

*Dynamic Group Blind Digital Signature
Scheme
A Basic Model of E-banking*

T04426



by

Asmat-e-Ayesha, Reg. # 268-FAS/MSCS/F05
Saadia Ajmal, Reg. # 273-FAS/MSCS/F05

Supervised by:

Dr. Malik Sikander Hayat Khiyal



Department of Computer Science
Faculty of Basic and Applied Sciences
International Islamic University, Islamabad

EMB

MS

005.8

ASD

C2

- 1- Internet banking
- 2- Banking and banking - Automation
- 3- Bank management ~~ASD~~ ★

MS

17-07-10

Accession No TH-4426

ASD

Department of Computer Sciences
International Islamic University, Islamabad

Final Approval


Dated: 22-09-07

It is certified that we have read the thesis report submitted by Asmat-e-Ayesha and Saadia Ajmal, Registration Number 268-FAS/MSCS/F05 and 273-FAS/MSCS/F05 respectively, and it is our judgment that this thesis is of sufficient standard to warrant its acceptance by the International Islamic University, Islamabad for the MS Degree in Computer Science.

COMMITTEE

External Examiner

Dr. Tasneem Shah
Professor,
Department of Computer Science
Air University
Islamabad



Internal Examiner

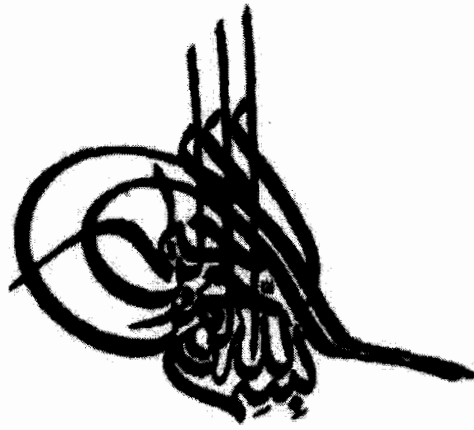
Ms. Fareeha Anwar
Lecturer,
Faculty of Applied Sciences
International Islamic University,
Islamabad, Pakistan



Supervisor

Dr. Malik Sikander Hayat Khyal
Professor, Fatima Jinnah Woman
University, Rawalpindi.





In The Name of

ALLAH ALMIGHTY

The Most Merciful The Most Beneficent

"Lo! In the creation of the heavens and the earth and the alternation of the night and the day, there are surely signs for men of understanding." (Al-Imran: 190-191)

A dissertation submitted to the
Department of Computer Science,
International Islamic University, Islamabad
as a partial fulfillment of the requirements
for the award of the degree of

MS in Computer Science

Dedication

We dedicate this work to our Dear
ALLAH T'AALAH
For all His blessings
and
for granting us with such loving relationships
Who
encouraged and helped us in our entire Life.

Asmat-e-Ayesha 268-FAS/MSCS/F05
Saadia Ajmal 273-FAS/MSCS/F05

Declaration

We hereby declare and affirm that this software neither as a whole nor as a part thereof has been copied out from any source. It is further declared that we have developed this software and accompanied report entirely on the basis of our personal efforts, made under the sincere guidance of our teachers. If any part of this project is proven to be copied out or found to be a reproduction of some other, we shall stand by the consequences.

No portion of the work presented in this report has been submitted in support of an application for other degree or qualification of this or any other University or Institute of learning.

Asmat-e-Ayesha 268-FAS/MSCS/F05
Saadia Ajmal 273-FAS/MSCS/F05

Acknowledgement

We bestow all praises to, acclamation and appreciation to Almighty Allah, The Most Merciful and Compassionate, The Most Gracious and Beneficent, Whose bounteous blessings enabled us to pursue and perceive higher ideals of life, Who bestowed us good health, courage, and knowledge to carry out and complete our work. Special thanks to our Holy Prophet Muhammad (SAW) who enabled us to recognize our Lord and Creator and brought us the real source of knowledge from Allah (SWT), the Qur'ān, and who is the role model for us in every aspect of life.

It is with genuine gratitude that we thank to our kind supervisor, **Dr. Malik Sikander Hayat Khiyal** who kept our morale high by his suggestions and appreciation. If it had not been for his constant supply of insight and encouragement, we are sure this project would not have been completed in good spirits nor would it have been completed in time.

Special thanks go to all our teachers. In particular, we would like to thank **Sir Shiraz Baig** for his dedication, inspiring attitude, untiring help and kind behavior throughout the project efforts and presentation of this manuscript.

Finally we must mention that it was mainly due to our parent's moral support and financial help during our entire academic career that enabled us to complete our work dedicatedly. We owe all our achievements to our most loving parents, who mean most to us, for their prayers are more precious before any treasure on the earth. We are also thankful to our loving brothers, sisters, and friends whose prayers have always been a source of determination for us.

Asmat-e-Ayesha 268-FAS/MSCS/F05
Saadia Ajmal 273-FAS/MSCS/F05

Project in Brief

Project Title: **Dynamic Group Blind Signature Scheme-
A Basic Model of E-banking**

Undertaken by: Asmat-e-Ayesha

 268-FAS/MSCS/F05

 Saadia Ajmal

 273-FAS/MSCS/F05

Supervised By: **Dr. Malik Sikander Hayat Khiyal**

Starting Date: October 2006

Completion Date: 2007

Tool Used: Java Development Tool Kit (jdk v 1.6), MS Access

Operating System: Microsoft Windows XP Professional

System Used: Pentium IV

Abstract

This research deals with the concept of group signature that allows a group member to sign messages anonymously on the behalf of entire group. We propose a dynamic group blind signature scheme which is an extension of Lysyanskaya and Ramzan's Group Blind signature scheme. In our scheme we introduce some properties of Dynamic group signature presented by Bellare et.al. We construct a new blind signature scheme that has two separate authorities, an issuer, for issuing the membership certificate to group members and an Opener, who can open the identity of a signature's originator in the case of a dispute. This scheme is based on the concept of PKI environment and Three Key Requirement i.e. Traceability, Anonymity and Non Frameability.

TABLE OF CONTENTS

Chapter No	Contents	Page No
1.	INTRODUCTION	1
1.1	CRYPTOGRAPHY.....	1
1.1.1	Encryption.....	2
1.2	AUTHENTICATION	3
1.3	DIGITAL SIGNATURES.....	4
1.3.1	How Digital Signature Technology Works.....	5
1.3.2	Digitally Signing Messages	7
1.3.3	Verifying Digital Signatures.....	8
1.3.4	Reasons for Invalid Signatures	10
1.3.5	Implementation of public-key digital signatures	11
1.3.6	Association of digital signatures and encryption.....	12
1.3.7	Application of Digital signature.....	13
1.3.8	Digital Signature Function and Blind Signatures	14
1.4	PKI	15
1.5	PUBLIC KEY CERTIFICATES.....	16
2.	LITERATURE SURVEY.....	18
2.1	FOUNDATIONS OF GROUP SIGNATURES: THE CASE OF DYNAMIC GROUPS BY ... MIHIR BELLARE, HAIXIA SHIY AND CHONG ZHANGZ	19
2.1.1	Disadvantages.....	21
2.2	GROUP BLIND DIGITAL SIGNATURES: A SCALABLE SOLUTION TO ELECTRONIC CASH. BY ANNA LYSYANSKAYA AND ZULFIQAR RAMZAN	22
2.2.1	Disadvantages:	22
2.3	FOUNDATIONS OF GROUP SIGNATURES: FORMAL DEFINITIONS BY M BALLARE,D. MICCIANCIO,BOGDAN WARINSCHI.....	23
2.3.1	Definition for security Requirement	23
2.3.2	Their Scheme	23
2.3.3	Construction of their scheme	24
2.3.4	Security Results:	24
2.3.5	Disadvantages of the scheme.....	24
2.3.6	Future Work.....	25
2.4	GROUP SIGNATURE SCHEME BASED ON AN RSA-VARIANT BY JAN C AND MARKUS MICHELS	25
2.4.1	Previous work:	25
2.4.2	An Approach for a Realization:	26
2.4.3	Working of Proposed scheme:	26
2.4.4	Efficiency Analysis.....	27
2.4.5	Conclusion	27

Chapter No	Contents	Page No
2.5	EFFICIENT GROUP SIGNATURE SCHEMES: FOR LARGE GROUPS. BY JAN CAMENSICH AND MARKUS STADLER	27
2.5.1	Construction of Scheme	28
2.6	PROBLEM DOMAIN AND PROPOSED SOLUTION.....	30
2.6.1	Problem Statement:.....	30
2.6.2	PROPOSED SOLUTION:.....	30
3.	RESEARCH METHODOLOGIES	34
3.1	GROUP SIGNATURE SCHEME	34
3.2	THE GROUP BLIND DIGITAL SIGNATURE MODEL.....	34
3.2.1	Security Requirements for Group Blind Digital Signature.....	34
3.2.2	Procedures Allowed by Group Blind Digital Signatures	35
3.3	OUR SCHEME.....	36
3.3.1	Entities Involved in our Scheme	36
3.3.2	Assumption of our Scheme.....	37
3.3.3	Notations.....	38
3.3.4	Key Usage.....	39
3.3.5	Algorithms of Our Scheme.....	41
4.	IMPLEMENTATION	45
4.1	JAVA	45
4.2	MS ACCESS.....	46
4.3	IMPLEMENTATION OF DYNAMIC GROUP BLIND SIGNATURE MODEL.....	46
4.4	Applying Dynamic Group Blind Digital Signatures to Electronic Banking.....	47
4.4.1	Set Up Phase	48
4.4.2	Withdrawal	52
4.4.3	Deposit	55
4.4.4	Anonymity Revocation	56
5.	RESULTS.....	58
6.	CONCLUSION AND FUTURE ENHANCEMENT.....	61
	REFERENCES.....	62
	APPENDIX	
	E-banking System.....	64

Chapter 1



Introduction

1. Introduction

Data communication over network is connection of a computer system to another computer system so that only the intended recipient receives and reads the message and the message received is identical to the message sent. The message would not be identical if it was altered in anyway, whether transmitted over faulty channels or intercepted by an eavesdropper. Transmission security translates into secure networks. Although many people regard networks as computers connected by wires, this definition of a network, while technically correct, misses the point. Rather, networks are transmitted data, the data flowing over wires.

All transmissions can be intercepted. And the cautious user looks at all transmissions as if they will be intercepted. You can minimize the risks of transmission interception, but you can never, under any circumstances, completely rule it out. After all, it is people who design and put wires in their place, and people can get to them. Accessing wires is somewhat comparable, although much more difficult, to accessing a transmission sent over airwaves, as on a CB radio. For example, as a ham, you may have a message intended only for other hams. Although hams are the main communicators on these frequencies, anyone with the right radio equipment can tune in and listen, so it's likely your message will be received and heard by other listeners who pick up the frequency, whether you want them to hear it or not.

1.1 Cryptography

As the field of cryptography has advanced, the dividing lines for what is and what is not cryptography have become blurred. Cryptography today might be summed up as the study of techniques and applications that depend on the existence of difficult problems. Cryptanalysis is the study of how to compromise (defeat) cryptographic mechanisms, and cryptology (from the Greek *kryptós logos*, meaning "hidden word") is the discipline of cryptography and cryptanalysis combined. To most people, cryptography is concerned with keeping communications private. Indeed, the protection of sensitive

communications has been the emphasis of cryptography throughout much of its history.

1.1.1 Encryption

Encryption is the transformation of data into a form that is as close to impossible as possible to read without the appropriate knowledge. Its purpose is to ensure privacy by keeping information hidden from anyone for whom it is not intended, even those who have access to the encrypted data. Decryption is the reverse of encryption; it is the transformation of encrypted data back into an intelligible form.

Encryption and decryption generally require the use of some secret information, referred to as a key. For some encryption mechanisms, the same key is used for both encryption and decryption; for other mechanisms, the keys used for encryption and decryption are different.

Private Key Encryption

Simple form of encryption is commonly known as private key or symmetric encryption. It's called private key encryption because each party must know before the message is sent how to interpret the message. For example, spies in the movies always have a sequence of statements that they exchange to be sure of each other's identity, like "the sun is shining" must be followed by "the ice is still slippery." This is an example of encrypting so that only the person for whom a message is intended will understand it.

Other systems have been developed so that information can be encrypted in a general way. Again, using history as an example, one encryption method is commonly referred to as Caesar's code. According to history, Caesar would send messages that were encoded by replacing each letter in the message with the letter three places higher in the alphabet (A was replaced by D, B by E, and so on). The recipient just had to change the letters back to find out what the message said. An enemy who intercepted the message and did not know the method of encoding it would be unable to decipher it. Clearly though, this encoding method is not terribly difficult to break. This is called private key

encryption because the method of encryption must be kept quiet. Anyone who knows the method could decode the message. It also is called symmetric because the same key is used to both encrypt and decrypt the message. Other private key methods have been devised to be more difficult to break.

Public Key Encryption

To overcome the drawbacks of private key systems, mathematicians have invented public key systems. Unknown until about 30 years ago, public key systems were developed from some very subtle insights about the mathematics of large numbers and how they relate to the power of computers. Public key means that anyone can publish his or her method of encryption, publish a key for his or her messages, and only the recipient can read the messages. This works because of what is known in math as a trapdoor problem. A trapdoor is a mathematical formula that is easy to work forward but very hard to work backward. In general it is easy to multiply two very large numbers together, but it is very difficult to take a very large number and find its two prime factors. Public key algorithms depend on a person publishing a large public key and others being unable to factor this public key into its component parts. Because the creator of the key knows the factors of his or her large number, he or she can use those factors to decode messages created by others using his or her public key. Those who only know the public key will be unable to discover the private key, because of the difficulty of factoring the large number.

The RSA Public Key Cryptography

The RSA Public Key Cryptography was invented by Ronald Rivest, Adi Shamir, and Leonard Adelman in 1978. The trick here is that encrypted messages can be passed from the sender to the receiver, and they can be decrypted by the receiver, without having to pass a secret decryption key between them.

Instead a public/private key pair is used. The public key, which can be safely published for all to know, is used to encrypt the message. The private key, which is held by the owner, and which is never shown to anybody, is used to decrypt the message.

1.2 Authentication

Today's cryptography is more than encryption and decryption. Authentication is as fundamentally a part of our lives as privacy. We use authentication throughout our everyday lives - when we sign our name to some document for instance - and, as we move to a world where our decisions and agreements are communicated electronically, we need to have electronic techniques for providing authentication.

Cryptography provides mechanisms for such procedures. A digital signature binds a document to the possessor of a particular key, while a digital timestamp binds a document to its creation at a particular time. These cryptographic mechanisms can be used to control access to a shared disk drive, a high security installation, or a pay-per-view TV channel.

The field of cryptography encompasses other uses as well. With just a few basic cryptographic tools, it is possible to build elaborate schemes and protocols that allow us to pay using electronic money to prove we know certain information without revealing the information itself, and to share a secret quantity in such a way that a subset of the shares can reconstruct the secret.

1.3 Digital signatures

A cryptographic primitive which is fundamental in authentication, authorization, and non repudiation is the *digital signature*. The purpose of a digital signature is to provide a means for an entity to bind its identity to a piece of information. The process of *signing* entails transforming the message and some secret information held by the entity into a tag called a *signature*. [1]

The starting point for this approach is the digital signature, first proposed in 1976 by Whitfield Diffie, then at Stanford University. A digital signature transforms the message that is signed so that anyone who reads it can be sure of who sent it[18]These signatures employ a secret key used to sign messages and a public one

used to verify them. Only a message signed with the private key can be verified by means of the public one. Thus, if Alice wants to send a signed message to Bob (these two are the cryptographic community's favorite hypothetical characters), she transforms it using her private key, and he applies her public key to make sure that it was she who sent it. The best methods known for producing forged signatures would require many years, even using computers billions of times faster than those now available.

To see how digital signatures can provide all manner of unforgeable credentials and other services, consider how it might be used to provide an electronic replacement for cash. The First Digital Bank would offer electronic bank notes: messages signed using a particular private key. All messages bearing one key might be worth a dollar, all those bearing a different key five dollars, and so on for whatever denominations were needed. These electronic bank notes could be authenticated using the corresponding public key, which the bank has made a matter of record. First Digital would also make public a key to authenticate electronic documents sent from the bank to its customers.

The differences between digital signatures and other electronic signatures are significant, not only in terms of process and result, but also because those differences make digital signatures more serviceable for legal purposes. However, some electronic signatures, though perhaps legally recognizable as signatures, may not be as secure as digital signatures, and may lead to uncertainty and disputes.

1.3.1 How Digital Signature Technology Works

Digital signature works on the principle of Public Key Infrastructure (PKI) — cryptography based on a concept of key pairs. Public and private key pairs are nothing but large prime numbers generated by a mathematical algorithm. Public key of an individual is made known to receivers while the private key is kept confidential. Private key helps to prove unequivocally that you are who you claim to be.

PKI serves four functions to secure electronic transactions — authentication,

confidentiality, integrity and legal non-repudiation. Three different functions serve to achieve authentication, confidentiality and integrity

Digital signatures are created and verified by means of cryptography, the branch of applied mathematics that concerns itself with transforming messages into seemingly unintelligible forms and back again. For digital signatures, two different keys are generally used, one for creating a digital signature or transforming data into a seemingly unintelligible form, and another key for verifying a digital signature or returning the message to its original form. Computer equipment and software utilizing two such keys is often termed an "**asymmetric cryptosystem**".

The keys of an asymmetric cryptosystem for digital signatures are termed the **private key**, which is known only to the signer and used to create the digital signature, and the **public key**, which is ordinarily more widely known and is used to verify the digital signature. A recipient must have the corresponding public key in order to verify that a digital signature is the signer's. If many people need to verify the signer's digital signatures, the public key must be distributed to all of them, perhaps by publication in an on-line repository or directory where they can easily obtain it

The processes of creating a digital signature and verifying it accomplish the essential effects desired of a signature:

Signer authentication: If a public and private key pair is associated with an identified signer as described below, a digital signature by the private key effectively identifies the signer with the message. The digital signature cannot be forged by a person other than the proper signer, unless the proper signer loses control of the private key, such as by divulging it or losing a computer-readable card and its associated personal identification number (PIN) or pass phrase.

Message authentication: The process of digitally signing also identifies the matter to be signed, typically with far greater certainty and precision than paper signatures. Verification also reveals any tampering with the message, since processing the hash

results (one made at signing and the other made at verifying) discloses whether the message is the same as when signed.

Affirmative act: Creating a digital signature requires the signer to provide her private key and invoke a software function to create a digital signature. This act can be the basis of a ceremony and can be used in staging the completion of a transaction.

Efficiency: The processes of creating and verifying a digital signature provide a high level of assurance that the digital signature is genuinely the signer's and are almost entirely automated or capable of automation. They can be set up to run with great speed and accuracy, with human interaction only for non-routine processing decisions. Compared to paper methods such as checking bank signature cards, methods so impracticable that they are rarely actually used, digital signatures yield a high degree of assurance without adding greatly to the resources required for processing.

1.3.2 Digitally Signing Messages

Public key cryptography gives a reliable method for digital signing and signature verification based on public/private key pairs. A person can sign a given digital message (file, document, e-mail, and so forth) with his private key. From a technical point of view, the **digital signing of a message** is performed in two steps:



Step 1: Calculate the Message Digest

In the first step of the process, a **hash-value** of the message (often called the **message digest**) is calculated by applying some cryptographic **hashing algorithm** (for example, MD2, MD4, MD5, SHA1, or other). The calculated hash-value of a message is a sequence of bits, usually with a fixed length, extracted in some manner from the message.

All reliable algorithms for message digest calculation apply such mathematical transformations that when just a single bit from the input message is changed, a completely different digest is obtained. Due to this behavior, these algorithms are very steady in cryptanalytical attacks; in other words, it is almost impossible, from a given hash-value of a given message, to find the message itself. This impossibility for retrieval of the input message is pretty logical if we take into account that a hash-value of a message could have a hundred times smaller size than the input message. Actually, the computing resources needed to find a message by its digest are so huge that, practically, it is unfeasible to do it.

It is also interesting to know that, theoretically, it is possible for two entirely different messages to have the same hash-value calculated by some hashing algorithm, but the probability for this to happen is so small that in practice it is ignored.

Step 2: Calculate the Digital Signature

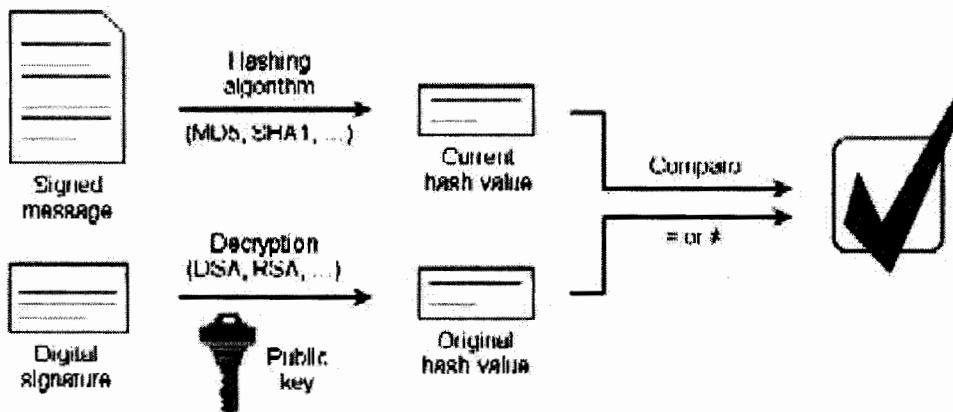
In the second step of digitally signing a message, the information obtained in the first step hash-value of the message (the message digest) is encrypted with the private key of the person who signs the message and thus an encrypted hash-value, also called **digital signature**, is obtained. For this purpose, some mathematical cryptographic **encrypting algorithm** for calculating digital signatures from given message digest is used. The most often used algorithms are **RSA** (based on the number theory), **DSA** (based on the theory of the discrete logarithms), and **ECDSA** (based on the elliptic curves theory). Often, the obtained digital signature is attached to the message in a special format to be verified later if it is necessary.

1.3.3 Verifying Digital Signatures

Digital signature technology allows the recipient of given signed message to verify its real origin and its integrity. The process of **digital signature verification** is purposed to ascertain if a given message has been signed by the private key that corresponds to a given public key. The digital signature verification cannot ascertain whether the

given message has been signed by a given person. If we need to check whether some person has signed a given message, we need to obtain his real public key in some manner. This is possible either by getting the public key in a secure way (for example, on a floppy disk or CD) or with the help of the Public Key Infrastructure by means of a digital certificate. Without having a secure way to obtain the real public key of given person, we don't have a possibility to check whether the given message is really signed by this person.

From a technical point of view, the verification of a digital signature is performed in three steps:



Step 1: Calculate the Current Hash-Value

In the first step, a hash-value of the signed message is calculated. For this calculation, the same hashing algorithm is used as was used during the signing process. The obtained hash-value is called the **current hash-value** because it is calculated from the current state of the message.

Step 2: Calculate the Original Hash-Value

In the second step of the digital signature verification process, the digital signature is decrypted with the same encryption algorithm that was used during the signing process. The decryption is done by the public key that corresponds to the private key used during the signing of the message. As a result, we obtain the **original hash-value** that

was calculated from the original message during the first step of the signing process (the original message digests).

Step 3: Compare the Current and the Original Hash-Values

In the third step, we compare the current hash-value obtained in the first step with the original hash-value obtained in the second step. If the two values are identical, the verification is successful and proves that the message has been signed with the private key that corresponds to the public key used in the verification process. If the two values differ from one another, this means that the digital signature is invalid and the verification is unsuccessful.

1.3.4 Reasons for Invalid Signatures

There are three possible reasons for getting an invalid digital signature:

If the digital signature is adulterated (it is not real) and is decrypted with the public key, the obtained original value will not be the original hash-value of the original message but some other value.

If the message was changed (adulterated) after its signing, the current hash-value calculated from this adulterated message will differ from the original hash-value because the two different messages correspond to different hash-values. If the public key does not correspond to the private key used for signing, the original hash-value obtained by decrypting the signature with an incorrect key will not be the correct one.

If the verification fails, in spite of the cause, this proves only one thing: The signature that is being verified was not obtained by signing the message that is being verified with the private key that corresponds to the public key used for the verification. Unsuccessful verification does not always mean that an attempt for digital signature adulteration is detected. Sometimes, verification could fail because an invalid public key is used. Such a situation could be obtained when the message is not sent by the person who was expected to send it or when the signature verification system has an incorrect public key for this

person. It is even possible for one person to own several different valid public keys along with valid certificates for each of them and the system attempted to verify a message received from this person with some of these public keys but not with the correct one (the key corresponding to the private key used for signing the message). In order for such problems to be avoided, most often when a signed document is sent, the certificate of the signer is also sent along with this document and the corresponding digital signature. Thus, during the verification, the public key contained in the received certificate is used for signature verification; if the verification is successful, it is considered that the document is signed by the person who owns the certificate. Of course, it is always necessary that, when certificates are used, we should believe the certificate only if its validity is verified or the certificate is self-signed but is obtained from the sender in a secure way (not from the Internet).[2]

1.3.5 Implementation of public-key digital signatures

Public-key digital signature schemes rely on public-key cryptography. In public-key cryptography, each user has a pair of keys: one public and one private. The public key is distributed freely, but the private key is kept secret by the user; another requirement is that it should be computationally infeasible to derive the private key from the public key.[3]

Generally, digital signature schemes include three algorithms:

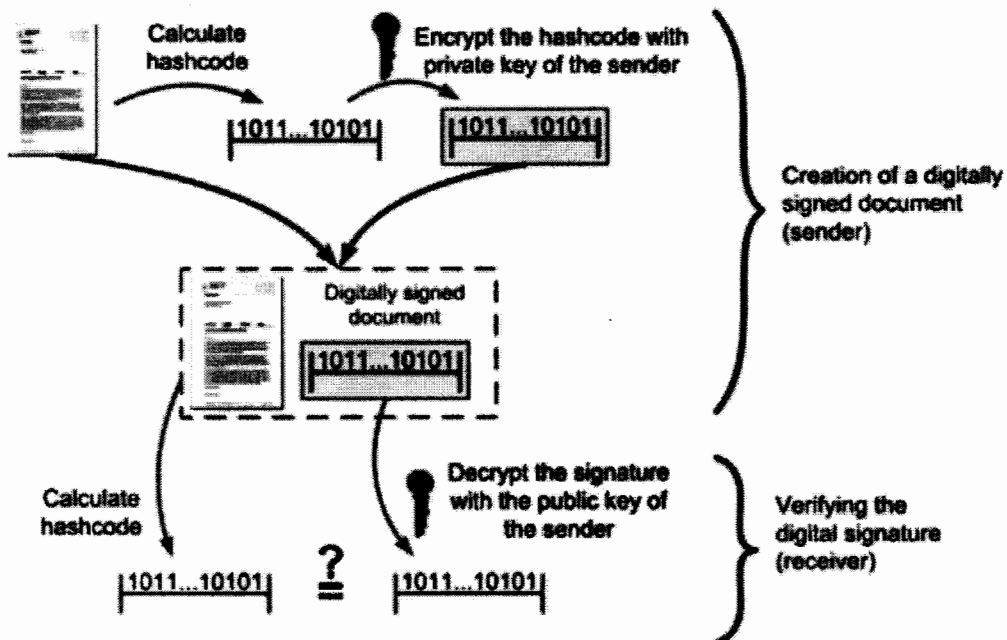
- A key generation algorithm
- A signing algorithm
- A verification algorithm

For example, consider a situation in which Bob sends a message to Alice and she wants to be certain it came from him. Bob sends his message to Alice, attaching a digital signature. The digital signature was generated using Bob's private key, and takes the form of a string of bits (normally represented as a string of characters (ie, digits and letters)). On receipt, Alice can then check whether the message really came from Bob by running

the verification algorithm on the message together with the signature, using Bob's public key. If they match, then Alice can be confident the message really was from Bob, because quality digital signature algorithms are so designed that it is very difficult to forge a signature to match a given message (unless one has knowledge of the private key, which Bob must keep secret).

1.3.6 Association of digital signatures and encryption

Creating and verifying a digital signature



If the calculated hashcode does not match the result of the decrypted signature, either the document was changed after signing, or the signature was not generated with the private key of the alleged sender.

Digital signatures use encryption techniques but the algorithms are not typically suited for direct encryption of bulk plaintexts; more efficient methods are available. Of course a signed document may be sent encrypted over a public communication channel (eg, the Internet) just as might be any other message.

More usually, Bob first applies a quality cryptographic hash function to the message, and then digitally signs the resulting hash (see diagram). An insecure hash can compromise the digital signature. For example, if it is possible to generate hash collisions, it might be

feasible to forge digital signatures[3].

1.3.7 Application of Digital signature

E-cash

In real life several types of payment systems are used frequently. Cash payment is probably the most frequently used money transfer system in the world. Coins and notes can be exchanged easily between customers and it requires no intervention from a financial institution. Because it is quite awkward to use cash when amounts get larger, banks have created checks that represent money. From a customer point of view checks are easier to carry, but cashing a check enquires intervention from a financial institution. Since the 1950's payment cards were introduced to tackle the considerable risk of using checks.

First it is important to identify the different parties that are involved in a typical electronic cash payment scenario. We can

distinguish three players here:

- the **payer** or customer, which wants to buy certain products or use particular services
- the **payee** or merchant, which delivers the products/services
- a **bank** that plays the role of a financial institution

Now, what do we mean exactly when we talk about the anonymity of a payment system?

Based on the literature we define payment system anonymity to consist of the following properties:

- **payer anonymity**: guarantees that by using that particular payment system, the identity of the payer cannot be revealed to any other party.
- **payee anonymity**: this property is the same as the previous property, except for the party that is involved in it. In this case it is the identity of the payee that should not be revealed.
- **payment untraceability**: this property ensures that nor the bank nor any other party can find out whose money is spent in a particular payment. Moreover, one cannot determine whether any two payment transactions originate from the same user or not. The latter

makes it impossible to keep track of the buying habits of the payer.

In a practical system however it is often not advisable to have payee anonymity. Customers don't want to spend their money to an anonymous identity, since the risk of being deceived is too high. Therefore from now on, when we talk about anonymity we only have in mind payer anonymity and payment untraceability.[4]

1.3.8 Digital Signature Function and Blind Signatures

Authentication schemes are used to guarantee that communication is done with the right persons or devices. In contrast, digital signatures allow to prove the authenticity of messages (Diffie, Hellman 1976). An issuer can use a digital Signature to sign electronic coins. Chaum suggested to —blind “ the signatures for making untraceable payments

Blind Digital Signatures:

Blind signature schemes are the central cryptographic component of digital-cash schemes, used to enable a Withdrawer to obtain a Bank's signature on some token without revealing this token to the bank, thereby creating a valid but anonymous e-coin. The concept of a Blind Digital Signature was introduced by Chaum (1983,1989).to enable spender anonymity in Electronic Cash systems. Such signatures require that a signer be able to sign a document without knowing its contents. Moreover, should the signer ever see the document/signature pair, he should not be able to determine when or for whom he signed it (even though he can verify that the signature is indeed valid). This intuitively corresponds to signing a document with your eyes closed. If you happen to see the document and signature later on, you can indeed verify that the signature is yours, but you will probably have great difficulty in recollecting when or for whom you signed the original document. At first this concept seems a little strange ,why would you want to sign something without seeing it? It turns out that, when applied properly, this notion has some very nice applications in situations where anonymity is a big issue. Two such applications are online voting and electronic cash. When you submit an online vote, you might like for that vote to be anonymous so no one can tell whom you voted for.

Similarly with electronic cash, you might not want someone else to know who you are when you spend it. This is similar to normal paper cash, when you make a purchase, the vendor more or less has no idea who you are, but he can probably tell whether the money you gave him is legitimate. Specifically, in this electronic cash scenario, a document corresponds to an electronic coin or note, and the signer represents a bank. The spender retains anonymity in any transaction that involves electronic coins if they are blindly signed. [1]

Group Digital Signatures

In a group digital signature scheme, members of a given group are allowed to digitally sign a document on behalf of the entire group. In addition, the signatures can be verified using a single group public key. Also, once a document is signed, no one, except a designated group manager, can determine which particular group member actually signed it. Companies can use group signatures to validate price lists, press releases, or digital contracts; customers would only need to know a single company public key to verify signatures. Companies are then capable of concealing their internal structure, while still being able to determine which employee signed a given document. Group Digital Signatures were first introduced and implemented by Chaum and van Heyst [14].

1.4 PKI

A public-key infrastructure (PKI) consists of protocols, services, and standards supporting applications of public-key cryptography. PKI sometimes refers simply to a trust hierarchy based on public-key certificates [1], and in other contexts embraces encryption and digital signature services provided to end-user applications as well. A middle view is that a PKI includes services and protocols for managing public keys, often through the use of Certification Authority (CA) and Registration Authority (RA) components, but not necessarily for performing cryptographic operations with the keys. Among the services likely to be found in a PKI are the following:

Key registration: issuing a new certificate for a public key.

Certificate revocation: canceling a previously issued certificate.

Key selection: obtaining a party's public key.

Trust evaluation: determining whether a certificate is valid and what operations it authorizes.

Key recovery has also been suggested as a possible aspect of a PKI.

1.5 Public Key Certificates

To verify a digital signature, the verifier must obtain a public key and have assurance that that public key corresponds to the signer's private key. However, a public and private key pair has no intrinsic association with any person; it is simply a pair of numbers. The association between a particular person and key pair must be made by people using the fact-finding capabilities of their senses.

In a transaction involving two parties, for example, the parties could bilaterally identify each other with the key pair each party will use, but making such an identification is no small task, especially when the parties are geographically distant from each other, communicate over an open, insecure information network, are not natural persons but rather corporations or similar artificial entities, and act through agents whose authority must be ascertained. Since reliably identifying a remote party involves considerable effort, establishing a remote party's digital signature capability specifically for each of many transactions is inefficient. Instead, a prospective digital signer will often wish to identify itself with a key pair and reuse that identification in multiple transactions over a period of time. To that end, a prospective signer could issue a statement such as: "Signatures verifiable by the following public key are mine". However, others doing business with the signer may well be unwilling to take the signer's own purported word for its identification with the key pair. Especially for electronic transactions made over

worldwide information networks rather than face to face, a party would run a great risk of dealing with a phantom or an impostor, or of facing a disavowal of a digital signature by claiming it to be the work of an impostor, particularly if a transaction proves disadvantageous for the purported signer. To assure that each party is indeed identified with a particular key pair, one or more third parties trusted by both of the others must associate an identified person on one end of the transaction with the key pair creating the digital signature received at the other end, and vice versa. That trusted third party is termed a "certification authority" in the ABA Guidelines, the Utah Act, and most technical standards.

To associate a key pair with a prospective signer, a certification authority issues a certificate, an electronic record that sets forth a public key and represents that the prospective signer identified in the certificate holds the corresponding private key. That prospective signer is termed the "subscriber". Thus, a certificate's principal function is to identify a key pair with a subscriber, so that a person verifying a digital signature by the public key listed in the certificate can have assurance that the corresponding private key is held by the subscriber also listed in the certificate.

To make a public key and its identification with a specific subscriber readily available for use in verification, the certificate may be published in a repository. Repositories are on-line databases of certificates available for retrieval and use in verifying digital signatures. Often, retrieval is accomplished automatically by having the verification program inquire of the repository to obtain certificates as needed.

Once issued, a certificate may prove to be unreliable, such as in situations where the subscriber misrepresents his identity to the certification authority. In other situations, a certificate may be reliable enough when issued but come to be unreliable sometime thereafter. For example, if the subscriber loses control of the private key, the certificate becomes unreliable, since digital signatures created by the lost private key would appear to be the subscriber's according to the certificate.

Chapter 2



Literature Survey

2. Literature Survey

Digital Signature Schemes enable people to electronically "sign" their documents in a secure and efficient manner. In other words, it is difficult to forge the signatures, yet verifying the validity of the digital signature is easy. The first construction of digital signature based on a number-theoretic assumption was given by Rivest, Shamir and Adleman [5]. The formal definitions of security for digital signatures were first outlined by Goldwasser, Micali, and Rivest [6]. They discussed the notion of an existential adaptive chosen-message attack which is the strongest form of possible attack one could imagine on a digital signature

An interesting variant on the basic digital signature is the blind digital signature. The concept of a Blind Digital Signature was introduced by Chaum [12] to enable spender anonymity in Electronic Cash systems. Such signatures require that a signer be able to sign a document without knowing its contents. Moreover, should the signer ever see the document/signature pair, he should not be able to determine when or for whom he signed it (even though he can verify that the signature is indeed valid). Intuitively, this corresponds to signing a document with your eyes closed. In the electronic cash scenario, a document corresponds to an electronic coin, and the signer represents a bank. The spender retains anonymity in any transaction that involves electronic coins since they are blindly signed. We give more formal protocols later in this chapter. A natural concern with cryptographic schemes is showing that they are secure.

The Group Digital Signature extends the traditional Digital Signature concept to a multi-party setting. In a group digital signature scheme, members of a given group are allowed to sign on behalf of the entire group. In addition, the signatures can be verified using a single group public key. Also, once a document is signed, no one, except for a designated group manager, should be able to determine which particular group member actually signed it. Group signatures should also be designed so that no group member can forge another member's signature on a given document.

For example, suppose you are the CEO of a company and you want some of your individual employees to validate price lists, press releases, or digital contracts on behalf of the entire company. In this case, you can set up a group signature scheme, and act as the group manager. Then your employees can sign or validate various documents on behalf of the entire company. By using this approach, you will conceal your company's internal structure, and your customers would only have to know a single company public key to verify the signatures on your documents. Moreover, only you can determine which employee signed which document. Additionally, the signatures would satisfy various security requirements of interest. Among other things, the signatures would be unforgeable, and it would be next to impossible for one of your employees to "fake" the signature of another one of your employees. In fact, it would also be impossible for the group manager to fake the signatures of the individual group members. This is just one simple application of group digital signatures. The group blind digital signature scheme given in this thesis sheds light on new applications of group digital signatures to electronic commerce protocols.

2.1 Foundations of Group Signatures: The Case of Dynamic Groups by Mihir Bellare, Haixia Shiy and Chong Zhangz [8]

In this paper Bellare. et. al.[8] presents a novel idea of introducing a new case in group signature schemes where Groups are dynamic in nature. However the bulk of existing practical schemes and applications are for dynamic groups, and these involve important new elements and security issues. This paper treats this case, providing foundations for dynamic group signatures, in the form of a model, strong formal definitions of security, and a construction proven secure under general assumptions. This is an important and useful step because it helps bridge the gap between and the previous practical work, and delivers a basis on which existing practical schemes may in future be evaluated or proven secure. They provide an extension of the existing treatment of static group to the dynamic case. This extension does not seem to be immediate. Dynamic groups are more complex, bringing in new elements, security requirement and issues.

A dedicated and detailed treatment is required to resolve the numerous existing issues and ambiguities. There are some important features of the model and definitions discussed in the paper Bellare. et. al.[8] which are described below.

Two authorities. Bellare. et. al.[8] separate the authority into two, an opener (who can open signatures) and an issuer (who interacts with a user to issue the latter a signing key). Each has its own secret key. This provides more security (compared to having a single authority) in the face of the possibility that authorities can be dishonest.

Trust levels. They consider different levels of trust in each authority, namely that it may be uncorrupt (trusted), partially corrupt (its secret key is available to the adversary but it does not deviate from its prescribed program) or fully corrupt (the adversary controls it entirely, so that it may not follow its program). In order to protect group members against dishonest authorities to the maximum extent possible, they formulate security requirements to require the lowest possible level of trust in each authority.

Three key requirements. Bellare. et.al.[8] formulate three key requirements, namely anonymity, traceability and non-frameability.

PKI. They assume that each group member or potential group member has a personal public key, established and certified, for example by a PKI, independently of any group authority, so that it has a means to sign information, using a matching personal private key that it retains. This is necessary in order for group members to protect themselves from being framed by a partially or fully corrupt issuer.

Publicly verifiable proofs of opening. In order to be protected against a fully corrupt opener, the opener is required to accompany any claim that a particular identity produced a particular signature with a publicly verifiable proof to this effect.

Concurrent join protocols. In an Internet-based system, Bellare. et. al.[8] would

expect that many entities may concurrently engage in the join protocol with the issuer. Bellare. et. al.[8] model captures this by allowing the adversary to schedule all message delivery in any number of concurrent join sessions.

2.1.1 Disadvantages:

This model captures the functionality of current efficient proposals for group signature schemes, in particular that of [1]. Although They do not know whether their scheme can be proven secure in their model, providing the model at least, enables one to address this question rigorously in the future. As is not uncommon with foundational schemes, their scheme is polynomial-time but not efficient.

2.2 Group Blind Digital Signatures: A Scalable Solution to Electronic Cash. By Anna Lysyanskaya and Zulficar Ramzan [9]

In this paper Lysyanskaya and Ramzan[9] construct a practical group blind signature scheme. Their scheme combines the already existing notion of blind signature schemes and group signature schemes. It is an extension of Camensich and Stadler's Group Signature Schemes [9] that add the blindness property. Lysyanskaya and Ramzan[9] show how to use blind group signature schemes to construct a electronic cash system in which multiple banks can securely distribute anonymous and untraceable e-cash. Moreover the identity of e-cash issuing bank is concealed which is conceptually novel. The space, time and computational complexities of the relevant parameters and operations are independent of the group size.

Lysyanskaya and Ramzan [9] show how to use group blind digital signature scheme in which multiple banks can securely distribute anonymous and untraceable e-cash.

Their scheme holds the following properties

- **Blindness:** The signer is unable to view the message he signs
- **Unforgeability:** Only group member can issue the valid signature
- **Undeniable signer identity:** The group manager always establishes the identity of the member who issued the valid signature.

- **Signer anonymity:** Only the group manager can determine which member issued the signature
- **Unlinkability:** Two message-signature pair where the signature issued by the same member can not be linked.
- **Security against framing attacks:** Neither the group manager nor the group member can sign on the behalf of the other member.

2.2.1 Disadvantages:

- Their scheme is not fully dynamic in nature, In their scheme the member once join the group remain there throughout the life of group.i.e No revocation
- Scheme is Online (each coin must checked by bank before vendor gives merchandise to user)
- Can be made offline if they compromise degree of user anonymity
- Computing a Signature requires:
 - 3 round protocols between user and bank with roughly 7-8 Kilobytes of data transferred during each round. (to achieve blinding)
- Approximately 1000 modular exponentiations
- They don't address important issues:
 - Maintaining universal list of coins, how central bank audits constituent banks, etc.
 - How keys are traversed over the insecure channels.
- They have the single authority for issuing membership certificate to group members as well as to open the signature in case opening/
- A trusted third party entity chooses not only the group public key and an opening key for the opening authority, but for each group member, chooses a signing key and hands it to the member in question.
- They also require an uncomfortably high degree of trust in the party performing setup, since the latter knows the signing keys of all members and can thus frame any group member.

2.3 Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions by Mihir Bellare, Daniele Micciancio, Bogdan Warinschi[19]

In this paper Bellare et. al. introduces strong, formal definitions for the core requirements of anonymity and traceability. They assume that trapdoor permutations exist to meet these formal definitions. The example of foundational work for group signature scheme primitives is public-key encryption.

2.3.1 Definition for Security Requirement

Bellare .et.al considers novel attack capabilities and success measures, and then formulate strong versions of the core requirements that we call full-anonymity and full-traceability.

Full-anonymity.

They define a strong adversary that may corrupt all the members of the group, including the one issuing the signature and capture the possibility that the adversary can see the outcome of opening attempts conducted by the group manager on arbitrary signatures of its choice.

Full-traceability

It asks that a group of colluding group members who pool their secret keys cannot create a valid signature that the opening algorithm would not catch as belonging to some member of the colluding group.

They show however that all existing requirements are implied by full-anonymity plus full- traceability that having to check only two security properties makes it easier to give formal proofs of security when new group signature schemes are invented.

2.3.2 Their Scheme:

In this scheme, they stress that the result of their scheme is not in the random oracle model and this construction is non-trivial" in the sense that the sizes of all keys depend only logarithmically (rather than polynomially) on the number of group members.

The construction uses as building blocks an IND-CCA secure asymmetric encryption scheme, simulation-sound adaptive non-interactive zero-knowledge (NIZK) proofs for NP and a digital signature scheme secure against chosen-message attack.

The formulations of anonymity and traceability are strong enough to capture all existing informal security requirements necessary for group signature scheme.

2.3.3 Construction of Their Scheme:

Their construction uses a digital signature scheme that satisfies the standard notion of unforgeability under chosen message attack. It is known that digital signature schemes, unforgeable under chosen message attack, exist assuming one-way functions exist, and hence certainly assuming the existence of a family of trapdoor permutations.

Encryption schemes:

Their group signature construction utilizes an encryption scheme that satisfies the standard notion of indistinguishability under chosen-cipher text attack.

2.3.4 Security Results

Fix digital signature scheme $DS = (Ks; Sig; Vf)$, public-key encryption scheme $AE = (Ke; Enc; Dec)$, NP-relation p over domain Dom , and its non-interactive proof system $(P; V)$ as above, and let $GS = (GKg; GSig; GVf; Open)$ denote the signature scheme associated to them as per our construction.

Lemma 1 *If AE is an IND-CCA secure encryption scheme and $(P; V)$ is a simulation sound, computational zero-knowledge proof system for $_$ over Dom then group signature scheme GS is fully-anonymous.*

Lemma 2 *If digital signature scheme DS is secure against forgery under chosen-message attack and $(P; V)$ is a sound non-interactive proof system for $_$ over Dom then group signature scheme GS is fully-traceable.*

2.3.5 Disadvantages of scheme:

The definitions and results of this paper are for the setting in which the group is static, meaning the number and identities of members is decided at the time the group is set up and new members cannot be added later.

2.3.6 Future work

Their work provides a basis from which to treat the case of dynamic groups, but the latter do give rise to new issues.. Providing formal definitions of security and provably-secure constructions for dynamic group signatures is the subject of on-going work.

Dynamic Groups and Other Extensions:

In such scheme, the group is dynamic. i.e., members can join and leave the group over time.

Their construction can be easily adapted to satisfy either definition of security for fully dynamic groups, e.g., by rekeying the entire group at the end of each time period. Due to the high (but unavoidable) cost of rekeying operations, fully dynamic group signatures should be used only if required by the application, and in all other situations the simpler incremental groups are clearly preferable.

2.4 Group Signature Scheme Based on an RSA-Variant by Jan Camensich and Markus Michels [14]

In this paper Camensich and Michels[14] propose a new group signature scheme that is well suited for large groups, i.e., the length of the group's public key and of signatures do not depend on the size of the group. Their scheme is based on a variation of the RSA problem called strong RSA assumption. Signatures can be verified with respect to a single public key of the group and do not reveal the identity of the signer. The membership manager is responsible for the system setup and for adding group members while the revocation manager has the ability to revoke the anonymity of signatures.

2.4.1 Previous work:

Various group signature schemes have been proposed so far. However, in the some schemes the length of signatures and/or the size of the group's public key depend on the size of the group and thus these schemes are not suitable for large groups. Only few schemes are the length of signatures and the size of the group's public key independent of the number of group members, while there is also a scheme that satisfies the length requirement as well, but these are inefficient.

2.4.2 An Approach for a Realization:

In this model, a group signature scheme normally consists of the following algorithms like setup, sign, verify, tracing, Vertracing. All algorithms are same like used in previous group signature scheme except Tracing and Vertracing

Tracing: A tracing algorithm that on input a signature, a message m , the revocation manager's secret key xR , and the group's public key Y returns the identity ID of the group member who issued the signature together with an argument arg of this fact.

Vertracing: A tracing-verification algorithm that on input a signature, a message m , the group's public key Y , the identity ID of a group member, and an argument arg outputs 1 if and only if arg was generated by tracing with respect to m , Y , xR .

The efficiency of a group signature scheme can be measured by the size of the public key Y , the length of signatures, and by the efficiency of the algorithms sign, verify, setup, tracing, and vertracing.

2.4.3 Working of Proposed scheme:

The basic idea of the scheme is the following.

The membership manager chooses a group $G = \langle g \rangle$ and a group element z . Furthermore, he chooses a second generator h such that \log of g h is unknown. Computing discrete logs in G to the bases g , h , or z must be infeasible. Finally, computing roots in G must be feasible only to the membership manager, i.e., he should be the only one who knows the order of G . The revocation manager chooses his secret key x and publishes $y = g^x$. Each group member chooses a prime e randomly in a determined range together with the membership manager. Only the group member learns e and stores it as a secret key. A membership certificate issued by the membership manager is an element $u \in G$ such that $u^e = z$ holds. Here they slightly deviate from the approach of Camenisch and Stadler, i.e., the membership certificate and the membership key are the same number. As a consequence, the issuing of certificates must be realized in a way that the membership manager is not able to learn the group member's secret key e .

A signature of a message m by a group member consists of a triple $(a, b, d) \in G^3$ and an SPK of integers u and e such that the pair (a, b) is an encryption of u under the revocation manager's public key (which is part of the group public key)

- d commits to e ,
- e lies in the necessary range, and
- $u^e = z$ holds.

The membership manager can reveal the identity of a signer by asking the revocation manager to decrypt (a, b) . The following paragraphs describe the new scheme in detail and provide security and efficiency analyses.

2.4.4 Efficiency Analysis

With $\epsilon = 9/8$, $\ell_n = \hat{\ell} = 1200$, $\ell_1 = 860$, $\ell_2 = 600$, and $k = 160$, signature generation and verification need little less than 130000 modular multiplications modulo a 1200-bit modulus in average, and the signature is about 1 Kbytes long. Compared to the most efficient scheme given in [9], this scheme is about three times more efficient and signatures are about three times shorter when choosing the same modulus for both schemes. However, the registration protocol is less efficient in their scheme. Signatures could be made shorter without compromising the security of the scheme if the parameter w in the signing procedure is chosen from a smaller domain, e.g., $\{0, 1\}^{\ell_2}$ instead of $\{0, 1\}^{\ell_2}$.

2.4.5 Conclusion

Splitting the membership and/or the revocation manager can be done. As the signature generation algorithm was derived from an interactive protocol, a group identification scheme (also called identity escrow) is obtained by using this protocol for identification.

2.5 Efficient Group Signature Schemes: For large groups. By Jan Camensich and Markus Stadler [15]

In this paper Camensich and Markus Stadler [15] propose the first group signature scheme whose public key and signature have length independent of the number of group members, and which can therefore be used for large number of groups. Previously proposed realization schemes have an undesirable property that the length of the public

key is linear in the size of group. This scheme has overcome the problems of the length of public key and of the signature as are in previous schemes, as well as the computational effort for signing and verifying, independent of the number of group members. Furthermore the public key remains unchanged if new members are added to the group. The scheme even conceals the size of group.

2.5.1 Construction of Scheme

Use of Signature of Knowledge of Discrete Logarithms

Jan Camensich and Markus Stadler [15] make use “proof” systems that allow one party to convince other parties about its knowledge of certain values, such that no useful information is leaked. They use construction based on the Schnorr signature scheme [16] to prove knowledge. However to avoid confusions with the notion of proofs of knowledge of and to point out these proof also serve as signatures, they call them signature of knowledge. The first primitive they define is a signature of the knowledge of the discrete logarithm of y to the base g . It is basically a schnorr scheme [16] on a message m of the entity knowing the discrete logarithm of y .

A More Efficient Variant

An evident solution to design a more efficient group signature scheme is to modify it in a way that allows to replace the SKLOGLOG signature, this is indeed possible if the membership key is computed using $y = xe \pmod{n}$ instead of $y = hx \pmod{n}$.

System Setup

The group manager compute the following values:

an RSA modulus n and two public exponent $e_1, e_2 > 1$ is relatively prime to $\varphi(n)$; two integer $f_1, f_2 > 1$ whose e_1 th roots and e_2 th roots cant be computed without knowing the factorization of n , a cyclic group $G = \langle g \rangle$ of order n in which computing discrete logarithms is infeasible, an element $h \in G$ whose discrete logarithm to the base g must not be known, his public key $Y = h^p$ for a randomly chosen value $p \in \mathbb{Z}_n$.

Membership Keys and blind Issuing of Membership Certificates

To become a group member, Alice computes her membership key as follows

$$y := x^{e_1} \pmod{n} \text{ for } x \in_R \mathbb{Z}_n^*$$

$$z := g^y$$

A certificate in this scheme is of the form

$$v = (f_1 y + f_2)^{1/e_2} \pmod{n}$$

Signing Messages:

To sign a message on behalf of the group, Alice performs the following computation:

$$- \tilde{z} := h^r g^y \text{ for } r \in_R \mathbb{Z}_n^*$$

$$- d := y_R^r$$

$$- V_1 := E\text{-SKROOTREP}[(\alpha, \beta) : \tilde{z} = h^\alpha g^{\beta e_1}] (m)$$

$$- V_2 := E\text{-SKROOTREP}[(\gamma, \delta) : \tilde{z}^{f_1} g^{f_2} = h^\gamma g^{\delta e_2}] (m)$$

$$- V_3 := SKREP[(\epsilon, \zeta) : d = y_R^\epsilon \wedge \tilde{z} = h^\epsilon g^\zeta] (m)$$

Opening Signatures:

When the group manager wants to open a signature $(\tilde{z}, d, V_1, V_2, V_3)$ on the message m , he computes $\hat{z} := \tilde{z}/d^{1/\rho}$ which corresponds to the signer's membership key z . To prove that z is indeed encrypted in \tilde{z} and d , the group manager computes

$$SKREP[\alpha : \tilde{z} = z d^\alpha \wedge h = y_R^\alpha] (''),$$

Which he can do because $1/\alpha \pmod{n}$ corresponds to his administration key.

Future Work: An obvious extension would be to assign the different roles of the group manager to different entities, i.e, to a membership manager who is responsible for adding new members to the group and to a revocation manager, who is responsible for opening signature. The functionality of these managers can also be shared among several entities.

TH - 4426

2.6 Problem Domain and Proposed Solution

2.6.1 Problem Domain:

The bulk of existing practical schemes and applications are for dynamic groups. These groups involve important new elements e.g introducing new concept of separating group authorities, different trust levels and new security issues etc.

We consider the Lysyanskaya and Ramzan [9] Group Signature Schemes that has the blindness property. Their scheme does not addresses the issues of transferability and divisibility, a trusted entity which acts as a passive trustee and chooses not only the group public key and an opening key for the opening authority, but also, for each group member, chooses a signing key and hands it to the member in question. There is no division of group managers into two entities i.e. opener (which can open any signature) and issuer (which issues the signature to members) as this involves more security and trust on authorities.

They also require an uncomfortably high degree of trust in the party performing setup, because it knows the signing keys of all members and can thus frame any group member. As there is only one authority, there exists no trust level that tells the possibilities of security compromises. Their join protocol is not designed to add more members concurrently. They don't provide the traceability property.

2.6.2 Proposed Solution:

We will consider the Lysyanskaya and Ramzan 97] scheme which is implemented for multiple distributed banking and doesn't fulfill the full requirements of dynamic group signature scheme and make it dynamic in nature to add the advantages discovered in Bellare. et. al.[8] .

To overcome the limitations of static group signature scheme, Bellare.et. al.[8] presented a foundation for dynamic group signature scheme. In their setting neither the number nor the identities of group members are fixed or known in the setup phase. An entity can join the group, and obtain a private signing key at a time, by engaging in an appropriate join protocol with the authority.

Bellare. et.al[8]scheme added more security requirements including unlinkability, unforgeability, collusion resistance, excludability and framing resistance.

Bellare. et. al.[8]separate the authority into two, an opener (who can open signatures) and an issuer (who interacts with a user to issue the latter a signing key). Each has its own secret key. This provides more security (compared to having a single authority) in the face of the possibility that authorities can be dishonest).

Concurrent join protocols. In an Internet-based system, many entities may concurrently engage in the join protocol with the issuer. Bellare. et. al.[8] model captures this by allowing the adversary to schedule all message delivery in any number of concurrent join sessions.

In Group Blind Signature Scheme of Lysyanskaya and Ramzan[9] function of group manager is not split in between two managers i.e. an issue and an opener. This is the desirable property that allows distribution of trust. The chosen scheme of Bellare. et. al.[8] has different advantages. First the vendors only has to know a single group public key to check the validity of any e-cash he receive. Second the spender's identity is completely anonymous to both the vendor and his bank since the signature is blind. Third neither the vendor nor the vendor's bank can determine the user bank. Fourth the bank which receives the e-cash just needs to check it with group blind public key .So to make it stronger we implement the new definitions provided by of Bellare. et. al.[8] in it.

For their definitions they give different algorithms and for security issues they made different oracles. The correctness and security definitions will be formulated via experiments in which an adversary's attack capabilities are modeled by providing it access to certain oracles.

In this dynamic group signature scheme different entities will involved namely, a trusted party for initial key generation, an authority called the issuer, an authority called the opener, and a body of users, each with a unique identity. The scheme is specified as a tuple

GS - (GK_g; UK_g; Join; Iss; GSig; GVf; Open; Judge) of polynomial-time algorithms whose intended usage and functionality are as follows.-

GKg - In a setup phase, the trusted party runs the group-key generation algorithm GKg on input 1^k to obtain a triple $(gpk; ik; ok)$. The issuer key ik is provided to the issuer, and the opening key ok is provided to the opener. The group public key gpk , whose possession enables signature verification, is made public.

UKg- A user that wants to be a group member should begin by running the user-key generation algorithm UKg on input $1k$ to obtain a personal public and private key pair $(upk[i]; usk[i])$. It is assumed that the table upk is public. (Meaning, anyone can obtain an authentic copy of the personal public key of any user. This might be implemented via a PKI.)

Join; Iss- Once a user has its personal key pair, it can join the group by engaging in a group-joining protocol with the issuer. The interactive algorithms Join; Iss implement, respectively, the user's and issuer's sides of this interaction. Each takes input an incoming message (this is $"$ if the party is initiating the interaction) and a current state, and returns an outgoing message, an updated state, and a decision which is one of $accept; reject; cont$. The communication is assumed to take place over secure (i.e. private and authenticated) channels, and we assume the user sends the First message. If the issuer accepts, it makes an entry for i , denoted $reg[i]$, in its registration table reg , the contents of this entry being the Final state output by Iss. If i $accept$, the Final state output by Join is its private signing key, denoted $gsk[i]$.

GSig- A group member i , in possession of its signing key $gsk[i]$, can apply the group signing algorithm GSig to $gsk[i]$ and a message m , to obtain a quantity called a signature on m .

GVf- Anyone in possession of the group public key gpk can run the deterministic group signature verification algorithm GVf on inputs gpk , a message m , and a candidate signature α for m , to obtain a bit. We say that α is a valid signature of m with respect to gpk if this bit is one.

Open- The opener, who has read-access to the registration table reg being populated by the issuer, can apply the deterministic opening algorithm $Open$ to its opening key ok , the registration table reg , a message m , and a valid signature σ of m under gpk . The algorithm returns a pair $(i; \delta)$, where $i \geq 0$ is an integer. In case $i \geq 1$, the algorithm is claiming that the group member with identity i produced σ , and in case $i = 0$, it is claiming that no group member produced σ . In the former case, δ is a proof of this claim that can be verified via the Judge algorithm.

Judge- The deterministic judge algorithm $Judge$ takes inputs the group public key gpk , an integer $j \geq 1$, the public key $upk[j]$ of the entity with identity j (this is \perp if this entity has no public key), a message m , a valid signature σ of m , and a proof-string δ . It aims to check that δ is a proof that j produced σ . We note that the judge will base its verification on the public key of j .

Chapter 3



Research Methodologies

3. Research Methodologies

3.1 Group Signature Scheme

Group signature schemes (thereafter denoted GSS) have been introduced by Chaum and van Heyst [17], in order to provide revocable anonymity to the signer, who is allowed to sign on behalf of a group. In such a scheme, an authority is able, in exceptional cases, to open" any group signature and thus recover the actual signer. Properties of group signature schemes make them very important cryptographic tools, with lots of applications (voting, bidding, anonymous attestation).

3.2 The Group Blind Digital Signature Scheme

It's a group signature schemes and added the blindness property to make it Group Blind Signature scheme. The applications of group blind digital signatures are online distributed banking, offline distributed banking and online voting.

3.2.1 Security Requirements for Group Blind Digital Signature

The security requirements of a Group Blind Digital signature are very similar to those of a Group Digital Signature. The only addition is that we require the blindness property in the signature. Here are the security requirements:

Blindness of Signatures: The signer is unable to view the messages he signs. Moreover, the signer should have no recollection of having signed a particular document even though he (or anyone else for that matter) can verify that the signature is indeed valid.

Unforgeability: Only group members can issue valid signatures on behalf of the entire group; i.e. only group members can issue signatures that are verifiable by the group public key.

Conditional Signer Anonymity: Given a message/signature pair anyone can easily check that the signature is valid, and that the message was signed by some particular group member, but only the group manager can determine which specific member issued the signature. One can consider an alternate model in which the signer's anonymity is

retained, even with respect to the group manager, but this is not the model we consider in this thesis.

Undeniable Signer Identity: The group manager can always determine the identity of the group member who issued a valid signature. Moreover, he can also prove to some other entity (such as a judge) which member signed a given document without compromising that particular group member's anonymity in previous or future messages he may sign.

Unlinkability: Determining if two different signatures were computed by the same group member is computationally infeasible.

Security Against Framing Attacks: No subset of group members (perhaps including the group manager) can sign a message on behalf of another group member. That is, if the `Open` procedure is invoked on the message, it should not specify the name of another group member not belonging to the original subset.

Coalition Resistance: Any subset of group members (not including the group manager) should be incapable of generating valid group signatures.

Untraceable. In particular, we want to prevent attacks in which a coalition of group members get together, pool their information, and generate signatures which are approved by the `Verify` procedure, but for which the `Open` procedure fails to reveal any group member.

3.2.2 Procedures Allowed by Group Blind Digital Signatures

The procedures given in our Group Blind Digital Signature model are identical to those given in the original group signature model. We list them here again for completeness.

Setup a probabilistic algorithm that generates the group's public key Y and a secret administration key S for the group manager.

Join an interactive protocol between the group manager and the new group member Bob that produces Bob's secret key x , and his membership certificate A .

Sign an interactive protocol between group member Bob and an external user Alice, which on input a message m from Alice and Bob's secret key x produces a signature s on m that satisfies the properties above.

Verify an algorithm which on input $(m; s; Y)$, determines if s is a valid signature for the message m with respect to the group public key Y .

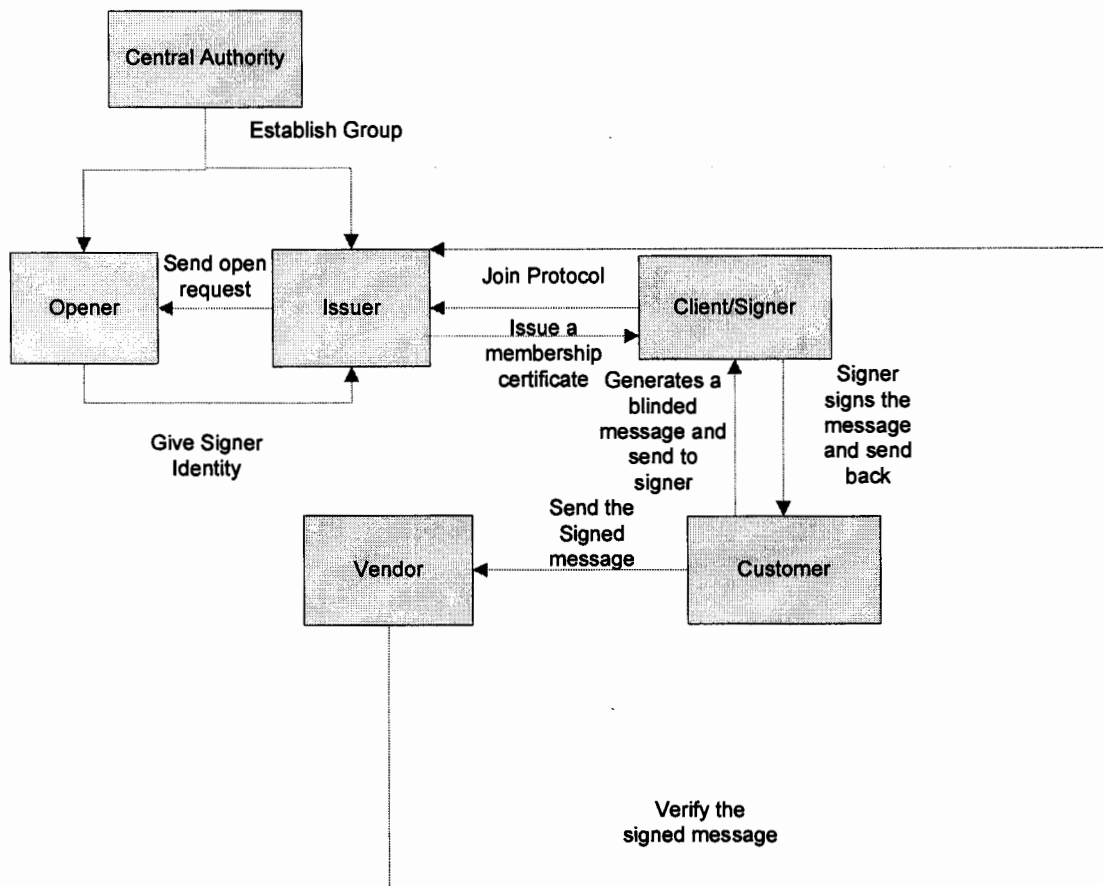
Open an algorithm which, on input $(s; S)$ determines the identity of the group member who issued the signature s .

3.3 Our Scheme

3.3.1 Entities Involved in Our Scheme

- **Central Authority** This authority will establish the group.
- **Issuer** It will appoint the new members of the group.
- **Opener** It will open the signature to revoke the anonymity of signer.
- **Users** who want to join the group.
- **Signers** members of the group who are allowed to create sign on behalf of group.
- **Client** They are the clients who give request to signer to create sign
- **Vendor** They don't belong to the group but just have the access to the group public key.

Dynamic Group Blind Digital Signature Scheme



3.3.2 Assumptions of Our Scheme

We establish different assumptions for our scheme to make it more practical which are described below.

- There will be some authority called central authority called CA, who will establish the group by running the Setup algorithm. This will be treated as trusted entity for the group and may be called as owner of the group.
- CA will appoint the group authorities, i.e. Issuer and Opener.
- CA will create group keys.

- CA will establish all the rules for group i.e. allowing or rejection of members.
- Issuer will add new members to the group, by issuing new certificates.
- Issuer will handle the group database i.e. read/write access.
- Opener has read only access to group database.
- We assume that each user $U[i]$, before joining the group, obtains a personal secret key $PS[i]$, associated to a personal certified public key $PP[i]$ (in a PKI) which he will use for various purposes i.e for his own authentication.

- The group manager will also have a certified pair of keys ($gmpk$; $gmsk$). This PKI environment is separated from the group environment, and thus the certification authority will be assumed fully trusted (the only one).

- This PKI will provide the non-repudiation, but also the non-frame ability property: even if the group authorities are corrupted, they cannot frame a group member. Such a PKI can be formalized by an any user-key generation algorithm which generates a personal public and private key pair ($PP[i]$; $PS[i]$) for a user U_i .

- We assume that all public keys are published and are accessible to any one over a secure way.

- For all the published public keys associated to group and group members, issuer has the right to delete them in special cases.

3.3.3 Notations

For our algorithms we use the following notations.

- $DgbSetup()$ for Set up algorithm
- $DgbJoin()$ for Join –Issue algorithm

- DgbSign() for sign algorithm
- DgbVerify() for verify algorithm
- DgbOpen () for open algorithm.

For the group keys we use the following notations.

- DgbSkis the group signature key.
- DgbPk.....is the group public key used for group signature verification.
- DgbEnc..... .is the group Encryption key.
- DgbCrtVf.....is the group Certificate Verification key used to verify the membership certificates.

For the individual keys we use the following notations.

- Ik..... is the issuer key
- ok..... is the opener key
- GS[i].....is the group secret key of the member.
- GP[i].....is the group public key of the member.
- PS[i].....is the personal secret key of the user.
- PP[i].....is the personal public key of the user.

3.3.4 Key Usage

Here we will describe the usage of above mentioned keys in our scheme. The keys are generated in different algorithms and are used for various purposes. Following is the description of their usage

➤ DgbSk

This key is the group signing key used to create the group signature on some attributes after creating the sign by some signer

➤ DgbPk

This key is the public key of the group used to verify the signature of the group. This key is available to anyone. Those who want to verify that coming signature is from this corresponding group will use it.

➤ DgbEnc

This key is the group encryption key used to encrypt the sign. This key is used only by the group members. Signer after creating the sign encrypt the sign and his identity using this key and produce the Cipher Text used by opener. This key is used to provide the anonymity of the signer.

➤ DgbCrtVf

This key is the certificate verification key used by that user who wants to join the group. After their validation issuer send membership certificate to them. They use this key to check the validation of legal issuer. This key will help to check the correctness of Honest Issuer.

➤ Ik

This key is used by the issuer to create membership certificate to members.

➤ Ok

This key is used by opener for opening the signature in order to revoke the anonymity of the signer.

➤ GS[i]

This key is the Group Secret key of the signer which he produces for group signatures.

➤ GP[i]

This key is the Group Public key of the signer used to verify the signatures.

3.3.5 Algorithms of Our Scheme.

For our scheme we use RSA Key generation algorithm for encryption/decryption and for signature creation /verification.

1. SetUp Algorithm.

This algorithm is run by CA i.e. central Authority. It has following steps.

- Choose key size of 1024 bits for all the keys.
- Create group encryption key DgbEnc and Decryption key DgbDec by running RSA encryption key generation algo i.e. Ke();
- Create a certificate creation DgbCrtCr and certification verification key DgbCrtVf by running RSA signature key generation algo Ks();
- Create group signature creation key DgbSk and group public key DgbPk again by running RSA signature keys generation algorithm. i.e Ks();

- Assign DgbCrtCr to issuer as his ik.

- Assign DgbCrtVf to opener as his ok.

In this phase CA will publish the DgbPk for worldwide access and DgbCrtVf and DgbEnc restrict to those who are eligible to join the group. Appoint the Issuer and opener of the group and assign give ik and ok to them respectively.

2 .Join Algorithm.

- Client (i) who wants to join the group will send a Join-request by providing his following information to Issuer.
 - User-name.
 - User-contact-no.

- User-request='Want to Join'.
- Issuer check the validity of user. If valid allow him to start group join protocol.
 - a- User will generate two other keys by running key generation Algo().
 - Verification key called GP[i]
 - Signing key called GS[i]
 - b User will create sign i.e sig[i] on his GP[i]by using his PS[i].
 - c- User sends sig[i] and pk[i] to issuer.
 - d- Issuer verify the sig[i] using the PP[i].
 - e- If verify, issuer issues the membership certificate to i
 - In membership certificate cert (i) issuer signs pk[i] of i using his certificate creation key ie. Ik.
 - f- Issuer will make an entry reg[i]in registration table reg by storing pk[i]and ig[i]
 - g- Issuer sends cert (i) to user i
 - h-User will verify this cert(i) by using certificate verification key i.e DgbCrVf.
- Issuer will end the group-joining protocol.

3 .Sign Algorithm

Let m be the message on which sign has to produce. Signer will create a sign using his GS[i]on a message m .

- Blindness will be created by Cham's RSA-based blind signature scheme, the signer's public key is $n; e$, and its secret key is $n; d$ where these quantities are as in the RSA system.
- A traditional RSA signature is computed by exponentiating the message m with the secret exponent d , all mod a public modulus n .
- For producing the hash function MD5 algorithm is used.
- The blind version adds a random value r , such that $gcd(r, N) = 1$.

- r is exponentiated with the public exponent $e \pmod{N}$, and the value r^e is used as a blinding factor.
- signer receives the product of the message and blinding factor $m(r^e) \pmod{N}$, which obscures the message.
- The blinded signature s' is then calculated as:

$$s' \equiv (m(r^e))^d \pmod{N}$$

- The author of the message can then remove the blinding factor to reveal s , the valid RSA signature of m :

$$s \equiv s' * r^{-1} \pmod{N}$$

- After creating the signature Signer encrypts his identity, public key attributes n and e by the Group Encryption Key i.e. DgbEnc and makes a cipher text C .
- After creating the Cipher Text signer again create simple RSA traditional signature called group signature on this Cipher Text and the name of Group with Group Signature Key i.e. DgbSk.
- Signer will send signature created on blinded message, Cipher Text and group signature back to person in question.

4. Verify Algorithm

Group signature will be verified by the RSA verification method using MD5 hash function.

- In this method Verifier gets the public key of the signer and the message.
- He computes his own hash value (call this **hash***) for the received message then
- decrypts the signed message with signers public key decrypts then compares **hash*** to the decrypted value of **sig** to see if they are the same.

Verification algo will return a bit

- if bit is 1 its valid signature of m
- Otherwise not valid

5. Open Algorithm

This algorithm return the name of the signer who sign the corresponding message
It takes signature , related Cipher Text with this signature and message as its
input parameters. And do the following steps

- Parse its opening key .i.e. ok
- Decrypt the Cipher Text with this key and get the identity of the signer
- Check the group database for the identity
- If found the entry then return the name of the signer.

Chapter 4



Implementation

4. Implementation

Implementation includes all the details that were required to make the system operational. The development tools and technologies to implement the system and also reasons for selecting particular tool is discussed. Then the modules being translated into the implementation tool will be described.

Software selection is very important step in developing a computer based system. The software that is used is capable of meeting the requirement of the proposed system. After considering the number of tools available these days such as Visual Basic, Visual C++, Visual C, MS Excel, Visual Basic.Net and MATLAB we choose Java Development Kit (jdk) version 1.6.0 and Microsoft Access for database management .

4.1 Java

The Java platform consists of the Java application programming interfaces (APIs) and the Java APIs are libraries of compiled code that you can use in your programs. They let you add ready-made and customizable functionality to save you programming time. Java programs are run (or interpreted) by another program called the Java VM. If you are familiar with Visual Basic or another interpreted language, this concept is probably familiar to you. Rather than running directly on the native operating system, the program is interpreted by the Java VM for the native operating system. This means that any computer system with the Java VM installed can run Java programs regardless of the computer system on which the applications were originally developed. For example, a Java program developed on a Personal Computer (PC) with the Windows NT operating system should run equally well without modification on a Sun Ultra workstation with the Solaris operating system, and vice versa.

4.2 Microsoft Access

Microsoft Access is a powerful program to create and manage your databases. It has many built in features to assist you in constructing and viewing your information. Access is much more involved and is a more genuine database application than other programs such as Microsoft Works.

It's a database program that employs the Relational Database Model. You can use databases to keep track of things like appointments or to organize other data.

Access provides a more intuitive way to enter, sort, and delete data. In a spreadsheet, it can be difficult to see which column or row a piece of information is in as you get farther from the row and column labels because there are only lines dividing the pieces of information. Using a form in Access can allow you to enter data all on one screen.

4.3 Implementation of Dynamic Group Blind Signature Model

We now give applications of Dynamic Group Blind Digital Signatures to electronic cash.

Electronic Banking

Consider a scheme in which there is a large group of members of a bank, monitored by two administrative authorities. Issuing authority, who issues the membership certificates to new member who joins the group and an opening authority who can open the identity of the signer in case of any dispute. In such system each group members can dispense electronic cash. We would like such a scheme to have the following properties:

- No signer should be able to trace any e-cash it issues. If the issues an electronic coin to a customer, and if it later happens to see that coin again (after it was spent), it should not be capable of determining which particular customer spent that coin. In addition, if the customer happens to spend several electronic coins, and the sees those coins, it should not be able to determine that those were spent by the same customer. Therefore, just as with paper money, people can spend their e-cash in a completely anonymous fashion.
- A vendor only needs to invoke a single universal verification procedure, based on the group public key, to ensure the validity of the group's signature on any

e-cash he receives. This procedure works regardless of which signer issued the e-cash. This makes the vendor's task much easier since he only needs to know the single group public key. We note that even if the signature on the coin is valid, it still might be possible that the coin is not valid for spending; e.g. if the coin has already been spent, then the signature is valid, even though the coin should not be spent.

- There is a single public key for the entire group members. The size of this public key is independent of the number of members. Moreover, the public key should not be modified if more members join the group. Thus, the scheme is still practical even if there are a large number of participating members.
- Given a valid piece of e-cash, Only the opener can tell which member in the group issued it. No vendor can even determine the signer from which the customer got her coin signed even though the vendor can easily check that the e-cash is valid. This restriction gives an extra layer of anonymity since we conceal both the spender's identity and the identity of the signer she uses.
- No subset of group(perhaps even including the Issuer) can issue coin on behalf of another group member; i.e. no subset of group or any other entities, can "frame" another group member.
- Any coalition of group member should be unable to construct valid looking electronic coins that are untraceable. In other words, no coalition should be able to construct coins such that the opener is incapable of associating that coin to any particular group member in the group.
- In case of any dispute, the opening authority can open the identity of the signer in question.

4.4 Applying Dynamic Group Blind Digital Signatures to Distributed Electronic Banking

We describe the necessary protocols for achieving the distributed banking scheme we envisioned earlier. The parties involved in our protocol are: Alice, Bob, Bank. Here Alice is a customer who happens to use Bank. Alice wishes to purchase some particular item from a vendor Bob.

4.4.1 Setup Phase

The trusting central authority of the bank in our scheme forms a group, and there are two authorities, an issuer and the opener in the group. If new user wish to join the group later, they may do so rather easily.

In this phase keys are generated by Trusted third party who publishes a group public key for public access and Certificate Verification key and Encryption keys restrict to those who are eligible to join the group. Appoint the Issuer and opener of the group and assign issuer key and opener key to them respectively.

Group Establishment and Creation of Keys

```
public class setGroup{
    PublicKey vfkey,enc;
    PrivateKey ik,ok;

    public static void main(String args[]){
        String issuer,group,opener;
        String gpk;
        setGroup gm=new setGroup();

        if (args.length != 3) {
            System.out.println("Usage: setGroup Group Name issuer name opener name");
        }
        else {try{
            System.out.println("Group Creation phase start...");
            System.out.println("Generating keys...");
            System.out.println("Generating certificate creation key for issuer...");
            System.out.println ("You choose MR/Ms..." +args[1] +" as ur Issuing Manager");
            gm.keyGen1(args[1]);
            opener=args[2];
            System.out.println("Generating Decryption key for opener...");
            System.out.println("You choose MR/Ms..." +args[2] +" as ur Opening
```



```

Manager");
    gm.keyGen2(opener);
    group=args[0];
    System.out.println("Generating Group Public Key (gpk)...");
    System.out.println("The name of your group is " +group);
    System.out.println(".....Some info abt Group.....");
    gm.keyGen3(group);
} catch(Exception e){}}

```

Join Protocol

Each Client performs the join protocol with the issuer, and is then capable of signing documents on behalf of the entire group of members.

In this Protocol, a client who wants to join the group requests to join the group by providing personal information. After authentication, Issuer allows him to start the join protocol. In Join protocol user generates his own personal keys and his signature, then sends them back to issuer. Issuer checks the validity of signature, if valid sends a membership certificate to the client who requested to join the group. Then join protocol is ended by the issuer

Join Request

```

public class issuerJoin {
    String alias;
    PublicKey pub;
    PrivateKey pri;
    byte[] plainText;
    public issuerJoin(){
        initComponents();
    }
    public static void main(String ar[]){
        issuerJoin ia=new issuerJoin();
    }
}

```

```

        ia.runServer();
    }
    public void btn_validityActionPerformed(ActionEvent e) {
        a=txt_clientId.getText();
        b=txt_clientname.getText();
        d=txt_clientNIC.getText();
        accountDatabase ab=new accountDatabase(a,b,d);
        String r=ab.response();
        System.out.println(r);
    }

    public void btn_protocolActionPerformed(ActionEvent e) {
        String s="Ur allowed to start the protocol...";
        String al="Allowed";
        sendData(s);
        alias=txt_clientname.getText();
        System.out.println(alias);
    }
    public void btn_certActionPerformed(ActionEvent e) {
        System.out.println("Creating..");
        createCert();
    }
    public void btn_endActionPerformed(ActionEvent e) {
        String end;
        end="SERVER>>> TERMINATE";
        sendData(end);    }
// Memebership Certificate
public void createCert(){

    String i=txt_clientId.getText();
    String cp=txt_sigpubkey.getText();

```

```

String conc=i+cp;
System.out.println("Concatinatinggggggggggg" +conc);
    try{
        FileInputStream in = new FileInputStream("issuer.dat");
        ObjectInputStream os = new ObjectInputStream(in);
        System.out.println("Got stream");
        String k=(String)os.readObject();
        pri=(PrivateKey)os.readObject();

        System.out.println("Got issuer keys");
        Signature sig = Signature.getInstance("MD5WithRSA");
        sig.initSign(pri);
        plainText = conc.getBytes("UTF8");
        System.out.println("Bytessss "+plainText);
        sig.update(plainText);
        byte[] signature = sig.sign();
        System.out.println(signature);
        System.out.println("end cert");
////////////////////////////////////
        FileInputStream in1 = new FileInputStream("certVfKey.dat");
        ObjectInputStream os1 = new ObjectInputStream(in1);
        System.out.println("Got varification stream");
        String k1=(String)os1.readObject();
        PublicKey pub=(PublicKey)os1.readObject();
        System.out.println("Get certificate varification key ");
        System.out.println(pub);
        sig.initVerify(pub);
        System.out.println("gotpub sig");

        System.out.println("iiiiiii"+plainText);
        sig.update(plainText);

```

```

    if (sig.verify(signature)) {
        System.out.println( "Signature verified" );
    }

```

4.4.2 Withdrawal

To withdraw money, the following steps are performed. Alice who is a customer and want to get his coin signed by the group member, creates an electronic coin C. This coin consists of some serial number, as well as some other information pertaining to currency, such as value, etc. Now Blind the coin by taking the modpower of r with e & n such that $(r^e \bmod n)$ and multiplying with C to get binded coin

Alice then asks bank to apply a Blind Digital Signature to C. One of the group member applies the signature s to C i.e create a digital signature using MD5 hash function & RSA sign function using his secret group key and attach it with the coin C Store the Coin Serial number and send it to user, and withdraws the appropriate amount of money from Alice's account. Alice now possesses a coin C along with a valid group signature on that coin C.

Coin Generation

```

public static void main(String args[]){
    Coin c=new Coin();
    c.runClient();
}
public Coin() {
    initComponents();
}
private void btn_genSerialActionPerformed(ActionEvent e) {
    rn=new Random();
    int i=rn.nextInt();
    txt_cSerial.setText(""+i); }
private void btn_getCertiActionPerformed(ActionEvent e) {
    try{
        String a,name;

```

```

a=txt_uName.getText();
name=a+"gkp.dat";
System.out.println(name);

private void btn_unBlindActionPerformed(ActionEvent e) {
//*****BLIND*****
SecureRandom random = SecureRandom.getInstance("SHA1PRNG","SUN");
    System.out.println("Random Number"+random);
    byte [] randomBytes = new byte[10];
    BigInteger gcd = null;
    BigInteger one = new BigInteger("1");
    do {
        random.nextBytes(randomBytes);
        r = new BigInteger(1, randomBytes);
        System.out.println(r);
        gcd = r.gcd(n);
    System.out.println("gcd: " + gcd); }
    while(!gcd.equals(one) || r.compareTo(n)>=0 || r.compareTo(one)<=0);
        BigInteger bm = ((r.modPow(e,n)).multiply(m)).mod(n);
        System.out.println(bm);
        bm1="" +bm;
        System.out.println("\nb = " + bm1);

    }catch(Exception w){
w.printStackTrace();
    }}

//***** UNBLIND*****

try{
    byte [] raw1 = bs.getBytes("UTF8");
    System.out.println(raw1);

```

```

BigInteger bs1 = new BigInteger(raw1);
ss = r.modInverse(n).multiply(bs1).mod(n);
System.out.println("UnBlind the signature to get original signature.. ");
System.out.println(""+ss);
System.out.println("Making the signed coin at this end");
String signercoin=msg+ss;
    } catch (Exception et) {
        et.printStackTrace();
    }
}

```

Signature Generation Process

```

public class signer {
    public static void main(String args[]){
        signer s=new signer();
        s.runServer();
    }
    private void btn_signCoinActionPerformed(ActionEvent e) {
        try{
            sig = Signature.getInstance("MD5WithRSA");
            sig.initSign(pri);
            byte[] plainText = bb.getBytes("UTF8");
            System.out.println("Bytessss "+plainText);
            sig.update(plainText);
            signature = sig.sign();
            System.out.println("Signature created for the coming coin "+signature);
        } catch (Exception le) {
            le.printStackTrace();
        }
    }
}

```

4.4.3 Deposit

To deposit a coin, the following steps must be performed.

1. Bob takes the coin C, along with group's signature on the coin, and gives it to Bank.
2. Bank looks at the coin, and verifies its validity by checking the signature on the coin. This can simply be done by using the group public key.
3. If all checks out, then Bank credits Bob's account with the appropriate amount.

In this protocol bank/Vendor/ Verifier gets the public key of the signer and the message. He computes his own hash value (call this **hash***) for the received message then decrypts the signed message with signers public key decrypts then compares **hash*** to the decrypted value of **signature** to see if they are the same.

Signature Verification Procedure

```

FileInputStream in1 = new FileInputStream("certVfKey.dat");
ObjectInputStream os1 = new ObjectInputStream(in1);
System.out.println("Got varification stream");
    String k1=(String)os1.readObject();
    PublicKey pub=(PublicKey)os1.readObject();
    System.out.println("Get certificate varification key ");
    System.out.println(pub);
    sig.initVerify(pub);
    System.out.println("gotpub sig");
    System.out.println("iiiiiii"+plainText);
    sig.update(plainText);
    if (sig.verify(signature)) {
        System.out.println( "Signature verified" );
    } else System.out.println( "Signature failed" );
    String aname=txt_clientname.getText();
    String sst="" +signature;

```

```

        sendData(""+plainText);

        for(int j=0;j<5000;j++);
        sendData(sst);
        }catch(Exception m){
            System.out.println(m);}}}}

```

4.4.4 Anonymity Revocation

If there is some reason to believe that a coin or a signer is fraudulent, then the identity of the spender can be determined as follows.

1. The coin C, the signer's signature on C, and the customer's signature on both are given to the opener.
2. Since the Opener is the group authority of all the members, he can use the Open algorithm to determine the identity of the original signer. It takes signature, related Cipher Text with this signature and message as its input parameters. Parse its opening key, decrypt the Cipher Text with this key and get the identity of the signer.

Opening Process

```

public class opener{
    PrivateKey ok;
    String id;
    byte[] cipherText ;
        public opener(byte[] C){
            getOpenKey();
            cipherText=C;
            String id=doDecrypt(cipherText);
            getregAccess(id);
        }
        public void getregAccess(String id){
            int i=0;

```



```
try {  
    String query="Select * from REG where signerID="+id;  
    s.execute(query);  
    ResultSet rs = s.getResultSet();  
  
    while ( rs.next() ){  
        i++;  
        String a=rs.getString(1);  
        String b=rs.getString(2);  
    }  
  
    if(i>=1){  
System.out.println("GOt signer.....yes this signature in this Cipher text is generated by  
this signer..");  
    }else  
        System.out.println("Dont Get Signer of this id..");  
        s.close();  
        con.close();  
    }catch (Exception err) {  
        System.out.println("ERROR: " + err);  
    }  
}
```

Chapter 5



Results

5. Results

In our scheme , we present a new DgbSS(Dynamic Group Blind Signature Scheme), which provides the security level (under by now classical RSA computational assumptions) in the random oracle model, with quite practical features: two authorities, PKI environment and , and Blind Signature. The blindness is achieved due to chaum's RSA blind signing method.

Our signature scheme, named DgbSS, provides anonymity in two levels, by encrypting signature with group encryption key which provides anonymity of signer and by blinding the message. Furthermore it is more secure and reliable than all those schemes of group signatures in which there is only a single authority both for issuing and opening.

Also Establishment of Public keys Infrastructure to achieve the anonymity and joining of group member, his authentication and group signature verification is there which make our scheme as for dynamic groups according to Foundation paper [8]

Our scheme has five algorithms of

- Setup
- Join
- Sign
- Verify
- Open

These algorithms establish a Dynamic Group Blind Digital Signature Scheme. In this scheme we introduce two separate authorities for issuing membership to group members and to open the signature. Our scheme is the first one in this combination of producing the blind signature with two separated authorities for join and Open algorithm and such a public key infrastructure which we develop to handle different issues.

Our Scheme holds the following properties. Here is detail description how we achieve these properties for our scheme.

1 Anonymity

This Scheme provides the anonymity of signer as well as the anonymity of the person who send request to sign the message.

- The anonymity of the signer is achieved when after creating his valid signature the signer encrypts the sign with the group encryption key. Where as its corresponding decryption key is only available to the opener. who is the trusted authority of the group. So the users who send the request to sign the message never know to whom his signature has been produced.

2 Blindness.

This Scheme provides the blindness property to create a valid signature on the blinded message.

- First Signature requesting authority blinds the methods i.e. hide the contents of the message. Signer creates his signature on this message. After this sender unblind the original message and get the real signs. Thus this property provides the anonymity of contents of the message.

3 Correctness

Only the signature generated by a honest member is accepted by the verification algorithm, and the open algorithm correctly identify the signer. Thus correctness in our scheme is achieved by running the verification as well as opening algorithm.

4 Public Key Infrastructure.

In our scheme PKI environment is of two parts i.e separated from the group environment and including the private /public keys used in the group.

- For outside the group the certification authority will be assumed fully trusted (the only one).
- For inside the group this will be establish with the group members keys And certificates issued by issuer.

Such a PKI is formalized by a user-key generation algorithm which generates a personal public and private key pair ($upk[i]$; $usk[i]$) for a user U_i .

This PKI will provide the **non-repudiation**, but also **non-frameability** property even if the group authorities are corrupted, they cannot frame a group member.

Comparison:

Comparison of our Dynamic group blind signature scheme with A. Lysyanskaya and Z.Ramzan's scheme[4]

	Dynamic group blind signature scheme	A. Lysyanskaya and Z.Ramzan's scheme
Blindness	✓	✓
Two authorities	✓	X
Trust levels	✓	X
Three Key Requirement	✓	✓
Transferability	✓	X
PKI	✓	X
Divisibility	✓	X

Chapter 6



Conclusion and Future Enhancement

6. Conclusions and Future Enhancement

We have implemented our Dynamic Group Blind Signature Scheme: A Basic model of e-cash software on the Pentium Machine with in a local host environment. But this model can be practically implemented over the internet on secure channels. we discuss below the conclusion of our model.

6.1 Conclusion

Previously, no group digital signature scheme used the blind signature protocol to achieve the anonymity of content of the message with two separate authorities for opening the signature at the time of dispute and for issuing. Membership to new members and then for implementing it in the E-Banking .which is the emerging technology all over the world. Our scheme results the decrease of high level of trust in one and only one authority for all the managing tasks of the group. In our scheme neither the number nor the identities of group members are fixed at the setup phase. The technique we adopted not only reduces the uncomfortable high degree of trust but also, but also reduces the work load of single entity of the group which is an achievement, with special reference to the Group Blind Signature Schemes. Our scheme, the Dynamic Group Blind Signature Scheme, also results in the efficient join-Issue protocol without need of certificate from third party for checking the authenticity of issuer and client as well .Hence for e-banking we create the rules for its entities and other rules for e-transactions.

5.2 Future Enhancement

In our scheme we don't give the idea of member revocation by the issuing manager. We don't consider the security oracles for any adversary as described in [8] for dynamic group signature scheme. Open algorithm returns the claim of a correct signer. This claim can be judged by the judge algorithm. The judge algorithm simply checks if the proofs open algorithm is correct. So to check this claim Judge Algorithm can also be implemented. In future we are planning to add the revocation algorithm.

References



References

- [1]. **Handbook of Applied Cryptography**, by A. Menezes, P. van Oorschot, and S. Vanstone, CRC Press, 1996.
- [2]. **How Digital Signatures Work: Digitally Signing Messages** By Svetlin Nakov
- [3]. http://en.wikipedia.org/wiki/Microsoft_.NET_Framework.
- [4]. **ON THE ANONYMITY OF ELECTRONIC CASH** *Bart De Win*
Department of Computer Science, K.U.Leuven
- [5]. R. Rivest, A. Shamir, and L. Adleman. "A method for obtaining digital signatures and public-key cryptosystems". *Communications of the ACM*, 21(2):120–126, Feb. 1978.
- [6]. S. Goldwasser, S. Micali, and R. Rivest. "A digital signature scheme secure against adaptive chosen-message attacks". *SIAM Journal of Computing*, 17(2):281–308, April 1988.
- [7]. D. Chaum, R. Rivest, and A. Sherman. "Blind signatures for untraceable payments editors". *Advances in Cryptology { CRYPTO ' 82*, Lecture Notes in Computer Science, pages 199–203. Plenum Press, New York and London, 1983, August 1982.
- [8]. Bellare, Shiy, Zhang. "Foundations of Group Signatures: The Case of Dynamic Groups". *Topics in Cryptology CT-RSA '05*, Lecture Notes in Computer Science, A. Menezes ed., Springer-Verlag, 20
- [9]. A-Lysyanskaya, Z. Ramzan. "Group blind signature: A scalable solution to electronic cash". *Financial Cryptography (FC'98)*, Lecture Notes in Computer Science, vol 1465, Springer-Verlag, pp184-197, 1998

- [10].David Pointcheval and Jacques Stern “ **Security Argument for Digital Signature and Blind Signatures**” *Journal of Cryptology*, Volume 13, Number 3. Pages 361-396, Springer-Verlag, 2000.
- [11].J. Camenisch and M. Michels.”**A group signature scheme with improved efficiency**”. In *Advances in Cryptology | ASIACRYPT'98*, vol. 1514 of LNCS, pp. 160-174, Springer-Verlag, 1998.
- [12].J. Kilian and E. Petrank.”**Identity escrow**” In *Advances in Cryptology | CRYPTO'98*, vol. 1642 of LNCS, pp. 169-185, Springer-Verlag, 1998.
- [13].D. Coppersmith.” **Finding a small root of a bivariate integer equation; factoring with high bits known**”. In *Advances in Cryptology | EUROCRYPT '96*, volume 1070 of LNCS, pages 178-189. Springer Verlag, 1996.
- [14]. J. Camenisch and M. Michels. “ **A group signature scheme based on an RSA-variant**”. Technical Report RS-98-27, BRICS, University of Aarhus, November 1998
- [15].J. Camenisch and M. Stadler”**Efficient group signature schemes for large groups**” In B. Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, volume 1296 of *Lecture Notes in Computer Science*, pages 410-424. Springer Verlag, 1997.
- [16].C.P.Schnorr”**Efficient signature generation for smart cards**”.*Journal of Cryptography* ,4(3):239-252,1991 .
- [17].D. Chaum and E. van Heyst.” **Group signatures**”.In D. W. Davis, editor, *EUROCRYPT'91*, volume 547 of LNCS, pages 257-265. Springer-Verlag, 1991.
- [18]. “**The Mathematics of Public-Key Cryptography**”, by Martin E. Hellman; *Scientific American*, August 1979.
- [19] M. Bellare, D. Micciancio and B. Warinschi. **Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions**. *Advances in Cryptology, EUROCRYPT 03*, *Lecture Notes in Computer Science* Vol. 2656, E. Biham ed., Springer-Verlag, 2003.

Appendix



E-banking System:

User Interface

1- Set up Phase

```

C:\Program Files\Java\jdk1.6.0_85\bin> java net.Group HBL a B
Group Creation phase Start...
Generating Group Key...
.....
Generating Group Public Key...
The name of your group is HBL
.....
Group signature key has been generated
Group verification key has been generated
.....
2.....
Generating certificate creation key for member...
You choose MR/Ms. as our Issuing Manager
.....
Issuing membership certificate key has been generated
Certificate verification key has been generated
.....
3.....
Generating Decryption key for opener...
You choose MR/Ms. as our Opening Manager
.....
Signature Opening key has been generated
Group Encryption key has been generated
1.Group Verification will be published for worldwide.....
2.Certificate Verification will be published for group members domain...
3.Group Encryption will be published for group members.....

```

2- Personal Key Pair Generation

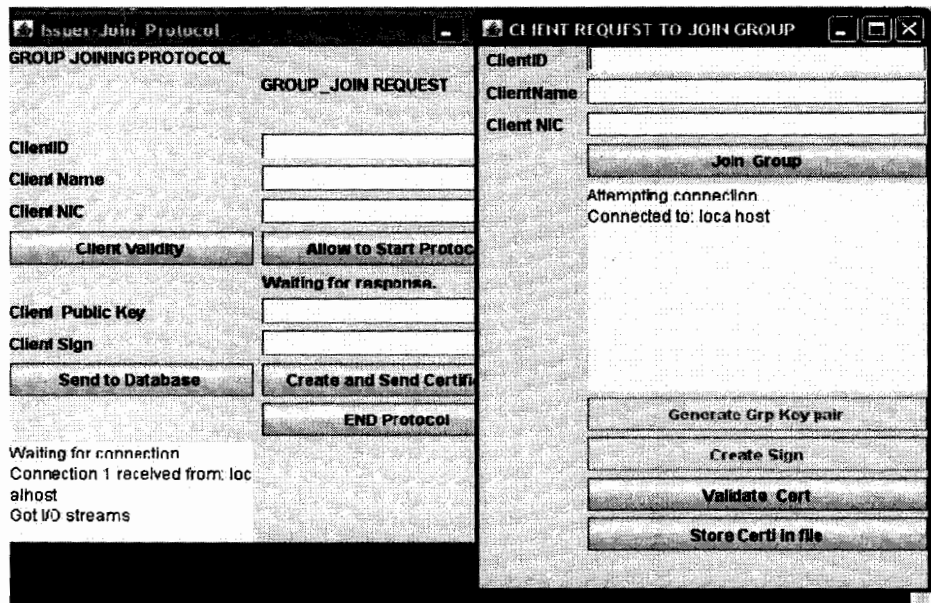
```

Command Prompt - java PersonalKeyPair Ayesha
47885425642298164246852225917211908661948222550488582120881151150114886947
29684129101104014651012108520880870662508151948120812081120807080507108520802
public exponent: 65537
.....
The default specification of the Private key is...
Don't use private CRT key: 1024 bits
modulus: 9529360220
39485292261433589424939963
1168128582428854256422916423
1308369492683512910110446465
08252687
public exponent: 65537
private exponent: 413246814
0802249822613049104011593432
01838112489254121242019132232
0856111222251948843011355072
26452785
private p: 110911138
2412991625459402180260818411
255971823682449
private q: 948882227
08811281348524241549274425
8192222222244
private exponent p: 872627688
02880917606306504762411910855
0626828245633
private exponent q: 416132620128927413222161129842588252026629887130869222406
29196429822011264445015412761442862414037592310615231884257021629655097221404206
0020634813259
CRT coefficient: 152880627689158668672802126708881228110142828001150408
18031329812481250662512504874122540810411050181811662018442951581162167011308141
022221582402

```

3 -Group Joining Phase

Group Join Protocol



4-Client Request to Join Request

