Washington, DC, USA, 2007. IEEE Computer Society.

[25] Yuan-Lung Yu and Tzer-Shyong Chen. An e_cient threshold group signature scheme. Applied Mathematics and Computation, 167(1):362-371, 2005.

[26] Fucai Zhou, Jun Zhang, and Jian Xu. Research on anonymous signatures and group signatures. Comput. Commun., 31(17):4199-4205, November 2008.

[27] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: formal definitions, simplified requirements, and a construction based on general assumptions. In Proceedings of the 22nd international conference on Theory and applications of cryptographic techniques, EUROCRYPT'03, pages 614-629. Springer-Verlag, 2003.

[28] Jan Camenisch and Jens Groth. Group signatures: better efficiency and new theoretical aspects. In Proceedings of the 4th international conference on Security in Communication Networks, SCN'04, pages 120-133. Springer-Verlag, 2005.

[29] R. Duran Diaz, L. Hernandez Encinas, and J. Munoz Masque. A group signature scheme based on the integer factorization and the subgroup discrete logarithm problems. In Proceedings of the 4th international conference on Computational intelligence in security for information systems, CISIS'11, pages 143-150. Springer-Verlag, 2011.

[30] Steven D. Galbraith and Mark Holmes. A non-uniform birthday problem with applications to discrete logarithms. Discrete Applied Mathematics, 160(10-11):1547-1560, 2012.

[31] David M'Raihi, David Naccache, David Pointcheval, and Serge Vaudenay. Computational alternatives to random number generators. In Fifth Annual Workshop on Selected Areas in Cryptography SAC098, volume 1556 of LNCS, pages 72{80, Kingston, Ontario, Canada, 1998. Springer.

[32] Jerey Hostein, Jill Pipher, and J.H. Silverman. An Introduction to Mathematical Cryptography. Springer Publishing Company, Incorporated, 1 edition, 2008.

[33] Behrouz A. Forouzan. Cryptography & Network Security. McGraw-Hill, Inc., 1 edition, 2008.

[34] Henk C. A. van Tilborg and Sushil Jajodia, editors. Encyclopedia of Cryptography and Security, 2nd Ed. Springer, 2011.

[35] D. Hankerson, A. Menezes, and S. Vanstone: Guide to Elliptic Curve Cryptography. 2004 Springer-Verlag New York, Inc.

[36] P. Rogaway and T. Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance, 2004.

[37] Group signature revocable anonymity scheme for network monitoring *Krystian Baniak, Institute of Telecommunications, Warsaw University of Technology, Poland :* Annales UMCS Informatica AI XI, 3 (2011) 101-115; DOI: 10.2478/v10065-011-0020-9

[38] A Survey of Group Signature Technique, its Applications and Attacks : Aayush Agarwal, Rekha Saraswat , International Journal of Engineering and Innovative Technology (IJEIT) Volume 2, Issue 10, April 2013.