Blind Digital Signatures, Group Digital Signatures and Revocable Anonymity

Network Security Course Project

Vijay Gabale (07305004)

Ashutosh Dhekne (07305016)

Sagar Bijwe (07305023)

Nishant Burte (07305915)

Department of Computer Science and Engineering,

IIT Bombay

Acknowledgements

The authors would like to thank Madhumita Chatterjee for providing us initial impetus to pursue the implementation of the Group Digital Signature. Her confidence in our implementation idea put to rest many apprehensions we had about the feasibility of our idea. Our healthy discussions with Prof. Bernard L Menezes gave a definite direction to this project. He encouraged us to look for innovative and practical applications of the security area we were studying.

Harin Vadodaria has mentored this project and provided many insights into presentation and general handling of this project. It was his idea to experiment Group Digital Signatures with different key sizes. This experiment now forms an integral part of this work.

We express our sincere thanks to Omkar Ambardekar, Saumabha Pal, Pratik Gupta, Pranav Dhekane, Nikhil Halkude, Chinmay Soman, Dhiraj Mundada for their views regarding the feasibility and application of the Group Digital Signature Scheme & providing valuable feedback.

Table of Contents

Abstract

Digital Signatures have found wide acceptance and usage due to digitization of documents and communications. Many flavors of the classical digital signature scheme are available and we present a brief study of Blind Digital Signatures and Group Digital Signatures. We also present a novel application of Group digital Signatures in the corporate world. We have implemented this application using Java Cryptographic Architecture. Our experiments show that Group Digital Signatures can be employed with relatively little overhead for large confidential documents. This can also alleviate burden off PKI by reducing number of public keys required for each individual.

Keywords

Blind digital signature, partially blind digital signature, Signature of knowledge, Group digital signature, Signature of knowledge.

1 Introduction

Digitization of most manual processes has necessitated the use of digital signatures for authenticity of documents. Various needs of authenticity are known and this has led to many flavors of the classic digital signature scheme. In most cases, analogies with physical signatures and existing scenarios can be found while in few other cases, the applications are entirely new.

Bank notes are a type of application where we have a physical analogy. Anyone handling the note knows how it looks. In addition, it has a signature of the Governor of The Reserve Bank of India to guarantee its authenticity. However, the Governor or the Bank is not aware of where and when a particular note is spent. With slight modification in this physical analogy, like the ability of spending a received note, this application renders itself very naturally to what is known as Blind Digital Signatures.

Taking this analogy one-step further, we may have multiple branches of the same bank, or a group of banks, which may together offer the notes to their customers. Then, a merchant seeing a note must be able to check that the note is indeed valid, but need not know which branch issued it. Thus, we are in the need of a scheme where we have group members sign messages, which look as though signed by the group.

We look at the implementation of a simple Blind Digital Signature scheme in the next section. Then, we move on to yet another flavor called Partially Blind Digital Signatures & then to the details of Group Digital Signatures. In section 5, we describe our experimental setup to demonstrate the use of Group Digital Signatures (without blinding) in a corporate organizational structure. We also look at some analysis of the results. We then briefly brush upon the topic of Revocable Anonymity. In section 7, we shed some light on this topic through our theory blended with implementation-based angle. We conclude with our observations about the Group Digital Signatures we implemented.

2 Blind Digital Signatures

A Blind signature [CHAU82] requires 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. Now 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 major concern. Two such applications are online voting and electronic cash. We shall now look at the voting scheme in more detail.

2.1 Online Voting

In an online voting scenario, on one hand, the voting center will be concerned of duplicate or fraudulent votes, and on the other hand, the voter will be concerned about the anonymity of his vote. The obvious solution then is to have a trusted Election Authority that will authenticate the owner of a vote. The voting center happily accepts any votes with the signature of the election authority. The election authority is responsible for checking validity and non-duplicity of votes.

The concern of the voter, however, is not yet solved. This is when blind signatures are useful. The election authority should be able to sign on a blinded vote sent by the voter and the voter should be able to extract a valid signature on the original unblinded vote.

2.2 Blinding using RSA

The RSA scheme can be used for effectively achieving the functionality required above. The voter computes the hash of his message H(m) and multiplies it with a random number r raised to the RSA public key of the election authority. Effectively, he blinds the vote creating $B = H(m).r^e$. He then sends this blinded vote along with his credentials to the election authority. The election authority checks the credentials of the voter and if found valid, signs on B. This sign is returned to the voter. The entity that is returned to the voter is $B^d = H(m)^d.r^{ed} = H(m)^d.r$. Since the voter knows the random number r, he can compute $H(m)^d$ from this entity. The voter has thus successfully obtained a valid signature on his unblinded message. When he sends this message to the voting center, it can verify the signature on the vote using the public key of the election authority. This procedure is depicted in Figure 1.

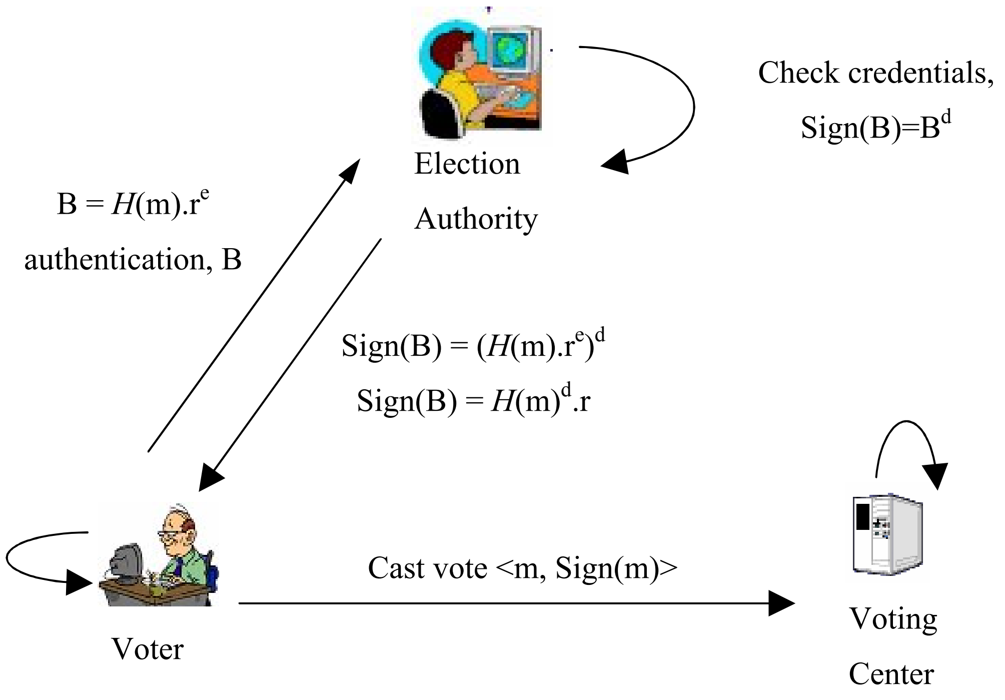

Figure 1 Blinding using RSA

2.3 Need for Grouping

This solution is elegant and simple, but not scalable. With thousands of voters trying to obtain blind signatures from the election authority, the system would become overloaded. We would like to have a number of election authorities who may issue the blind signatures and yet the voting center should be able to verify the signature using a single public key. We then, would require the notion of group signatures.

After an overview on an interesting variation of blind digital signatures, called partial blind digital signatures, we shall discuss about group signatures in detail.

3 Partially Blind Digital Signatures

The partially blind signature scheme [CHIE01] is a variation of the blind digital signature scheme in which the signer can impose common information on the signature such that the verifier needs the message, the common information and the signature to check the validity of this signature. This common information is non-removable from the signature. So only modifying this part without changing the signature is impossible. Due to its un-traceability and partial blindness property, the partially blind signature plays an important role in many e-commerce applications.

In the simple blind digital signature scheme, the blindness property allows the signer no opportunity to impose any common information on the signature. However, in some applications, a blind signature embedded with some common information is required. For example, the common information may represent the amount of an e-cash or the validity term of the signature. We shall now see this application in detail.

3.1 Scenario

Firstly, the requester prepares a blinded data and the common information, and sends them to the signer. If the signer agrees on this common information then he signs the blinded data with the common information imposed on the signature. The requester, then, derives the signature from the signed message without being able to remove or change the imposed common information. To verify the signature successfully, the signature holder should hand in the message, the signature and the agreed common information. Therefore, the agreed common information would be genuinely shared among the requester, the signer and the verifiers.

3.2 Application to E-cash

In the e-cash scenario (with simple blind digital signature scheme), the bank has to keep all the e-cashes used until now to avoid the double spending of money. With the use of partially blind digital signature, this can be avoided by carrying the common information as ‘expiry date of the e-cash’. The bank assures that the signed e-cash carries this agreed common information. So now, the bank needs to keep only the still-alive e-cashes in the databases to check double spending. Therefore, banks do not have to maintain a large database. The expired e-cashes could be eliminated from the databases without much trouble.

The phases are involved in this scheme are discussed below.

3.2.1 Money Withdrawal

1. The requester creates a blinded electronic coin C. This coin consists of some serial number, as well as some other information pertaining to currency, such as value.
2. This is then sent to bank for applying partially blind digital signature along with the common information—expiry date.
3. Bank reads the common information then applies signature on the 'coin + common information'.

3.2.2 Spending

1. Requester gives the coin C along with the bank's signature on C to the vendor.
2. Vendor checks whether the bank's signature on C is authentic using bank's public key.
3. If this checks out, vendor gives the coin to his bank for deposit, and awaits the bank's response.

3.2.3 Deposit

1. Bank verifies coin's validity by checking the signature on the coin.
2. Bank also checks the expiry date on the coin to validate/invalidate the coin as appropriate.
3. If this is validated then it checks the database of already spent coins to do one more validation.
4. If this is also validated, bank credits vendor's account with appropriate amount. In addition, it updates the database of already spent coins.
5. Now vendor can give requester the requested merchandise.

4 Group Digital Signatures

In a group digital signature scheme [RAMZ99], [CAME97], members of a given group are allowed to sign on behalf of the entire group. The signatures can be verified using a single group public key. In addition, once a document is signed, no one except a designated group manager should be able to determine which particular group member actually signed it. Group signatures are also designed so that no group member can forge another member's signature on a given document.

4.1 Organizational structure & group signatures

For example, suppose the CEO of a company wants some of his employees to validate price lists, press releases, or digital contracts on behalf of the entire company. In this case, he can set up a group signature scheme, and act as the group manager. Then the employees can sign or validate various documents on behalf of the entire company. By using this approach, the company's internal structure will be concealed, and the customers would only have to use a single company-wide public key to verify the signatures. Moreover, only the manager 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 an employee to fake the signature of another employees. In fact, it would also be impossible for the group manager to fake the signatures of the individual group members. The group digital signature scheme explained further sheds light on a new application of group digital signatures to organizational hierarchical structures.

4.2 Group Signatures: Wish list

Any group signature scheme must satisfy certain security requirements. The group signature scheme that we have implemented abides following security wish list

4.2.1 Unforgeability

Only group members can issue signatures that are verifiable by the group public key.

4.2.2 Conditional Signer Anonymity

Anyone can easily check that a message signature pair was signed by some group member, but only the group manager can determine which member issued the signature.

4.2.3 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.

4.2.4 Security against Framing Attacks

No subset of group members (perhaps including the group manager) can sign a message on behalf of another group member.

4.2.5 Coalition Resistance

No subset of group members (perhaps including the group manager) should be able to collude and generate valid group signatures that are untraceable.

4.3 Procedures in Group Signature Scheme

We now describe the original group digital signature scheme of Camenisch and Stadler [CAME97] that we have implemented. This group digital signature scheme consists of the following procedures.

4.3.1 Setup

This procedure is executed by the group manager in which he chooses two large prime numbers & calculates RSA public & private keys. He also generates a cyclic group of prime order with two generators ‘g’ & ‘a’.

4.3.2 Join

This is an interactive protocol between the group manager and the new group member. It produces member's secret key x, and his membership certificate A.

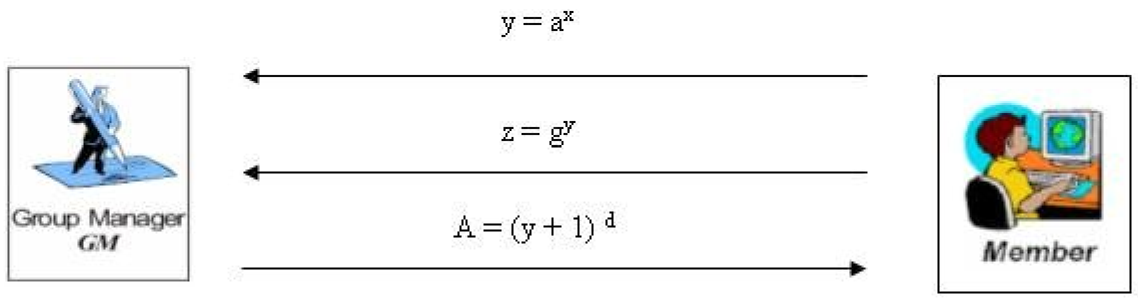

Figure 2 Join Protocol

Here, group member chooses a random number x as his secret key (the collision with other secret keys of group member is taken care of) & proves his signature of knowledge. He then sends z, encrypted with pre-shared secret key as an identity check, after which group manager issues him certificate A signed with private key d.

4.3.3 Sign

Here a group member take message m & sign it using secrete key x & certificate A.

4.3.4 Verify

This is an algorithm which on input (m, sign, group public key Y), determines if ‘sign’ is a valid signature for the message m with respect to the group public key Y.

4.3.5 Open

Open procedure is an algorithm, which, on input (m, sign) determines the identity of the group member who issued the signature ‘sign’ on the message m.

4.4 Signature of knowledge

A signature of knowledge is a construct in which a signer can use knowledge of some secret information to sign a message digitally. This differs from the notion of a proof of knowledge, which is a way for one person to convince another person that he knows some fact without actually revealing that fact. *In a signature of knowledge, the signer ties his knowledge of a secret to the message being signed.* A signature of knowledge is used for the purpose of both signing a message and proving knowledge of a particular secret.

In the join protocol, a group member picks up a number x as his private key. Now to acquire certificate from group manager, he has to prove the correspondence between him and his private key x. One way to achieve this is to send x as plain text. However, a member cannot afford revealing his private key. Instead he can use 'infeasibility of calculating discrete log' and send $a^x = y$. Still manager would not believe the member since y might be any arbitrary number. Group manager has to be convinced that y is indeed calculated from x, i.e. this member is a valid possessor of secret key x. Unlike in digital signatures, where this task can be resolved by encrypting a message with private key and verifying it using a corresponding public key, we do not have anything like a public key pertaining to the secrete key x. Nevertheless, we do have a group public key. Can the group member sign, using x, attach the knowledge of x, y to the message and convince group manager the knowledge of x with the help of the group public key?

The answer is yes and through a very elegant concept call 'Signature of Knowledge'. Our goal is to help group member convince group manager the knowledge of x using a group public key consisting of parameters of initial set up phase. Here group member ties his knowledge of x with the message to be sent to group manager as follows.

4.4.1 Signature

If *l* is the security parameter, the group member generates *ri (i = 1 to l)* random numbers and calculates corresponding factors $Pi = g^{ri}$ for each random number. Now signature consists of two constructs, a hash i.e., C and a list of factors calculated from random numbers mixed with secret key.

$C = H(m, y, g, P1\ldots l)$

$S_i = r_i \qquad if\ c\,[i] = 0$

$Si = r_i - x \qquad if\ c\,[i] = 1$

This tuple, ($C, S_1 \dots S_l$) is the signature of knowledge of discrete logarithm of y which is actually the secret key, x. The group member sends this tuple with message and y.

4.4.2 Verification

If group manager has to cross verify this signature, he must be able to recompute the hash C. All he has is $S_1 \dots S_l$, list of factors derived from x, which is indeed to be proven and group public key parameter a. He would re compute the P_i as follows

$$P_i = a^{S_i} \qquad if\ c\ [i] = 0$$

$$P_i = a^{S_i} * y \qquad if\ c\ [i] = 1$$

Since $a^x = y$, $P_i = a^{S_i} * y = a^{r_i - x} * a^x = a^{r_i}$ which is exactly how we had calculated P_i. Now since group manager is able to reconstruct P_i, he can compute the hash C & verify with received hash.

Here group member tied his identity with message and appended some clues for group manager to deduce that the group member indeed possessed the secret key with which he had signed the message. Had this been not the case, had group member used some different secret key x' and sent arbitrary y, group manager would not have able to verify the signature. On the contrary, if group member does not know discrete logarithm of y, x, it is infeasible for him to construct the tuple $(C, S_1 \dots S_l)$.

Another peculiarity about signature of knowledge, that we observed, is S_i's are derived depending on whether the i^{th} bit is 0 or 1 & the same fact is used to reconstruct Pi. This intuitively corresponds to non-interactive challenge response protocol at the group manager's end, where for each i^{th} bit of C, he challenges group member for S_i, which is actually sent to him by the group member as a form of message. Since probability of number of 0's & 1's would be close to ½, the attacker without knowing x has literally no chance to compute the tuple (C, $S_1 \dots S_n$).

4.5 Revoking anonymity of the group member

Once the group manager is convinced of knowledge of x, without actually looking at it, he issues a certificate A to group member. If we consider organizational structure where each group is assigned a project pertaining to a client, whenever a group member has to send a message to a person outside the group, he will produce signature of knowledge of x blended with the message. The scheme is developed such that the client is able to verify this signature using group public key.

We observe that, all the client comes to know is that a document has come from a particular group. The legitimacy & authenticity of the group is verified. However, client never comes to know about the identity of the group member who actually signed the document. So can the group member exploit this fact to create fraudulent document & send to the client? Well, the answer would have been yes but not without been caught.

During the join procedure, group member has to sign ‘y’ using a pre-shared secret key between him & group manager. (In a way, he ties his identity with message.) Now group manager keeps a record of which ‘y’ belongs to which member. He does not actually know x, but when the client reports the fraudulent document along with the signature, the group manager looks in the table for a member who could come up with this signature. He may then also prove a member guilty in court.

This is quite different from the case where if client cannot verify the signature – here this means that the document is not at all signed by the group. The scheme thus abides to security wish list mentioned previously.

4.6 Quasi coalition attack

We observe here that, group member was given liberty to choose x. Can this loophole be exploited? Yes, and this gives rise to a very subtle attack called quasi coalition attack.

Since x can be chosen at will, certain group members can collude and choose their secrete keys such that they produce a valid secret key with pertaining certificate. If a document is signed using this key, the document becomes untraceable. Meaning that, open procedure will fail to point out any of group members since entry corresponding to such a key does not exits at manager’s side. In worst case, this blot might go to manager since he is the one who issues certificates. One possible instance of such attack is explained in appendix A.5.

A simple remedy for this attack is let group members choose their random number x but force group manager to mix a random factor u so that the secret key become x' = x +u. This is how modified join protocol works.

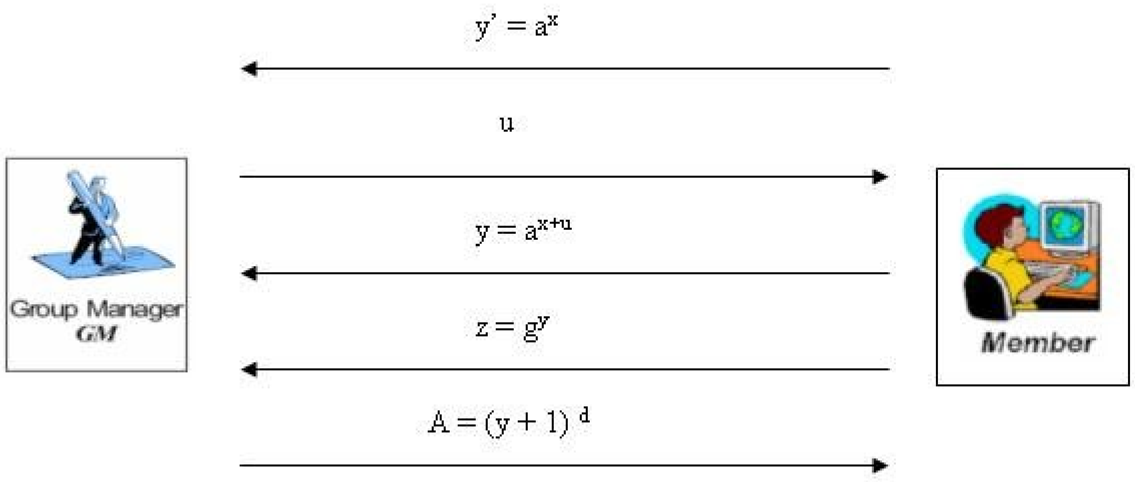

Figure 3 Quasi Coaliation Attack-Remedy

So now, group members can no more collude & combine their secret keys to launch coalition attack.

5 Revocable Anonymity

Another state-of-the-art application of group and blind digital signatures apart from banking and voting is in anonymity services provided by set of servers called anonymisers, which allows user to connect to the rest of the internet world without revealing his identity and private information [KOPS06].

In case of serious crimes, however, anonymity services may be forced by law to collect connection data and deanonymise some of their users. Moreover, the revocation of anonymity should preserve the privacy of all lawful users, especially without the need of logging all communication data. Therefore, a privacy-friendly solution for incorporating revocation in an anonymous communication system will be not to reveal the identity of a user to any other entity involved in the revocation procedure, but the law enforcement agency alone. No user will need to provide more identifying information than his connection (IP) address that is what he needs to communicate with the system anyway.

Here each user, whenever wants the service of anonymier, will execute join protocol. Here the group will be formed by the set of the users those are currently using the anonymity services. This is a dynamic group rather than a static one. Here group memberships are retained during the period of session only.

5.1 User login procedure

This procedure is quite similar to the join protocol of the group digital signature explained in Section 4.3.2. Additionally, the user may use the signature of a trusted third party over the user's private key and IP to prove his binding to IP address to the group manager. This is a blind signature and the signer (third party) may use cut and choose protocol for verifying the IP address of the user.

5.2 Sending messages anonymously

User can now send messages anonymously according to the Anonymiser protocol. The additional step he has to do is to sign the messages with his secret group signature key. A verifier V sitting between the last Anonymiser and the recipients will check for any "suspicious" messages. V will drop any unsigned message. If V detects a "suspicious" request, he demands the disclosure of the identity. V uses group's public key for this purpose.

5.3 Revoking anonymity

The prerequisite for revoking anonymity is that V gets a court order O. O contains a public key of the law enforcement agency L and a relation R, which says for every message m if m is "suspicious" or "good"; $R : \{m\} \rightarrow \{\text{“good”}, \text{“suspicious”}\}$.

If V detects a "suspicious" message m, revealing the identity of the sender works as follows

- V shows m, Sig(m) and O to group manager (GM).
- GM checks that R(m) = "suspicious" and verifies Sig(m).
- GM reveals that Sig(m) was done by Y and group certificate *cert* of Y.
- V verifies the certificate and shows m, Sig(m), O, *cert* to anonymisers $A_1, \dots, A_n$.
- Each Anonymiser Ai of these k Anonymisers checks that R(m) = "suspicious" and verifies Sig(m), *cert.*
- The k Anonymisers jointly proxy re-encrypt user's ID and and his binding to ID (signed by third party) using law enforcement's public key in O.One of the k Anonymisers sends second encryption (say E) and the proof P that both the encryptions decrypt to same content to V.
- V verifies P and sends E along with m Sig(m),*cert* and P to L.
- L checks that R (m) = "suspicious" and verifies Sig(m), *cert* and P.
- L decrypts E to ID and his binding.
- L verifies binding.

This scheme is zero-knowledge with respect to any entity involved in the revocation procedure but the law enforcement agency. Another advantage is, that the user needs to authenticate himself only once to anonymously send as many messages as he wants. Moreover, the very privacy-friendly user identification by his connection address may be sufficient for his authentication, as the responsibility to find the real identity behind the address may be assigned to the law enforcement agency.

6 Our Implementation

We have implemented the Group Digital Signature scenario discussed above to acquire an in-depth understanding of the scheme. We have used Java Cryptographic Architecture for this implementation. The JCA provided us with the basic cryptographic framework over which we had to build the entire scheme discussed in Section 4.3. Each of the entities in the scheme, namely the group manager, the group member, and the client, form three different components in our implementation.

6.1 Core Components

6.1.1 Group Manager

This component is responsible for setting up the initial parameters of the cryptographic scheme. This includes computation of a RSA public key – private key pair, and the parameters of the Group public key. These parameters are persisted to the group manager's computer and must be maintained throughout the lifecycle of this scheme. This enables the group members to approach the manager for joining the group. In the real world, a Group Manager would invoke this component and thus become ready for join requests from his group members. In fact, the implementation allows the manager to be oblivious of most of the intricacies of the scheme.

The group manager is provided with a facility of revoking the anonymity of a member if he comes across a fraudulent document analogous to revoking identity of a user responsible for sending "Suspicious" messages in Section 5.3. For this, the manager maintains a list of ids and their corresponding

6.1.2 Group Member

The group member component obtains a membership certificate from the manager by executing the Join protocol. The secret key is computed by the member and is now available for signing documents on behalf of the group. The member is expected to perform the join protocol only once.

The employee may now sign numerous documents, which will be verifiable by external clients through the group public key.

6.1.3 Client

The client obtains signed documents and verifies their signature. If he finds a document that is authentic but fraudulent, he can send it to the group manager. The internal structure of the group remains unknown to the client even in this case.

6.2 Experimental Observations

We observed that the size of the document to be signed does not play much role in the time taken for issuing signatures or verifying them. This happens because the signature is effectively cast on the hash of the message, which is always of a fixed size.

A change of the security parameter however, affects both signing and verifying operations drastically. Moreover, the size of the signature generated also increases drastically with the security parameter. We found that a security parameter of 256 bits generates a signature file of about 44 KB while that of 512 bits generates a signature file of more than 132KB. In addition, the time required to sign and verify as we migrate from 256 bits to 512 bits increase by almost eight times.

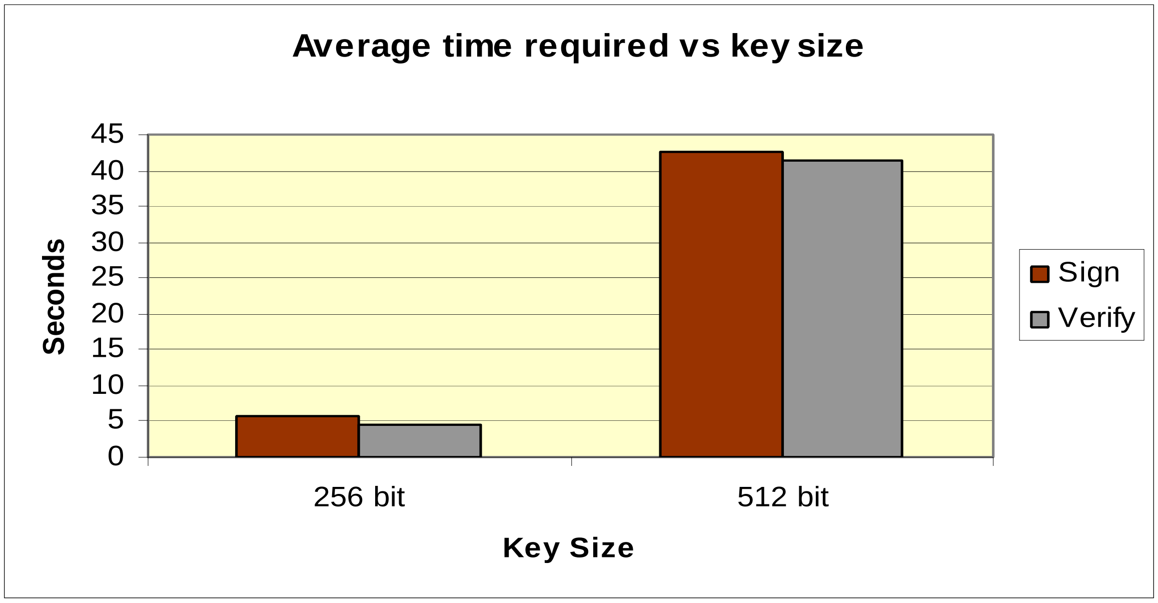

Figure 4 Comparison of security parameter

7 Critique

Through this project, we have understood subtle cryptographic principles and protocol design methodologies.

The study of partially blind signatures gave us the idea of linkage between the common information, which is not blinded, and the signature. Even if the common information appears as plain text, any alteration to this invalidates the partially blind signature. This common information acts like an agreement between two parties & this could be exploited when designing agreement phase of cryptographic protocols.

The principle, which drives Group Signatures, is the signature of knowledge, which lets a signer convince verifier that he indeed possesses the secret with which he signed the message by attaching some clues with the message itself. In this way, number of group members can have their separate private keys but we require a single group public key to verify the signature made by any member. We believe that this concept is most suitable to existing organizational structures that have been developed over the years where the communication between a customer or client and a group inside an organization can be made trustworthy by incorporating group signatures. Every document would then be signed by any of the group members and thus the customer is assured of authenticity of the communication. As a byproduct, we discovered that in this case, each member need not have a separate private-public key pair associated with PKI but only a secret key & certificate given to him by the group manager locally. This scenario when deployed in the real world will reduce the burden off PKI substantially since now it needs to maintain public keys of only group managers and not of whole bunch of group members.

We were exposed to the fact that how an inadvertently induced loophole in the protocol causes havoc—Quasi Coalition attack. Indeed protocol design remains the most challenging task in current web applications and cryptographic schemes.

We also tackled cryptographic implementation constraints one of which was generating a large prime number. There is no simple yet efficient algorithm [STAL06], which accomplishes this task. The Java libraries provide a BigInteger class, which allows us to create probable primes. We understand that the prime number generated will fail with a probability of 1/ (2^{100}), which is almost zero for all practical purposes. Also finding out a generator for cyclic group was a non-trivial task, which we eased out by choosing the group to be of prime order so that we could use the test to find a primitive root to generate a valid generator.

Experimental evaluations also gave us useful insights as to why cryptographic operations are considered costly. As we increase the security parameter, the time and memory required to sign or verify signatures shoots up substantially. This gives the trade off between the strength of security and resource overhead. The computations of double discrete log [Appendix A.2] and root log [Appendix A.3] become considerably CPU and memory intensive operations if we want a stronger security scheme. This also increases user latency of signing or verifying the documents.

The legacy of Group, Blind, Partially blind signatures prompted us to find some more applications where we can apply them. Indeed, we found some applications that can be tuned and digitized by inculcating these schemes. For instance, if a student wants to apply for a patent through an educational institute like IITB, he would want IITB to sign on his documents, without actually looking at the idea (blind signatures). He would then send it to the Patent Granting Agency. Several insurance agents of an insurance company can be allocated separate private keys to sign on documents for their customers where these signs are verifiable using single company public key (group signatures).

8 Conclusion

We have proposed an innovative application of the Group Digital Signature scheme for large organizations that wish to hide their internal organizational structures from their clients and still provide their employees the power to sign documents. Apart from the traditional applications to banking and voting, this work proposes new prospects for the underlying signature scheme.

Our survey with employees of numerous organizations indicates that most organizations rely on simple hand-written signatures or non-authenticated emails for document exchange. With increasing awareness about security, we believe that the technique we implemented will earn real-world acceptance in the years to come.

The blinding scheme that we introduced in Section 2 is being used for banking applications. We feel that this coupled with Group Signatures will also find wide spread applications as increasingly more institutes seek to collaborate.

9 References

[RAMZ99] Z.A.Ramzan, *Group Blind Digital Signatures: Theory and Applications*, Master of Science, MIT, 1999. http://citeseer.ist.psu.edu/ramzan99group.html

[CHAU82] David Chaum, *Blind signatures for untraceable payments*. In *Proc. CRYPTO 82*, pages 199-203, New York, 1983. Plenum Press.

[CAME97] Jan Camenisch and Markus Stadler, *Efficient group signatures for large groups*. In *Proc. CRYPTO 97*, pages 410-424. Springer-Verlag, 1997. Lecture Notes in Computer Science No. 1294.

[KOPS06] S. Kopsell, R. Wendolsky, H. Federrath, *Revocable Anonymity. In Proc. ETRICS 2006,* pages 206-220, LNCS 3995, Springer-Verlag, Heidelberg 2006.

[CHIE01] H. Chien, J. Jan, Y. Tseng, *RSA-based partially blind signature with low computation*, in: IEEE 8th International Conference on Parallel and Distributed Systems, 2001, pp. 385–389.

[STAL06] William Stalllings, Cryptography & Network Security, Fourth Edition, Pearson Education, 2006

A. Appendix

A.1. Preliminaries

Let G be a cyclic group of order n generated by some g Є G (hence G = <g>). Let a Є Z_n^*. The scheme relies on the hardness of computing the following number theoretic primitives.

Definition 1

The discrete logarithm of y Є G to the base g is the smallest nonnegative integer x satisfying $g^x = y$.

It is a widely held assumption that G, g, and n can be chosen such that the discrete Logarithm problem is infeasible to solve.

Definition 2

The double discrete logarithm of y Є G to the bases g and a, is the smallest non-negative integer x satisfying: $g^{a^x} = y$; if such an x exists.

We conjecture that G, g, n, and a can be chosen such that the double discrete Logarithm problem is infeasible to solve.

Definition 3

An e-th root of the discrete logarithm of y Є G to the base g is an integer x satisfying $g^{x^e} = y$; if such an x exists.

A.2. Signature of knowledge of Double Discrete Log

A signature of knowledge of a double discrete logarithm of y to the bases g and a, on message m, with security parameter l denoted $SKLOGLOG_l[\alpha|y= g^{a^{\alpha}}](m)$, is an $(l+1)$-tuple $(c,s_1, \ldots, s_l)$ Є $\{0,1\}^l \times Z^l$ satisfying the equation

$$c = \mathcal{H}_l(m, y, g, a, P_1, \ldots, P_l), \ \ where\ P_i = \begin{cases} g^{(a^{s_i})} & if\ c[i] = 0 \\ y^{(a^{s_i})} & otherwise \end{cases}$$

A.3. Signature of knowledge of e^{th} root of Discrete Log

A signature of knowledge of an e^{th} root of discrete logarithm of y to the bases g , on message m, with security parameter l denoted SKROOTLOG$_l$$[\alpha | y = g^{\alpha^e}](m)$, is an $(l+1)$-tuple $(c, s_1, \ldots, s_l) \in \{0,1\}^l \times Z^l$ satisfying the equation

$$c = \mathcal{H}_l(m, y, g, e, P_1, \ldots, P_l), \ \ whereP_i = \begin{cases} g^{(s_i^e)} & if\ c[i] = 0 \\ y^{(s_i^e)} & otherwise \end{cases}$$

A.4. Cut and Choose Protocol

This protocol is used to get blind sign on the message, still assuring the signer that message is not suspicious.

Consider an example where Alice wants blind signature of Bank on her coin. Alice and the Bank then execute a cut and choose protocol. Here, Alice first chooses some number, say, 9, and prepares these many coins $C_1 \ldots C_9$. Each coin should be identical; the only difference between them should be their serial numbers. Then, Alice blinds each of these coins, and sends them to the Bank. The bank picks all but one of the coins, and tells Alice to reveal the blinding factor for those coins. That is, it asks Alice for the appropriate information so it can unblind these coins. After Alice reveals the blinding factors for these eight coins, the bank unblinds the coins, and examines them. If they are all of the correct form, then the Bank applies a blind signature to the remaining coin, and sends it to Alice. The bank has high assurance that the remaining coin is of the proper form—because if Alice included even a single improper coin among the first nine, then she would be caught with probability of 8/9.

A.5. Quasi Coalition Attack

Let us understand how the carefully chosen private keys lead to Quasi-Coalition attack.

Private Key	Certificate
x	$A=(a^{x} + 1)^{d}$ $=a^{xd}(1 + a^{-x})^{d}$
-x	$B=(a^{-x} + 1)^{d}$
rx	$C=(a^{rx} + 1)^{d}$
-rx	$C (A B^{-1})^{-r}$

Three group members choose their secret key as *x*, *-x* & *rx* & receive the corresponding certificates. However, it so happens that these certificates when combined gives rise to a valid certificate with a valid secret key *–rx*.

$$\begin{aligned} C (A B^{-1})^{-r} &= C (a^{xd}(1 + a^{-x})^{d} ((a^{-x} + 1)^{d})^{-1})^{-r} \\ &= C\, a^{-xdr} \\ &= (a^{rx} + 1)^{d}\, a^{-xdr} \\ &= (a^{-rx} + 1)^{d} \end{aligned}$$

This certificate would have been issued had the private key of group member been (-*rx*).