A NEW STRONG PROACTIVE VERIFIABLE SECRET SHARING SCHEME WITH UNCONDITIONAL SECURITY

A THESIS IN

Computer Science

Presented to the Faculty of the University
of Missouri-Kansas City in partial fulfillment of
the requirements for the degree

MASTER OF SCIENCE

by

PRIYANKA KONERU

B.Tech., Koneru Lakshmaiah Univerity, 2008

Kansas City, Missouri

2010

A NEW STRONG PROACTIVE VERIFIABLE SECRET SHARING SCHEME WITH UNCONDITIONAL SECURITY

Priyanka Koneru, Candidate for the Master of Science Degree

University of Missouri-Kansas City, 2010

ABSTRACT

In secret sharing scheme, the master secret and all the private shares (which are distributed by the dealer to the shareholders) are the two secrets which are to be maintained confidentially. In all the secret sharing schemes proposed till date, private shares are reused to reconstruct the master secret. But we proposed a new way of Proactive Secret Sharing Scheme in which, instead of renewing the private shares frequently at the beginning of each timeslot during the share renewal process, each time master secret is renewed. In this way private shares can be reused for a longer period of time and to construct different master secrets. In addition, after each renewing process, shareholders can work together to verify that their private shares are consistent without revealing private shares. We also proposed protocols to generate and renew master secret, authenticate public shares of the master secret, add or revoke shares and change threshold of the master secret. Thus an enhancement to Proactive Secret Sharing is proposed in this thesis and this unique feature simples the implementation of PSS as the change is to be made only to the master secret (central server) without effecting all private shares.

APPROVAL PAGE

The faculty listed below, appointed by the Dean of the School of Computing and Engineering, have examined a thesis titled "A New Strong Proactive Verifiable Secret Sharing Scheme with Unconditional Security", presented by Priyanka Koneru, candidate for the Master of Science degree, and hereby certify that in their opinion it is worthy of acceptance.

Supervisory Committee

Lein Harn, Ph.D, Committee Chair
School of Computing and Engineering

Yugyung Lee, Ph.D
School of Computing and Engineering

Vijay Kumar, Ph.D
School of Computing and Engineering

TABLE OF CONTENTS

Contents Page

ABSTRACT iii

ILLUSTRATIONS vii

GLOSSARY ix

ACKNOWLEDGEMENTS x

Chapter

1. INTRODUCTION 1

1.1 Proactive Secret Sharing 3

1.2 Contributions 6

1.3 Thesis Outline 8

2. RELATED WORK 9

2.1 Basic Schemes 9

2.1.1 Shamir's Secret Sharing Scheme 12

2.1.2 Proactive Secret Sharing Scheme 19

2.1.2.1 Ostrovsky and Yung 23

2.1.2.2 Herzberg 24

2.1.2.3 Cachin et al.'s Asynchronous Scheme 28

3. OUR SCHEME 29

3.1 Motivation 31

3.2 Contribution 32

3.3 Share Generation and Authentication 33

3.3.1 Private Share Generation 33

3.3.2 Private Share Generation Protocol 33
3.3.3 Public Shares Authentication 34
3.3.4 Public Shares Authentication Protocol 34
3.4 Master Secret Generation/Renewal and Reconstruction 35
3.4.1 Master Secret Generation/Renewal 35
3.4.2 Master Secret Generation/Renewal Protocol 36
3.4.3 Master Secret Reconstruction 37
3.4.4 Master Secret Reconstruction Protocol 38
3.5 Verifiable Secret Sharing 39
3.5.1 Strong Verifiable Secret Sharing 46
3.5.2 Strong Verifiable Secret Sharing Protocol 47
3.6 Shares Adding/Revoking Protocol 50
3.7 Changing Threshold Protocol 51
3.7.1 Decreasing the Threshold 51
3.7.2 Increasing the Threshold 51
3.7.3 Strong t-consistency Check 52
4. CONCLUSION 53
4.1 Open Problems and Future Work 54
REFERENCES 55
VITA 59

ILLUSTRATIONS

Figures Page

2.1 Share Generation Protocol 16

2.2 Secret Reconstruction Protocol 17

2.3 Share Renewal Protocol 27

3.1 Private Share Generation Protocol 33

3.2 Public Shares Authentication Protocol..... 34

3.3 Master Secret Generation/Renewal Protocol 36

3.4 Master Secret Reconstruction Protocol 38

3.5 Unconditionally Secure VSS Protocol..... 42

3.6 Feldman VSS Scheme..... 44

3.7 Benaloh's Scheme..... 45

3.8 Strong Verifiable Secret Sharing Protocol..... 48

TABLES

Table Page

2.1 Lagrange's Interpolating Formula..... 18

3.1Constant Term of the Interpolating Polynomial 37

GLOSSARY

SSS:	Secret Sharing Scheme
PSS:	Proactive Secret Sharing
VSS:	Verifiable Secret Sharing
APSS:	Asynchronous Proactive Secret Sharing
SVSS:	Strong Verifiable Secret Sharing
SVPSS:	Strong Verifiable Proactive Secret Sharing
s	Secret
t	Threshold
D	Dealer
DoS	Denial of Services
RSA Algorithm:	Rivest, Shamir and Adleman Algorithm
PKI Schemes:	Public Key Infrastructure Schemes

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my thesis advisor Dr. Lein Harn for his most valuable suggestions and encouragement without whom this work would have not been possible. I appreciate his generosity and valuable time that he spent to discuss and clarify my doubts that came up during the course of this work.

I am very thankful to Dr. Yugyung Lee and Dr. Vijay Kumar for serving as members of my thesis committee.

I take this opportunity to offer my gratitude to my parents Mr. Venkateswara Rao Koneru, Mrs. Padma Koneru, and my dearest brother Mr. Sonesh Koneru for their love, encouragement and support.

I sincerely appreciate the encouragement and guidance given by my best friend Mr. Sri Harsha Jasti though out my Masters Career.

Finally, I would like to thank my academic advisor Ms. Coretta Carter, all my teachers and professors in my school and college and all my friends who have always been there for me.

CHAPTER 1

INTRODUCTION

Secret sharing schemes protect the secrecy and integrity of information by distributing the information over different locations. The (*t, n)* threshold secret sharing schemes were introduced by Shamir [29] and Blakley [6] independently in 1979 for protecting the cryptographic keys. In a (*t , n)* threshold secret sharing scheme, a secret *s* is divided into *n* shares by a dealer *D*, and is distributed among *n* shareholders $P = \{P_1, P_2 \ldots\ldots P_n\}$ in such a way that at least *t* shares from the shareholders are required to reconstruct the secret. Shamir's (*t, n)* secret sharing scheme is information theoretically secure; as fewer than *t* shares can gain no information about the secret.

In Shamir's (*t, n*) threshold scheme, a dealer generates *n* shares based on a $(t - 1)^{th}$ degree polynomial. Shamir's Secret Sharing Scheme assumes that the dealer who divides the secret and distributes shares to shareholders without making any mistake. Secret reconstruction is based on Lagrange interpolating polynomial of any *t* private shares. Since secret sharing was proposed initially by Shamir [29] and Blakley [6], research on this topic has been extensive. In the "classic" secret sharing schemes, there are assumed to be no faults in the system.

If the length of any share is equal to the length of the secret, then that secret sharing scheme is called ideal. If any unauthorized subset of shares provide absolutely no information about the secret in the information-theoretic sense, then the scheme is said to be perfect. Shamir proposed the first *(t, n)* threshold secret sharing scheme which is both ideal and perfect.

Shamir's secret sharing scheme is based on Lagrange interpolating polynomial and is information-theoritic secure (unconditionally secure). All shareholders who receive their shares from the dealer must unconditionally trust that the shares they received are valid. In 1985, Chor

et al [12] introduced a new notion called *verifiable secret sharing* (VSS) to the original secret sharing scheme. Using the verifiable property, all the shareholders can verify the consistency of their shares. Even if the dealer is malicious, VSS ensures that there is a well-defined secret which the shareholders can later reconstruct.

Later, Benaloh [2] developed an interactive VSS based on Shamir's scheme. Feldman [17] and Pedersen [26] developed non-interactive VSS using different commitment schemes. Feldman's scheme protects the secrecy of the secret in a computational sense while Pedersen's scheme protects the secrecy of the scheme in information-theoritic sense. After analyzing both Benaloh's VSS [2] and Pedersen's VSS [26] , Lin et al. [21] in his very recent publication stated that all shares in both schemes are *t*-consistent, but shares may still violate the security requirements of a secret sharing scheme(i.e., at least *t* or more than *t* shares are required to reconstruct the secret). So they presented a notion of strong VSS. A strong VSS scheme can ensure that:

(1). All shares are *t*-consistent, and

(2). All shares satisfy the security requirements of a secret sharing scheme.

The protection provided by the Secret sharing schemes is insufficient as the secret is stored for a long period of time. In many applications, secret needs to be stored for a long period of time. Intuitively, the secret can be protected by multiple shareholders using the secret sharing scheme. Given sufficient time, adversary can attack the shareholders to obtain the secret. We can overcome this limitation using Proactive secret sharing in which shares are refreshed periodically without compromising the secrecy of the shares. Ostrovsky and Yung [25] presented the notion of proactive security which refers to security and availability against the mobile adversary.

Herzberg et al. [19] proposed the first proactive secret sharing (PSS) scheme against the mobile adversary.

The main properties of Proactive secret sharing are:

- Shares are renewed frequently without compromising the secret to obtain new shares, so that even if the old shares are exposed, the adversary can know nothing about the secret.
- Lost or corrupted shares can be recovered without compromising the secrecy of the shares.

1.1 Proactive Secret Sharing

In proactive secret sharing, time is divided into multiple slots and shares are renewed at the beginning of each time slot; the new shares are completely different from the old shares. The secret is secure and is protected under the assumption that within each time slot the adversary can attack at most *t-1* shares. PSS uses Share generation protocol for the generation of new shares and old shares can be discarded, rendering useless information with the collection of fewer than *t* shares by the adversary. PSS uses Share recovery protocol to generate new shared if some of the old shares have been lost or corrupted. The share renewal protocol in proactive secret sharing relies on verifiable secret sharing (VSS) which allows shareholders to verify consistency of renewed shares.

Proactive secret sharing scheme based on Verifiable Secret Sharing (VSS), provides strong security for a secret sharing against the active attacker. It combines the secret sharing scheme with a periodical share update process to ensure the overall security of a system. Through update mechanism, old shares become useless. Even to steal a secret; however, an attacker needs to intrude on at least *t* participants during the same time period if security is maintained in a (*t, n*) threshold secret sharing scheme.

PSS imagine that servers are recovered at the rate at which they are compromised. A significant flaw in PSS is that "recovery" of servers that have been compromised or suffered irreparable hardware failures is problematic. Merely replacing the machine's hardware or software might eliminate the fault, but it does not undo all of the damage; each machine's identity on the network is defined by some secret state (e.g., private keys), and after a failure, this state might be deleted, corrupted, or known to the adversary. Consequently, the recovered server needs new secret state and in essence, a new identity, and we may as well consider it to be a new server.

Ordinary PSS schemes ([25], [19], [9]) generate new shares of the shared secret, but they assume that other secret state associated with each server is never compromised, so the identities of the shareholders are fixed over the lifetime of the system.

A better approach is to allow shareholders to be replaced with different nodes. This provides a reasonable methodology for a system administrator: identify a set of "fresh" nodes, e.g., recently added nodes or newly recovered nodes with new keys, and direct the secret shares to be moved to them. However, PSS schemes do not permit the replacement of one shareholder with another. Furthermore, a difficulty in defining such schemes is that a naive transfer of shares from an old group of servers to a new group may reveal the secret if more than t faulty servers exist between the old group and the new group.

An additional point is that PSS schemes do not allow the threshold t to change, which may be desirable in a long-lived system. The threshold has a meaning: it represents an assumption about how easily nodes can become corrupted. If current events dictate a reevaluation of this assumption, it would be better to change to a new threshold rather than start over.

These schemes allow shareholders to determine whether the dealer sent them valid shares of the secret, hence allowing them to come to a consensus regarding whether the secret was shared successfully. In this context, the dealer is semi-trusted; it does not reveal the secret, but it might attempt to fool servers into accepting an invalid sharing of the secret. Verifiable secret sharing is an important component in many distributed secret sharing protocols involving un-trusted participants because the protocols typically involve each server acting as a semi-trusted dealer to all of the others.

Feldman's [17] and Pedersen's [26] schemes have similar efficiency but slightly different security guarantees. Feldman's [17] scheme is perfectly binding (i.e., an un-trusted dealer cannot fool shareholders into accepting an invalid sharing) and computationally binding (meaning that secrecy is subject to computational hardness assumptions and the amount of computation available to the shareholders). Pedersen's [26] scheme, on the other hand, is computationally binding and perfectly hiding. We focus on Feldman's [17] scheme because it is notationally easier to describe than Pedersen's [26] scheme; however, the systems we describe could be adapted to either scheme.

1.2 Contributions

In secret sharing schemes, a secret *s* is divided into *n* shares by a dealer *D* and is distributed among *n* shareholders. In a long lived system, over a period of time there is chance that the adversary may attack the system by gaining the knowledge of at least *t* shares to recover the secret. Additionally, systems may fail to function properly, for instance due to hardware failure or an attack.

Proactive secret sharing (PSS) schemes, address the problem that shares can be exposed or lost over time due to Byzantine faults. In PSS, servers execute a share regeneration (renewal) protocol, in which a new set of shares of the same secret is generated and the old shares are discarded, rendering useless any collection of *t* or fewer shares the adversary may have learned. Furthermore, PSS schemes typically provide a share recovery protocol so that a full set of new shares can be generated even if some of the old shares (up to some maximum number of faults tolerated) have been lost.

All the existing Proactive secret sharing schemes assume that an adversary can attack only the shares but not the master secret. Therefore, shares are renewed at the beginning of each time period. In our paper, we considered a complete different approach of Proactive secret sharing in which the master secret is to be renewed frequently but not the private shares.

This thesis makes the following major contributions:

- In the secret sharing scheme, the master secret *s* is determined by a mutually trusted dealer *D* and is stored in a centralized server. The dealer divides the secret into *n* shares and distributes it among *n* shareholders and the shares are stored separately in *n* distributed servers. In general PSS, it is believed that the adversary can attack only the shares but not the master secret. So the shares are renewed frequently at the beginning of

each time period. But we followed a completely different approach in which the master secret is to be renewed frequently without disturbing the private shares.

- If once the shareholders are able to reconstruct the master secret, then the secret is no longer secure. So if all the shares are kept secure during the secret reconstruction, then the dealer needs to renew only the master secret. Using this proposal, PSS becomes even more efficient since the shares can be reused for long period of time.
- The shares are distributed to the shareholders using a secure channel. We also considered the situation where a mobile adversary can successfully comprise some shareholders to know about the secret.
- Without changing the private shares, the dealer can change the threshold of the master secret which provides an efficient way to change the threshold dynamically.
- In most proactive secret sharing schemes, the access privilege of shares is controlled by the dealer. Whenever old shares are compromised, the dealer need to revoke the compromised shares. Our approach allows dealer to add/revoke shares without re-distributing remaining shares. However, some changes to the master secret is to be performed.

Our approach make the secret sharing even more practical and flexible. The private shares remain unchanged during the renewal of master secret and can be reused to construct several master secrets. One common feature in all of our proposed protocols is, to make changes only to the master secret without making any changes to the private shares. This unique feature simplifies the implementation of PSS since making changes to the master secret effects data stored in central server only. Our proposed scheme reduces the overhead in restoring all the renewed (new) shares once an update/renewal protocol is triggered.

1.3 Thesis Outline

The Thesis is organized as follows. In the next chapter, we discussed about the work related to secret sharing and proactive secret sharing; the first proposed Shamir's secret sharing scheme, Ostrovsky and Yung who introduced the concept of proactive security, first proposed proactive secret sharing scheme by Herzberg et al. and Cachin et al.'s asynchronous proactive secret sharing scheme. We discussed the protocols for share generation, secret reconstruction and share renewal (proactive secret sharing) processes. In the 3rd chapter, we mentioned our approach of proactive secret sharing; public and private share generation and authentication protocols. Then we discussed about master secret renewal protocol, strong verifiable secret sharing protocols which can be applied for the proposed scheme. Later in this chapter, we discussed about shares adding or revoking and changing the threshold protocols. We concluded the 4th chapter by discussing the open problems in our proposed scheme and future work.

CHAPTER 2

RELATED WORK

2.1 Basic Schemes

In this chapter, we briefly discussed share generation and secret reconstruction protocols of Shamir's secret sharing scheme. And in the later part of this chapter, we discussed about proactive secret sharing which addresses the problem that given sufficient time the adversary can gain information about the secret in long-lived systems. We discussed about proactive security which was first mentioned by Ostrovsky and Yung and then we discussed about the first Proactive Secret Sharing Scheme which was proposed by Herzberg and the first asynchronous model of proactive secret sharing scheme by Cachin et al.

Secret sharing and its many variations form an important primitive in cryptography. It finds extensive use in key management, distributed storage system etc. The basic model for secret sharing distinguishes at least two protocols: (i) a *share generation protocol* in which the secret s is divided into n sub-shares and distributed by a dealer among n participants, and (ii) a *secret reconstruction protocol* in which the secret is recovered by pooling the shares of a qualified subset of the participants (at least t shares are required to reconstruct the secret s). Basic schemes ([29], [6] for threshold secret sharing) solve the problem for the case that all players in the scheme are honest.

Secret sharing has been classified based on the following types :

(1). *Cryptographic* (the secrecy of the secret depends on the difficulty of solving certain number theoretic hard problem) or *Information theoretic* ([29], [6]) (the secrecy of the secret is not dependent on the hardness of any computational problem).

(2). *Threshold* ([29], [6]) (for a fixed threshold t, any set of $t + 1$ parties can uniquely reconstruct

the secret i.e. access structure is the set of all different combinations of $t + 1$ parties) or *Non-threshold* [3] (generalization of threshold; Access structure may have sets of parties of different size).

(3). *Static* ([29], [6], [3]) (the shares of secret remain the same after the distribution) or *Proactive/Mobile* ([10], [19], [25]) (the shares can be refreshed or redistributed without changing the secret in order to maintain secrecy over long periods).

In this paper, we focused on Proactive/Mobile type of secret sharing. In a Secret sharing scheme, a dealer divides a secret into multiple shares and distributes it among shareholders in such a way that any authorized subset of shareholders can reconstruct the secret; but any unauthorized subset of shareholders gain no information about the time.

Shamir's original scheme doesn't prevent malicious behavior of dishonest shareholders during secret reconstruction and also the secret is stored for a long period of time. We can overcome these limitations using Proactive Secret Sharing (PSS).

Proactive Secret Sharing (PSS) scheme address the problem that even if the shares can be exposed or lost over certain period of time due to byzantine faults, new set of shares can be generated for the same master secret. Using Share regeneration/renewal protocol in PSS, the servers can obtain new shares and the old shares can be discarded, rendering useless information to the shareholder by the collection of t or fewer than t shares from the shareholders. Even if some of the old shares might have been lost, Share recovery protocol helps in the generation of a full set of new shares.

Proactive security was first suggested by Ostrovsky and Yung [25] and this concept was applied to the secret sharing schemes by Herzberg et al. [19]. Basically the idea is that, if the information stored by the servers in order to share a given secret stays the same for all lifetime of

the system, then an adversary can eventually break into a sufficient number of servers, to learn and destroy the secret. On the other hand, let the time is divided into periods. At the beginning of each period the information stored by the servers in a given time period changes, while the shared secret stays the same. Then the adversary probably does not have enough time to break into necessary number of servers. Moreover, the information he learns during the period t is useless during the period $t + i$, for $i = 1, 2, ...$ So, he has to start a new attack from scratch during each time period.

Proactive security refers to security and availability in the presence of a so-called mobile adversary. Herzberg et al. [19] have further specialized this notion to robust secret sharing schemes and have given a detailed efficient proactive secret sharing scheme. Consider the following problem: if the information stored by the players in order to share a given secret stays the same for a long period of time (e.g. the lifetime of the system), then an adversary may gradually break into a sufficient number of players, to learn and destroy the secret. A way to address this problem is to divide the time into periods. At the beginning of each period the information stored by the players in that period changes, while the shared secret stays the same. The system is set up in such a way that the adversary does not have enough time to break into a required set of players. Moreover, the information that the adversary learns during a particular period is useless during later periods. So, the adversary has less change to break into the system and learn the secret.

Proactive security provides enhanced protection to long-lived secrets against a mobile adversary, i.e., the adversary which is allowed to potentially move among players over time with the limitation that it can only control some subset of players at a time unit. In fact, proactive security adds protection by “time diffusion”. Namely, all shares are periodically refreshed. This

renders useless the knowledge obtained by the mobile adversary in the past. Proactive systems also use robustness techniques to enhance availability by tolerating (detecting and correcting) malicious players. Moreover, it also allows recoveries of the previously corrupt players, by "removing" the adversary influence and restoring their (correct) information. This gives the system a self-healing nature. As a result, the system can tolerate a mobile adversary.

2.1.1 Shamir's Secret Sharing Schemes

A secret sharing scheme divides a secret *s* into *n* shares by a dealer *D* and distributes them among *n* shareholders $P = \{P_1, P_2 \ldots\ldots, P_n\}$ in such a way that at least *t* shares are required to reconstruct the secret and less than *t* shares gain no information about the secret. The (*t, n*) threshold secret sharing schemes were introduced by Shamir [29] and Blakley [6] independently in 1979. A (*t, n*) threshold secret sharing scheme allows any *t* or more than *t* shareholders to reconstruct the secret; while fewer than *t* shareholders can gain no information about the secret.

This technique enables the construction of robust key management schemes for cryptographic systems that can function securely and reliably even when misfortunes destroy half the pieces and security breaches expose all but one of the remaining pieces [29].

In [22], Liu considers the following problem:

Eleven scientists are working on a secret project. They wish to lock up the documents in a cabinet so that the cabinet can be opened if and only if six or more of the scientists are present. What is the smallest number of locks needed? What is the smallest number of keys to the locks each scientist must carry?

It is not hard to show that the minimal solution uses 462 locks and 252 keys per scientist. These numbers are clearly impractical, and they become exponentially worse when the number of scientists increases. In this paper we generalize the problem to one in which the secret is some data *s*. Our goal is to divide *s* into *n* shares $p_1, \ldots\ldots p_n$ in such a way that:

(1). knowledge of any t or more shares makes the secret s easily computable;

(2). knowledge of any $t-1$ or fewer shares leaves the secret s completely undetermined (in the sense that all its possible values are equally likely).

Such a scheme is called a (t, n) threshold scheme.

Efficient threshold schemes can be very helpful in the management of cryptographic keys. In order to protect data we can encrypt it, but in order to protect the encryption key we need a different method (further encryptions change the problem rather than solve it). The most secure key management scheme keeps the key in a single, well-guarded location (a computer, a human brain, or a safe). This scheme is highly unreliable since a single misfortune (a computer breakdown, sudden death, or sabotage) can make the information inaccessible. An obvious solution is to store multiple copies of the key at different locations, but this increases the danger of security breaches (computer penetration, betrayal, or human errors).

By using a (t, n) threshold scheme with $n = 2t - 1$ we get a very robust key management scheme: We can recover the original key even when $[n/2] = t - 1$ of the n pieces are destroyed, but our opponents cannot reconstruct the key even when security breaches expose $[n/2] = t - 1$ of the remaining t pieces.

In other applications the tradeoff is not between secrecy and reliability, but between safety and convenience of use. Consider, for example, a company that digitally signs all its checks. If each executive is given a copy of the company's secret signature key, the system is convenient but easy to misuse. If the cooperation of all the company's executives is necessary in order to sign each check, the system is safe but inconvenient. The standard solution requires at least three signatures per check, and it is easy to implement with a $(3, n)$ threshold scheme. Each executive is given a small magnetic card with one pi share, and the company's signature

generating device accepts any three of them in order to generate (and later destroy) a temporary copy of the actual signature key *s*. The device does not contain any secret information and thus it need not be protected against inspection. An unfaithful executive must have at least two accomplices in order to forge the company's signature in this scheme.

Some of the useful properties of this *(t, n)* threshold scheme are:

(1). The size of each piece does not exceed the size of the original data.

(2). When *t* is kept fixed, shares can be dynamically added or deleted without affecting the other shares.

(3). It is easy to change the shares without changing the original secret *s*; all we need is a new polynomial *f(x)* with the same free term. A frequent change of this type can greatly enhance security since the shares exposed by security breaches cannot be accumulated unless all of them are values of the same edition of the *f(x)* polynomial.

(4). By using tuples of polynomial values as shares, we can get a hierarchical scheme in which the number of shares needed to determine *s* depends on their importance. For example, if we give the company's president three values of *f(x),* each vice-president two values of *f(x),* and each executive one value of *f(x),* then a (*3, n*) threshold scheme enables checks to be signed either by any three executives, or by any two executives one of whom is a vice-president, or by the president alone.

Consider a system comprising a set of *n servers* that hold shares of a secret and communicate through a network. At any time, a server is either *correct* or *compromised*. A compromised server might stop executing, deviate arbitrarily from its specified protocols (i.e., Byzantine failure), and/or disclose or change information stored locally. A compromised server can be *recovered* and become correct after the following actions are taken.

- Reset hardware and system configurations
- Reload the code (thereby eliminating Trojan horses)
- Reconstitute the state of each server (which might have been corrupted)
- Obsolete any confidential information an attacker might have obtained from compromised servers.

Given a time interval T , a server is considered *correct during T* if and only if that server is correct throughout interval T ; otherwise, the server is deemed *compromised during T* .

In Shamir's (*t, n*) threshold scheme, Share generation is based on $(t\text{-}1)^{th}$ degree polynomial and secret reconstruction is based on Lagrange interpolating polynomial of at least *t* private shares.

Share Generation Protocol:

Dealer *D* divides the secret *s* among *n* shareholders such that atleast *t* shares are required to reconstruct the secret. The share generation protocol is discussed in Figure 2.1.

The share generation protocol is as follows:

(i). Dealer creates a random polynomial $f(x)$ of degree $(t-1)$ and a constant term a_0 where a_0 is the secret s in a finite field(which is known to all the shareholders and the dealer as well).

$$f(x) = a_0 + a_1x + a_2x^2 + \ldots\ldots\ldots + a_{t-1}x^{t-1}$$

where $a1,a2,\ldots\ldots a_{t-1}$ are random polynomials.

(ii). Dealer randomly selects n distinct points ($x_i \neq 0$), calculates each share value and distributes the share to each shareholder secretly. ($share_i(s) = (x_i, f(x_i))$, $i = 1,2\ldots\ldots,n$).

x_i value is a known value. So for our convenience, we chose $x_i = i$.

Therefore, $s_1 = f(1)$, $s_2 = f(2)\ldots\ldots$, $s_n = f(n)$ and the secret $s = f(0)$.

Figure 2.1 Share Generation Protocol

Secret Reconstruction Protocol:

Any subset of t or more shares can be used to reconstruct the secret. Without loss of generosity, the subset is : $f(1), f(2)....f(t)$.

(i). Lagrange interpolating formula is used to find the polynomial $f(x)$, such that degree of $f(x)<t$ and $f(i) = share_i(s)$ for $i = 1,2,.....,n$.

(ii).The reconstructed secret must be $f(0)$.

Given any t pairs of $(i, f(i))$, with distinct i values, there is a unique polynomial $f(x)$ of degree $t-1$, passing through all these points. This polynomial can be effectively computed from the pairs $(i, f(i))$.

Figure 2.2 Secret Reconstruction Protocol

Lagrange's interpolating formula

The polynomial $f(x)$ can be calculated using Lagrange's interpolating formula as stated in Table 2.1.

Table 2.1 Lagrange's interpolating formula

$L_i(x)$	$F(x)$
$L_i(x) = \dfrac{\prod_{j \neq i} (x - x_j)}{\prod_{j \neq i} (x_i - x_j)}$	$f(x) = \sum_{i=1}^{t} f(i) * L_i(x)$ where $L_i(x)$ is the Lagrange interpolating polynomial.

Shamir's secret sharing scheme is information theoretic secure and it satisfies the basic requirements of secret sharing scheme. This can be verified as:

(1). With the knowledge of any subset of *t* or more shares, it is possible to reconstruct the secret.

(2). By knowing less than *t* shares, the secret cannot be predicted.

During secret reconstruction phase, shareholders may present fake shares and thus the other honest shareholders get nothing but a faked secret. For the fair reconstruction of the secret, cheater detection and/or identification are very essential. However, Shamir's original secret sharing scheme doesn't prevent malicious behavior of dishonest shareholders.

There are several papers on cheater detection and/or identification for secret sharing schemes ([1],[7]) which consider exactly *t* shares during secret reconstruction. Some other papers ([5],[23]) proposed secret sharing scheme based on an error-correcting code in which faked shares can be treated as error codes to be detected and corrected based on coding technique.

There are other papers in which more than *t* shares are used during secret reconstruction phase, and the redundant shares can be used for cheater detection and/or identification ([18], [30]).

2.1.2 Proactive Secret Sharing Scheme

Proactive security provides a method for maintaining the overall security of a system, even when individual components are repeatedly broken into and controlled by an attacker. In particular it provides for automated recovery of the security of individual components, avoiding the use of expensive and inconvenient manual processes. Ostrovsky and Yung [25] introduced the concept of proactive security and showed how a large class of multi-party protocol problems can be solved in a proactive way, in a setting where secure communication channels are available.

Proactive security for secret sharing was first suggested by Herzberg et al. [19] where they presented, among other things, a proactive polynomial secret sharing scheme. The proposed proactive polynomial secret sharing scheme in [19] uses the verifiable secret sharing scheme of [27]. Proactive security refers to security and availability in the presence of a mobile adversary. Herzberg et al. [19] further specialized this notion to robust secret sharing schemes and gave a detailed efficient proactive secret sharing scheme. “Robust” means that in any time period, the shareholders can reconstruct the secret value correctly.

The secret value needs to be maintained for a long period of time. The life time is divided into time periods which are determined by the global clock. At the beginning of each time period the servers engage in an interactive update protocol. The update protocol will not reveal the value of the secret. At the end of the period the servers hold new shares of the secret. The update/ share renewal protocol relies on verifiable secret sharing (VSS) which allows shareholders who have renewed their shares to verify that all renewed shares are consistent. It is a common

expectation that once we have the concept of proactivity very often it is quite easy to add it on top of an existing distributed protocol as VSS, not withstanding many known VSS are not easy to adapt for proactive property.

Secret sharing alone does not defend against *mobile adversaries* which attack, compromise, and control one server for a limited period before moving to another. Given enough time, a mobile adversary might compromise t servers, obtain t shares, and thus learn the secret. The defense here is *share refreshing*, whereby servers periodically create a new, independent secret sharing and then replace old shares with new ones. Because the new and old shares are independent, a mobile adversary cannot combine new shares with old shares in order to reconstruct the secret.

Secret sharing with share refreshing is known as *proactive secret sharing* (PSS). PSS reduces the *window of vulnerability* during which an adversary might compromise more than t servers in order to learn the secret. Without share refreshing, the window of vulnerability is unbounded; with PSS, the window of vulnerability is shortened to the period between two consecutive executions of share refreshing.

Prior work on PSS protocols assumes a *synchronous system*, which implies bounds are known for message delivery delays and processor execution speeds. Any assumption constitutes a vulnerability, and the assumption of a synchronous system is no exception. Denial-of-service (DoS) attacks, in particular, might delay messages and/or consume processor cycles, thereby invalidating the defining assumptions for a synchronous system.

This paper describes APSS, a PSS protocol for *asynchronous systems* in which message delivery delays and processor execution speeds do not have fixed bounds. We are eliminating an assumption and thus eliminate vulnerability. Besides implementing secret sharing, APSS can be

used for *threshold cryptography*, where servers store shares of a private key and perform cryptographic operations using these shares (without ever materializing the entire private key).

Asynchronous Proactive Secret Sharing was first described in Zhou [32] and Zhou et al. [33]. This paper [33] restructures the original protocol, presents extensions, and provides a more rigorous proof than the one given in Zhou [32]. A second proactive secret sharing protocol for asynchronous systems, developed independently, is described in Cachin et al. [9], where a formal model and proof are presented. This protocol differs from APSS in two significant ways:

(1). It employs a randomized multi-valued validated Byzantine agreement protocol Cachin et al. [8] so that all correct servers will agree on the set of sub-sharings to use in generating a new sharing for the secret. APSS eschews the agreement and instead generates multiple new sharings, each with a different label. There is no need to generate only one new sharing, so APSS avoids the need to run an agreement protocol.

(2). It employs a bivariate polynomial to implement subsharing recovery. The scheme, which can be retrofitted into APSS, circumvents the exponential explosion that the (l, l) secret sharing causes for APSS; the protocol of Cachin et al. [9] has polynomial-time communication and message complexity.

However, the solution in Cachin et al. [9] does not apply to refreshing shares of an RSA private key because of its use of bivariate polynomial—for RSA, $\varphi(n)$, the modulus, is not known to servers. In contrast, APSS can be easily adapted: because of its use of (l, l) secret sharing, APSS can use the (n, n) threshold RSA scheme and share-refreshing scheme outlined in Rabin [28]—these schemes do not require operations with modulus $\varphi(n)$.

APSS Correctness Requirements:

The objective of APSS is to provide a protocol for share refreshing so that the following properties hold.

APSS Secrecy: An adversary learns nothing about secret *s*.

APSS Integrity: At any time, with high probability, secret reconstruction returns *s* when it terminates.

APSS Availability: If messages sent during an execution of secret reconstruction are delivered before a subsequent execution of share refreshing starts, then that execution of secret reconstruction terminates.

APSS Progress: Execution of share refreshing terminates on all servers that are correct during that execution of share refreshing; at termination, correct servers have deleted old shares and any related information.

These correctness requirements must hold in a given assumed system model and adversary model.

2.1.2.1 Ostrovsky and Yung

As a way to cope up with network worms and viruses, the concept of proactive secret sharing was first introduced by Ostrovsky and Yung [25]. If an adversary infects a shareholder at a constant rate, shareholders are rebooted and restored to the current state at an equal rate. Hence, they assume that in any given time period, $t < [n / 2]$ shareholders may be faulty. (Note that this threshold is better than the $t < [n / 3]$ typically required for asynchronous schemes, and is possible only because the correctness of their protocol is based on the unrealistic synchrony assumption that servers that fail to respond within some fixed amount of time are faulty.) Shareholders protect their shares by triggering a renewal protocol to refresh their shares and generate new shares from the old shares (discards their old shares and continue to use the new shares in place of old shares). In [25], the update protocol is implemented via a generic secure multi-party computation protocol on the existing shares. These multi-party protocols ([4], [11], [27]) are general but inefficient. Some are implemented in terms of many instances of verifiable secret sharing. This seminal work is important because it was the first to demonstrate that proactive secret sharing is theoretically possible; however, the Ostrovsky and Yung scheme is infeasible in practice because performing nontrivial calculations using generic secure multi-party protocols is expensive. Furthermore, Ostrovsky and Yung assume that the network is synchronous, and that secure channels are uncompromised by past corruptions of the endpoints. Practical implementations of secure channels involve secret keys that would be exposed by a compromise of the endpoints, and hence it is unclear how to recover the node in that case, since the adversary now knows the node's secret keys. Although they show that "recovery" of a machine's state is possible in theory by having all of the other participants construct it via a

secure multi-party computation, it is unclear how one might perform recovery efficiently in practice.

2.1.2.2 Herzberg et al.

An effective way to protect the secret from adversary attacks is to periodically update the shares. Herzberg et al. [19] in 1995 developed Proactive Secret Sharing Scheme to the existing Shamir's Secret Sharing Scheme. Their paper focuses mainly on constructing semantically secure and robust (at any time honest servers/shareholders should have correct shares and this correctness can be verified by other servers) secret sharing scheme. All these honest shareholders must co-operate with the other shareholders who do not have correct shares and help them in recovering their lost shares. The recovered shares must be kept secret except to the intended shareholder.

Herzberg used the basic share generation and secret reconstruction protocols of original Shamir's secret sharing scheme. After initialization of Shamir's SSS, at the beginning of each time period, all the honest servers (shareholders) can trigger an update phase during which all the servers perform a share renewal protocol. During the share renewal process, all the shareholders (servers) should have access to a common global clock so that there will be process synchronization between all the servers and also all the servers can apply the renewal protocol at a certain period of time. Time is divided into multiple slots which is determined by the common global clock. Each server is connected to a communication broadcasting medium (communication channel) such that all the messages sent though this channel reaches every party connected to it. At the beginning of each time slot, servers trigger update protocol and at the end of the timeslot new shares will be generated.

Servers store shares that constitute a (*t, n*) secret sharing for *s*. The initial shares are generated by a trusted party and are assumed to have been delivered securely to all participating servers. Thereafter, servers replace old shares with new shares by periodically invoking share refreshing. *Secret reconstruction* is used to reassemble a secret from the shares when needed.

Complications arise when executions of share refreshing is concurrent with secret reconstruction, because share refreshing might delete (old) shares that secret reconstruction needs. This is an instance of the well-known readers/writers problem [13], with secret reconstruction the reader and share refreshing the writer. One solution, in the spirit of Lamport's [20] concurrent reading while writing, is to use version numbers in conjunction with verifiable secret sharing; execution of secret reconstruction detects whether shares it reads constitute a sharing of *s* and iterates if they do not.

A secret sharing scheme is quite vulnerable when an adversary breaks into a system before the lifetime of the secret expires. Ben-Or et al. [4] in 1988 discussed on distributed fault tolerance systems and described some of the possible solutions to avoid these attacks. Among several different adversary attacks, some of the frequently prone attacks are:

- Passive adversary attacks, and
- Active adversary attacks

Passive adversary attacks results in spoofing the data without actually modifying or correcting the data. In active adversary attacks, much in contrast with passive adversary attacks, the adversaries penetrate into the system, infiltrate the system, damage and/or destroy the data already existing in the system. Herzberg proposed PSS based on Shamir's SSS to prevent these attacks.

An adversary can corrupt the shares at any time period. If an adversary corrupts the shares during an update phase, then it is believed that they are corrupted during both the adjacent time periods of the update phase. Herzberg assumed that at any time period, an adversary can attack at most *k* shares (where k less than n/2) which guarantees the existence of *k+1* shareholders at each time period. Servers can be corrupted in many ways by an adversary like learning any information in the server, modifying the data, disconnecting it, changing the intended behavior of the server etc.

One of the important requirements of Proactive secret sharing scheme is an authenticated broadcast channel and secure communication channels between the shareholders. The shareholders renew their shares without the involvement of the dealer. At the beginning of each time period and after initialization, when all the shareholders (servers) trigger an update phase, all the servers should perform the share renewal protocol. After triggering the share renewal protocol, each shareholder will obtain a new share on the new *t-1* polynomial. The shareholders should agree on the new polynomial with same secret *s* without revealing the secret.

Assumption :

A secret *s* is divided into *n* shares and is distributed among *n* shareholders. For some *t*-1 degree polynomial *f(x)* each shareholder *(i)* holds a share *f(i)* after initialization.

Share Renewal protocol

(i). Each shareholder *(i)* where $i \varepsilon [1,n]$ randomly picks *t-1* numbers from the finite field. These numbers define a polynomial *p(x)* of degree *t-1*.

(ii). Each shareholder*(i)* distributes the shares $p_i(x)$ among the shareholders using Verifiable Secret Sharing Scheme.

(iii). Each shareholder*(i)* receives all the shares $p_1(i), p_2(i) \ldots\ldots p_n(i)$ and computes the new shares by adding the sum of all new shares to his old share. The new share is computed as follows:

$$h(i) = f(i) + \sum_{k=1}^{k=n} p_k(i)$$

where h(i) is the new share of i^{th} *shareholder ;*

f(i) is the old share

Figure 2.3 Share Renewal Protocol

In this way new shares can be computed; and after generation of new shares, the shareholders can discard their corresponding old shares.

There are protocols for share renewal protocols in the presence of active attackers, detection of corrupted shares, recovery of lost or corrupted shares which are defined in the original paper of Herzberg [19].

2.1.2.3 Cachin et al.'s Asynchronous Scheme

The protocol of Cachin, Kursawe, Lysyanskaya, and Strobl [9] is the first efficient scheme in the asynchronous model, also for t < [n/3]. Whereas the Herzberg et.al scheme [19] computes each new share *P'(i)* as a function of a corresponding old share *P(i)*, the Cachin scheme is based on resharing the shares of the secret and combining the resulting sub-shares to form new shares of the secret. Their paper first presents a protocol for asynchronous verifiable secret sharing, then shows how to build an asynchronous proactive secret sharing scheme by having each honest shareholder create a VSS of its share. Their VSS scheme is similar to the one of Stinson and Wei [31], and the method of computing new shares from subshares is based on the linearity of Lagrange interpolation, which was proposed by Desmedt and Jajodia in [16]; however, the authors seem to be unaware of either of these earlier works.

To handle share recovery for participants who have lost or never received their shares, Cachin et al. use a two-dimensional sharing in which shares are one-dimensional projections *P(i, y)* and *P(x, i)*; thus, any participant can interpolate its share given the points of overlap with at least *t+ 1* other shares. Cachin et al.'s protocol requires that a significant amount of information be broadcast by each participant to each other participant even in the absence of faults, whereas there are other schemes [19] achieves better efficiency in the common case by using a coordinator. Moreover, their protocol does not support changing the set of shareholders.

CHAPTER 3

OUR SCHEME

For some secret sharing applications, the secret reconstructed is not revealed to the shareholders, therefore the secret/shares can be repeatedly used without having to be changed. But for other applications, in which the reconstructed secret is revealed to the shareholders, a new secret must be chosen and its corresponding shares must be regenerated and then secretly distributed to shareholders again. This is inefficient because of the overhead in the generation and distribution of shares. In [18], Harn et al. proposed a *l*-span secret sharing scheme for the general sharing policy to solve the share regeneration problem by extending the life span of the shares from *1* to *l* i.e., the shares can be repeatedly used for *l* number of times to generate *l* different secrets.

This allows secrets to be shared in a more efficient way in which same set of shares can be used to reconstruct *l* different secrets. Even though this protocol allows the shares to be used for a longer period of time, the protection provided is insufficient as the secret is stored for longer period of time. Given sufficient time, an adversary might attack the system and gain sufficient knowledge about the secret. So to avoid this, we proposed a new Proactive secret sharing scheme, in which the secret is refreshed periodically at the beginning of each time slot without changing the private shares. In this way, private shares can be reused to generated different master secrets.

In secret sharing scheme, the master secret and all the private shares (which are distributed by the dealer to the shareholders) are the two secrets which are to be maintained confidentially. In all the secret sharing schemes proposed till date, private shares are reused to reconstruct the master secret. But in our approach, instead of renewing the private shares

frequently at the beginning of each timeslot; during the share renewal process, each time master secret is renewed. In this way private shares can be reused for a longer period of time and also private shares can be reused to construct different master secrets.

Certificate Authority (CA) is a server which issues digital certificates to be used by other parties. The digital certificate contains a public key and identity of the owner of the certificate. CA is very important in Public Key Infrastructure (PKI) schemes. In such key management applications our scheme can be applied. CA has a private key for generating digital signatures.

Instead of storing the entire key secretly, it is better to divide the key into number of shares and share the key. This private key can be shared as a secret; and is shared in a secret sharing scheme. Once if all the private shares are able to recover the master secret, then the secret is no longer secure. So it is necessary to renew the master secret keeping all the private shares secure. If all the private shares are kept secure during secret reconstruction, dealer only needs to renew master secret but not the private shares. So in this way, the secret sharing scheme becomes even more efficient as the private shares can be reused for a longer period of time.

3.1 Motivation

In a secret sharing scheme, a mutually trusted dealer determines the master secret and stores it in a centralized server. The dealer generates private shares and distributes them among *n* shareholders which are stored separately in *n* distributed servers. In all the existing Proactive Secret Sharing Schemes, it is believed that a mobile adversary can attack only the private shares but not the master secret, so at the beginning of each time period, new shares are generated leaving discarded all the old shares. In this paper, we followed a completely different approach where the master secret is renewed at the beginning of each time period. Once if all the private shares are able to recover the master secret, then the secret is no longer secure. So it is necessary to renew the master secret keeping all the private shares secure. If all the private shares are kept secure during secret reconstruction, dealer only needs to renew master secret but not the private shares. So in this way, the secret sharing scheme becomes even more efficient as the private shares can be reused for a longer period of time. It is believed that the dealer is a trust worthy dealer and he divides the shares and distributes them to all the shareholders using a secured channel. When some of the old shares are compromised by an adversary, then without redistributing the remaining shares, the dealer (who has complete access privileges on the private shares) can revoke the compromised shares. Shares adding/revoking protocol for the proposed proactive secret sharing is mentioned in details in 3.6. However some changes are to be made to the master secret. An efficient way to change the threshold dynamically is mentioned in 3.7. In the proposed proactive secret sharing scheme, without redistributing the distributed private shares, the dealer can change the threshold of the master secret.

3.2 Contribution

In the original Shamir's secret sharing scheme, the dealer is believed to be trust worthy and hence it is believed that he distributes valid shares to the shareholders. In Shamir's secret sharing scheme, a dishonest dealer might distribute inconsistent shares to the shareholders thus preventing them from recovering the secret. So, there is a need to check the consistency of the dealer. Feldam [17] and Pederson[26] introduced the concept of Verifiable secret sharing scheme to the original secret sharing scheme using which the shareholders can verify whether the dealer has sent them valid shares or not. Here, the dealer is not fully trusted; he may not reveal the secret but he can try attempting to share an invalid secret. In our proposal, we introduced a new notion of verifiable proactive secret sharing scheme which is called Strong Verifiable Proactive Secret Sharing Scheme (SVPSS). SVPSS ensures that all shares are strong t-consistent (if in a set of n shares, any subset of t or more than t shares can recover the secret).

In a *(t , n)*-SS scheme, let $m \geq t$, a set of m shares $s_1, s_2, \ldots, s_m$ are said to be consistent if any subset containing t shares of the set reconstructs the same secret. There are seven protocols which are defined under SVPSS. They are : *the private shares generation protocol(3.3.2) , the public shares authentication protocol(3.3.4), the master secret generation/renewal protocol(3.4.2), the master secret reconstruction protocol(3.4.4), the strong verifiable secret sharing protocol(3.5.2), the shares addition/revoking protocol(3.6.1) and the changing threshold protocol(3.7.1)*. All the above mentioned protocols are unconditionally secure. As the change is made only to the master secret, this unique feature simplifies the PSS as the change is to be made only to the central server (the master secret) without effecting all the private shares. Our proposed scheme reduces the overhead in restoring all the renewed shares once an update/renewal protocol is triggered.

3.3 Share Generation and Authentication

3.3.1 Private Share Generation

Dealer divides the secret *s* into *n* shares and divides it among *n* shareholders in such a way that at least *t* (where *t* is the threshold value and the threshold value can be altered) shares are required to reconstruct the secret. The protocol for private share generation is designed as follows:

3.3.2 Private Share Generation Protocol

(i). Dealer selects a prime *r* where $r > s$; *s* being the secret. Dealer select a random private share $y_i^{\,u} \varepsilon z_r$, for each shareholder U_i and sends the private share secretly to the dealer (using a secure authenticated channel).

(ii). Dealer selects a random polynomial $f^1(x)$ and makes this polynomial publicly known.

(iii).Each shareholder computes his own private share as $y_i = y_i^{\,u} + f^1(x_i)$

The polynomial $f^1(x)$ is used for VSS.

Figure 3.1 Private Share Generation Protocol

3.3.3 Public Shares Authentication

After renewing each master secret on the centralized server, dealer needs to broadcast public shares of the renewed master secret to all shareholders. All shareholders can use their private shares to work together to authenticate the public shares of the master secret.

3.3.4 Public Shares Authentication Protocol

All shareholders follow the secret reconstruction protocol to invoke their private shares and public shares, y_v'', for $v = 1,2,...,n-t+1,$ to recover the constant term of Lagrange interpolation polynomial and compare this recovered value with the computed one-way hash value of $b = h(y_1' \| y_2' \| ... \| y_{n-t+1}')$. If these two values are identical, the public shares, y_v', for $v = 1,2,...,n-t+1,$ of the master secret *s* has been authenticated.

Figure 3.2 Public Shares Authentication Protocol

Theorem The protocol can allow shareholders to authenticate the public shares, y_v', for $v = 1,2,...,n-t+1,$ of the master secret.

Proof In the master secret generation protocol, only dealer can construct a polynomial with degree n exactly such that this polynomial passes through n private shares of shareholders and the specified constant term. Thus, after verifying the one-way hash value b generated by the dealer, all shareholders can be convinced that the input values of this one-way hash output is also generated by the dealer.

3.4 Master Secret Generation/Renewal and Reconstruction

3.4.1 Master Secret Generation/Renewal

Dealer need to construct a secret sharing scheme for the given master secret s in such a way that any t or more than t shares can be used to reconstruct the secret, but less than t shares gain no information about the secret. In this protocol, instead of renewing private shares periodically in a proactive secret sharing scheme, one can use the same private shares to reconstruct different master secrets. Thus, dealer needs to publish some public shares of each master secret. This approach simplifies the implementation of PSS since secure channel for distributing private share to each shareholder is only needed during initialization. By doing so, the efficiency of PSS can be improved as shares can be reused for long period of time.

3.4.2 Master Secret Generation/Renewal Protocol

(i). Let the threshold of the new master secret is *t*. Dealer constructs a polynomial $f(x)$ with degree *n* such that $f(x_i) = y_i,$ for $i = 1,2,...,n,$ and $f(0) = y_0 = s,$ where *s* is the master secret, $x_0=0$ and x_i is a publicly known parameter associated with each shareholder U_i .

(ii). Dealer computes $f(x_v^{'}) = y_v^{'},$ where $x_v^{'} \notin x_i,$ for $v = 1,2,...,n-t+1.$ Public shares $f(x_v^{'}) = y_v^{'},$ for $v = 1,2,...,n-t+1,$ of the master secret *s* are made publicly known.

(iii).Dealer computes the one-way hash value of all public shares as $b = h(y_1^{'} \| y_2^{'} \| ... \| y_{n-t+1}^{'}).$ Dealer repeats Step 2 to construct another polynomial *f''(x)* With degree *n* such that $f''(0) = b$ and generate public shares, $f''(x_v^{'}) = y_v^{''},$ for $v = 1,2,...,n-t+1$. The public shares, $f''(x_v^{'}) = y_v^{''},$ for $v = 1,2,...,n-t+1,$ and the one-way hash value *b* are made publicly known.

Figure 3.3 Master Secret Generation/Renewal Protocol

Remarks:

(a) Dealer can use the Lagrange Interpolating formula to construct the polynomial $f(x)$ as $f(x) = \sum_{i=0}^{n} y_i \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j} \bmod p.$

(b) The polynomial $f(x)$ is of degree *n* and the threshold value is *t* (i.e. $t \le n$).

Therefore, dealer need to publish $n-t+1$ public shares for secret reconstruction.

(c) The one-way hash value *b* and the public shares $f''(x_v') = y_v'', \text{ for } v = 1,2,...,n-t+1,$ are used to authenticate the public shares, $f(x_v') = y_v', \text{ for } v = 1,2,..., n-t+1,$ of the master secret.

3.4.3 Master Secret Reconstruction

Let the set of shareholders $\{U_{i_1}, U_{i_2} ..., U_{i_t}\}$ wants to recover the master secret *s* that is hidden in the constant term of the polynomial *f(x)*. According to the Lagrange interpolation formula, with knowledge of *t* private shares, $f(x_{i_i})$, for *i=1,2,...t,* and *n-t+1* public shares, $f(x_v'),$ for *i=1,2,...n-t+1*, the constant term of the interpolating polynomial is

Table 3.1 Constant Term of the Interpolating Polynomial

$$f(0) = \sum_{i=1}^{t} [f(x_{i_i})(\prod_{r=1, r\neq i}^{t} \frac{-x_{i_r}}{x_{i_i} - x_{i_r}})(\prod_{r=1}^{n-t+1} \frac{-x_r'}{x_{i_i} - x_r'})] + \sum_{i=1}^{n-t+1} [f(x_i')(\prod_{r=1}^{t} \frac{-x_{i_r}}{x_i' - x_{i_r}})(\prod_{r=1, r\neq i}^{n-t+1} \frac{-x_r'}{x_i' - x_r'})] \bmod p.$$

Shareholders need to protect their private shares in order to reuse them in reconstructing multiple master secrets. The following protocol protects private shares unconditionally.

3.4.4 Master Secret Reconstruction Protocol

(i). Each shareholder U_{i_i} uses his private share $f(x_{i_i})$ on the secret polynomial, *f(x),* to compute

$$s_i = f(x_{i_i})(\prod_{r=1, r\neq i}^{t} \frac{-x_{i_r}}{x_{i_i} - x_{i_r}})(\prod_{r=1}^{n-t+1} \frac{-x_r'}{x_{i_i} - x_r'}) \bmod p.$$

(ii).Each shareholder U_{i_i} selects a random polynomial $f_i(x)$, with degree *t-1*, such that $f_i(0) = s_i$, and computes $f_i(x_{i_r})$, for $r = 1,2,...,t.$ U_{i_i} sends each $f_i(x_{i_r})$ to shareholder U_{i_r} secretly, for $r = 1,2,...,t,$ and $r \neq i.$

(iii).After receiving all $f_r(x_{i_i})$, for $r = 1,2,...,t, r \neq i,$ each shareholder U_{i_i} computes

$y_i = \sum_{r=1}^{t} f_r(x_{i_i}) \bmod p,$ and makes this value publicly known.

(iv). After revealing $y_r,$ for $r = 1,2,...,t,$ a Lagrange interpolating polynomial with degree *t-1* on these *t-1* points can be constructed. The constant term of the polynomial is $s' = \sum_{i=1}^{t} s_i \bmod p.$.

The master secret is $s = s' + \sum_{i=1}^{n-t+1} [f(x_i')(\prod_{r=1}^{t} \frac{-x_{i_r}}{x_i' - x_{i_r}})(\prod_{r=1, r\neq i}^{n-t+1} \frac{-x_r'}{x_i' - x_r'})] \bmod p.$

Figure 3.4 Master Secret Reconstruction Protocol

Theorem: The master secret reconstruction protocol protects private shares unconditionally.

Proof: In Step 2, each shareholder uses secret sharing scheme to divide his private share into t pieces and in Step 3, each shareholder uses additive homomorphism of linear polynomials to hide each piece of private share in the sum of shares of t randomly selected polynomials. By revealing the sum of shares, the security of each private share is unconditionally protected.

3.5 Verifiable Secret Sharing

In a *(t, n)* secret sharing scheme, consistency check should be performed on all the shares for fare reconstruction of the secret s. In [18], Lein et al. defined consistency in the following way:

Let S be the domain of a secret s and T be the domain of shares corresponding to the secret. We say that the function $F_I : T^t \rightarrow S$ is an induced function of the *(t, n)*-SS for each subset with $|I| = t$. This function defines the secret s as follows with any set of t shares $s_{i1}, \ldots, s_{it}$

$$s = F(I) = F_I(s_{i1}, \ldots, s_{it}), \text{ where } I = \{i1, \ldots, it\}.$$

Actually, the sharing secret s is computed from the polynomial $f_I(x)$ which is constructed by the interpolation of the points $(i_1, s_{i1}), \ldots, (i_t, s_{it})$.

Definition (*Consistency*): In a *(t , n)*-SS scheme, let $m \geq t$, a set of m shares $s_1, s_2, \ldots, s_m$ is said to be consistent if any subset containing t shares of the set reconstructs the same secret. Formally, let $T = \{T_1, \ldots, T_u\}$ be the set of u elements where each element contains t shares of the set of m shares, where $u = \binom{m}{t}$ denotes the total number of these subsets, then we have

$s^i = F(Ti) = F_{Ti}(s_{i1}, \ldots, s_{it})$, where $i = 1, \ldots, u$.

$s1, s2, \ldots, sm$ are consistent which means that $s^1 = \cdots = s^u$. Moreover, if $s_1, s_2, \ldots, s_m$ are consistent, then the reconstructed secrets s_i for $i = 1, \ldots, u$ are all identical.

Remark: In fact, since all shares are generated by a polynomial in Shamir's *(t, n)*-SS scheme, to check whether m, where $m \geq t$, shares are consistent or not, we only need to check whether the interpolation of m points $(1, s_1), \ldots, (m, s_m)$ yields a polynomial with degree $t - 1$ or not. If this condition is satisfied, we can conclude that all secrets s^i for $i = 1, \ldots, u$ are identical and all shares are consistent. This approach to check shares consistency only requires *one* computation instead of u combinations of t out of m shares. In VSS, all shareholders work together to perform this consistency check to verify if the dealer has shared correct shares or not.

The concept of VSS was first introduced by Chor et al.[12] in 1985. In verifiable secret sharing (VSS) the object is to resist malicious players as discussed in [12], such as

(1). a dealer sending incorrect shares to some or all of the participants, and

(2). participants submitting incorrect shares during the reconstruction protocol

The verifiability is an important property in secret sharing scheme; as all the shareholders work together to verify that all shares are *t-consistent* without revealing the shares or the secret. If any subset of t out of n shares can produce the same secret, then the shares are said to satisfy the property of *t-consistency*. Benaloh [2] introduced the notation of *t-consistency* to determine whether a secret sharing is verifiable or not. In secret sharing scheme, all the shareholders trust the dealer that their shares are consistent. There are two aspects of the security in a VSS. One is the security of the secret and the other is the security of the verification. Verifiable Secret Sharing (VSS) schemes guarantee the robustness of the sharing and the detection of corrupt players.

Normal definition of VSS is as follows :

Suppose there is a dealer D and participants $P_1, \ldots, P_n$ connected by private channels. They also have access to broadcast channel. There is a static adversary A, that can corrupt a set

of the participants from Δ_A including the dealer D. Here static means that the participants controlled by the adversary are fixed.

Let π be a protocol, consisting of two phases: *Share* and *Reconstruct*.

- At the beginning of the *Share* the Dealer inputs a secret $s \in K$.
- At the end of *Share* each participant P_i is instructed to output a Boolean value ver_i.
- At the end of *Reconstruct* each participant is instructed to output a value in K.

The first unconditionally secure proactive VSS was proposed by Stinson and Wei [31]. A generalization of this scheme to general access structures has later been given in [24]. Recently D'Arco and Stinson ([14], [15]) showed that some existing unconditionally secure proactive schemes [30, 15] can be broken.

An unconditionally secure VSS, can be defined as follows:

The protocol π is unconditionally secure Verifiable Secret Sharing protocol if the following properties hold:

(i). If a good player P_i outputs $ver_i = 0$ at the end of Share then every good player outputs $ver_i = 0$;

(ii). If the dealer D is good, then $ver_i = 1$ for every good P_i;

(iii). If a group of good players P_i output $ver_i = 1$ at the end of *Share*, then there exists an $s' \epsilon K$ such that the event that all good P_i output's at the end of *Reconstruct* is fixed at the end of *Share* and $s' = s$ if the dealer is good;

(iv). If $|K| = q$ and s is chosen randomly from K, and the dealer is good, then any forbidden coalition cannot guess at the end of *Share* the value s with probability better than $1/q$.

Figure 3.5 Unconditionally Secure VSS Protocol

If the dealer is malicious and deals with inconsistent shares among the shareholders, then the shares can no longer derive the secret from such inconsistent shares. So, Verifiable secret sharing scheme allows the shareholders to check whether shares are consistent or not. Feldman and Pederson VSS are based on hard to invert the homomorphism functions, and in particular on the hardness of computing discreet logarithms over Z_p for the prime p. Feldman's [17] VSS is

based on computational assumptions in protecting secrecy of the secret; while Pederson's [26] scheme is based on theoretical assumption of protecting the secrecy of the secret. Feldman's [17] and Pedersen's [26] schemes have similar efficiency but slightly different security guarantees. Feldman's [17] scheme is perfectly binding (i.e., an un-trusted dealer cannot fool shareholders into accepting an invalid sharing) and computationally binding (meaning that secrecy is subject to computational hardness assumptions and the amount of computation available to the shareholders). Pedersen's [26] scheme, on the other hand, is computationally binding and perfectly hiding. We focus on Feldman's [17] scheme because it is notationally easier to describe than Pedersen's [26] scheme; however, the systems we describe could be adapted to either scheme. We discussed Feldman and Benaloh's scheme below.

Feldman Scheme :

(i).Let $p=mq+1$ (*where p,q are primes and m is an integer*). Let g be an element of Z_p of order q. The dealer chooses a polynomial f over Z_q with co-efficients $f_0, f_1, ..., f_k$ and broadcasts the values $g^{f0}, g^{f1}, ..., g^{fk}$. the it secretly transmits the value $x_i = f(i)(mod\ q)$ to P_i. Each shareholder(server) checks its own share by verifying the following equation

$g^{x}{}_i = (g^{f}{}_0)(g^{f}{}_1)^i(g^{f}{}_2)^{i2}.....(g^{f}{}_k)^{ik}(mod\ p)$

(ii).If the equation holds good, then each shareholder (P_i) broadcasts its message accepting as proper. If all the shareholders find their shares as correct; then it is believed that the dealer honestly shared consistent shares to the shareholders.

(iii). By the homomorphic properties of the exponential function($g^{a+b} = g^a g^b$), the above equation holds good for all $i \varepsilon \{1....n\}$ if and only if the shares were dealt correctly. $g^{x}{}_i$ is not only used to verify the correct dealing of shares but also to check the consistency of shares during secret reconstruction.

Figure 3.6 Feldman VSS Scheme

Benaloh's Scheme :

In Shamir's Secret Sharing Scheme, all the shares are *t-consistent* if and only if the interpolation of the points $(1, s_1), (2, s_2)....(n, s_n)$ yields a polynomial of degree at most *t-1*. Benaloh's protocol allows the shareholders to perform the required validation while verifying the dealers authenticity and integration. The *(t, n)* VSS proposed by Benaloh can only ensure that all shares are *t*-consistent; but shares may not satisfy the security requirements of a *(t ,n)* secret sharing scheme.

Proof:

(i). As in the secret sharing scheme, the dealer choses a random polynomial *P* and distributes its shares among the shareholders.

(ii).Dealer constructs a very large polynomial $P_1, P_2....P_k$ of degree *t* and distributes it shares.

(iii).Shareholders choose a random set of polynomials $(m<k)$.

(iv).Dealer reveals shares of the *m* chosen polynomials $P_{i1},...,P_{im}$ and sums of remaining *k-m* sums $P + \sum^{k}_{j=m+1} P_j$ then shares the result as well.

(v). Each share-holder or verifier ascertains that all revealed polynomials are degree-t, and corresponds to its own known share.

Throughout this process, the secret *s* remains safe and unexposed.

Figure 3.7 Benaloh's Scheme

3.5.1 Strong Verifiable Secret Sharing

Without revealing the actual secret and the corresponding shares, all shareholders need to work together to determine that any subset of *t* shares are able to recover the same secret. A verifiable secret sharing scheme allows the users to check for the consistency of shares by working together.

Benaloh proposed that shares in Shamir's secret sharing scheme are *t*-consistent if and only if the interpolation of *n* shares yields a polynomial of degree at most *t-1*. This implies that if the interpolating polynomial of *n* shares is with degree at most *t-1*, then all shares are *t*-consistent. However, the property of *t*-consistency does not guarantee that all shares satisfy the security requirements of a secret sharing scheme as pointed out in [21]. For example, if the interpolating polynomial of *n* shares is with degree *t-2*, then all shares are both (*t-1*)-consistent and *t*-consistent. The polynomial with degree *t-2* can be reconstructed with only *t-1* (which is less than the threshold *t*) shares. This condition violates the security requirement of a secret sharing scheme, i.e., at least *t* shares are needed to reconstruct the secret. Therefore, a new notion of strong *t-consistency* is presented below.

Strong t-consistency- A set of shares are said to be strong t-consistent, if in a set of n shares, any subset containing t or more than t shares reconstruct the same secret (i.e. $n \geq t$).

Shares in Shamir's secret sharing scheme [29] are strong *t*-consistent if and only if the interpolation of *n* shares yields a polynomial of degree *t-1* exactly. It is obvious that if all shares in Shamir's secret sharing scheme are generated by a polynomial with degree exactly *t-1* , then (a) all shares are *t*-consistent, and (b) all shares satisfy the security requirements of a secret sharing scheme.

According to the security requirements of secret sharing scheme, at least t shares are needed to reconstruct the secret. If there are more than t shares presented, strong t-consistency check can be performed by constructing a Lagrange interpolation polynomial of these shares. If the interpolating polynomial has degree exactly $t-1$, shares are strong t-consistent. In this paper, Strong Verifiable Secret Sharing (SVSS) protocol has been proposed that allows all shareholders to verify that their shares are strong t-consistent.

3.5.2 Strong Verifiable Secret Sharing Protocol

Verifiable secret sharing is needed whenever the master secret has been renewed or shares have been added/ revoked. Shareholders have to verify their private shares after the modification; and check whether they are strong $(n+1)$-consistent without revealing their corresponding private shares and the master secret.

SVSS Protocol

(i). Each shareholder U_i selects a random polynomial $f_i(x)$ with degree exactly equal to n, and computes $f_i(x_r)$, for $r=1,2,....,n$ and $f_i(x_v')$, for $v=1,2,.....,n-t+1$. U_i sends each $f_i(x_r)$ to shareholder U_r secretly, for $r=1,2,.....,n$ and $r \neq i$, and publishes $f_i(x_v')$, for $v = 1,2,..., n-t+1$.

(ii). After receiving $f_r(x_v')$ and $f_r(x_i)$, for $v = 1,2,..., n-t+1$, and $r = 1,2,..., n$, each shareholder U_i uses his private share y_i of the polynomial $f(x)$ to compute

$z_i = f(x_i) + \sum_{r=1}^{n} f_r(x_r) \bmod p$, and make this value publicly known. Also, each

Shareholder computes $z_v = f(x_v') + \sum_{r=1}^{n} f_r(x_v') \bmod p$, for $v = 1,2,..., n-t+1$.

(iii). After revealing z_i, for $r = 1,2,..., n$, and z_v, for $v = 1,2,..., n-t+1$, a Lagrange interpolating polynomial of these $2n-t+1$ points, z_i and z_v, can be constructed. If this polynomial has degree n exactly, shareholders are convinced that their private shares are strong n-consistent.

Figure 3.8 Strong Verifiable Secret Sharing Protocol

A SVSS protocol can guarantee that all shares are strong *n*-consistent without revealing private shares and the secret. In Step 1, each shareholder selects a random polynomial with degree *n*. In Step 2, each shareholder use additive homomorphism of linear polynomial to hide his private share in the sum of his private share and the sum of shares of *n* random polynomials. By revealing this sum, the security of each private share is unconditionally protected. In Step 3, if the interpolating polynomial has degree *n* exactly, the degree of polynomial *f(x)* is at most *n*. This is because the interpolating polynomial is $f(x)+\sum_{r=1}^{n} f_r(x) \bmod p$ and each shareholder U_i has contributed one polynomial $f_i(x)$ with degree *n* exactly in $\sum_{r=1}^{n} f_r(x) \bmod p$. Since $f(x_i) = y_i'' + f'(x_i)$ and $f'(x)$ is a polynomial with degree *n* exactly, shareholders can further conclude that the polynomial *f(x)* is with degree *n* exactly. Thus, all shares are strong *n*-consistent.

3.6 Shares Adding/Revoking Protocol

When dealer wants to assign new shares to new shareholders, dealer can just select new points on the polynomial $f(x)$ of the master secret. After this modification, all shareholders need to work together to verify that their shares are strong $(n+1)$-consistent. This is because new shares have been added and shareholders want to verify whether these new shares are consistent with the existing shares.

When dealer wants to revoke old shares, dealer needs to invoke the master secret renewal protocol in Section 3.4. In this protocol, dealer need to construct a new polynomial. This new polynomial will pass through all remaining shares and the old master secret except the revoked shares. Dealer computes the public shares of the renewed master secret and makes these public shares publicly known. After revoking old shares, shareholders need to verify whether these remaining shares are consistent since a new polynomial and new public shares are generated.

In both situations, shareholder's shares have not been changed. No private channel is needed to distribute private share to each shareholder.

3.7 Changing Threshold Protocol

Dealer divides the secret *s* into *n* shares and divides it among *n* shareholders in such a way that at least *t (where t is the threshold value and the threshold value can be altered)* shares are required to reconstruct the secret where as less than *t* shares gain no information about the secret. Dealer can either increase or decrease the threshold *t* of the master secret *s*. Whenever dealer decide to change the threshold of the master secret, our proposed PSS allows dealer to accomplish the task without changing private shares. This provides an efficient way to change threshold dynamically.

3.7.1 Decreasing the Threshold

When dealer want to reduce the threshold without changing the old master secret, dealer just need to publish additional public shares.

3.7.2 Increasing the Threshold

On the other hand, when dealer want to increase old threshold to a new threshold *t'* without changing the old master secret, the dealer can select a polynomial *g(x)* with degree [*n+(t'-t)*] exactly and $g(0) = 0$. Dealer publishes this polynomial *g(x)*. Dealer can employ the public shares authentication protocol in Section 3.3.3 to enable all shareholders to authenticate this public polynomial. Each shareholder U_i updates his new share value by himself as $f(x_i) + g(x_i).$ Also, public shares of the master secret should be updated as $f(x_v') + g(x_v') = y_v' + g(x_v'),$ for $v = 1,2,\ldots, n-t+1,$

3.7.3 Strong t-consistency Check

Strong *t*-consistency check does not need for both situations. Under the situation to decrease the threshold, since the polynomial and the shares are unchanged, shares must be strong (*n+1*)-consistent as before. Under the situation to increase the threshold, since the new share is the sum of share of $f(x)$ and the share of $g(x)$ which is a public polynomial with degree [*n+(t'-t)*], the new shares must be strong [*n+(t'-t)+1*]-consistent if the shares of $f(x)$ are strong (*n+1*)-consistent.

CHAPTER 4

CONCLUSION

In secret sharing scheme, the master secret and all the private shares (which are distributed by the dealer to the shareholders) are the two secrets which are to be maintained confidentially. In all the proactive secret sharing schemes proposed till date, private shares are reused to reconstruct the master secret. But in our approach, instead of renewing the private shares frequently at the beginning of each timeslot during the share renewal process, each time master secret is renewed.

All the existing Proactive secret sharing schemes assume that an adversary can attack only the shares but not the master secret. Therefore, shares are renewed at the begin of each time period. In our proposal, we considered a complete different approach of Proactive secret sharing in which the master secret is to be renewed frequently but not the private shares. After each renewing process, private shares remain the same and can be reused for reconstructing different master secrets. In addition, after each renewing process, shareholders can work together to verify that their private shares are consistent without revealing private shares. If once the shareholders are able to reconstruct the master secret, then the secret is no longer secure. So if all the shares are kept secure during the secret reconstruct, then the dealer needs to renew only the master secret. By adopting the proposed solution, implementation of secret sharing makes proactive secret sharing even more practical and flexible.

There are several properties like Asynchrony, Efficiency, Threshold adaptivity which are to be satisfied by proactive secret sharing schemes to be useful in real systems. We mentioned our approach of proactive secret sharing; public and private share generation and authentication

protocols. Then we discussed about Master secret renewal protocol, strong verifiable secret sharing protocols which can be applied for the proposed scheme.

4.1 Open Problems and Future Work

Proactive secret sharing scheme can be used in Public Key Infrastructure (PKI) schemes where Certificate Authority (CA) is used to generate digital signatures. The digital certificate contains a public key, digital signature and identity of the owner of the certificate. In such key management applications our scheme can be applied. CA has a private key for generating digital signatures. Instead of storing the entire key secretly, it is better to divide the key into number of shares and share the key. This private key can be stored as a secret; and is shared in a secret sharing scheme. Once if all the private shares are able to recover the master secret, then the secret is no longer secure. So it is necessary to renew the master secret keeping all the private shares secure. If all the private shares are kept secure during secret reconstruction, dealer only needs to renew master secret but not the private shares. So in this way, the secret sharing scheme becomes even more efficient as the private shares can be reused for a longer period of time.

If once the share holders are able to recover the master secret, then the secret is no longer secure. So it is necessary to refresh both the shares and secret at the same time. The future work could be to incorporate both secret and share refreshment together in a complete key management systems.

One of the limitations of proactive secret scheme is that they assume the set of share holders remain same forever. The proactive secret sharing scheme could further be extended in adding and deleting shareholders while sharing the same secret.

REFERENCES

1. Araki, T. Efficient (k, n) threshold secret sharing schemes secure against cheating from n − 1 cheaters. In Proceedings of ACISP'07 (Queensland, Australia, July 2007) LNCS, vol. 4586, pp. 133–142.
2. Benaloh, J.C. Secret sharing homomorphisms: keeping shares of a secret. In Proceedings on Advances in Cryptology --- Crypto '86 (Santa Barbara, CA, USA, Jan 1987) LNCS, vol. 263, pp. 251-260.
3. Benaloh, J.C., and Leichter, J. Generalized secret sharing and monotone functions. In Proceedings on Advances in Cryptology --- Crypto'88 (Santa Barbara, CA, USA, Aug 1988) LNCS, vol. 403, pp. 27-35.
4. Ben-Or, M., Goldwasser, S., and Wigderson, A. Completeness theorems for non-cryptographic fault tolerant distributed computation. In Proceedings of ACM Symposium on theory of computing (Chicago, IL, USA, May 1988) pp. 1-10.
5. Bhndo, C., De Santis, A., Gargano, L., and Vaccaro ,U. Secret sharing schemes with veto capabilities, In Proceedings of the First French-Israeli Workshop on Algebraic Coding'93 (Paris, France, July 1993) LNCS, vol. 781, pp. 82–89.
6. Blakley, G.R. Safeguarding cryptographic keys. In Proceedings of National Computer Conference---AFIPS (New York, NY, USA, June 1979) vol. 48, pp. 313–317.
7. Brickell, E.F., Stinson, D.R. The detection of cheaters in threshold schemes. In Proceedings on Advances in Cryptology --- Crypto'88 (Santa Barbara, CA, USA, Aug 1988) LNCS, vol. 403, pp. 564–577.

8. Cachin, C., Kursawe, K., Petzold, F., and Shoup V. Secure and efficient asynchronous broadcast protocols. In Proceedings on Advances in Cryptology --- Crypto'01 (Santa Barbara, CA, USA, Aug 2001) vol. 2139, pp. 524–541.

9. Cachin, C., Kursawe, K., Lysyanskaya, A., and Strobl, R. Asynchronous verifiable secret sharing and proactive cryptosystems. In Proceedings of the 9th ACM conference on Computer and Communications Security'02 (Washington, DC, USA, Nov 2002) pp. 88-97.

10. Canetti R., and Herzberg, A. Maintaining security in the presence of transient faults. In Proceedings on Advances in Cryptology --- Crypto'94 (Santa Barbara, CA, USA, Aug 1984) vol. 839, pp. 425-438.

11. Chaum., D., Cr´epeau., C., and Damgard, I. Multiparty unconditionally secure protocols. In Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing (Chicago, Illinois, May 1988) pp. 11-19.

12. Chor, B., Goldwasser, S., Micali, S., and Awerbuch, B. Verifiable secret sharing and achieving simultaneity in the presence of faults. In Proceedings of IEEE Symp. on Foundations of Computer Science'85 (Portland, OR, USA, Oct 1985) pp. 383–395.

13. Courtois, P.J., Heymans, F., and Parnas, D. Concurrent control with "readers" and "writers." Communications of the ACM, 14, (1971), pp. 667–668.

14. D'Arco, P., and Stinson, D. On Unconditionally Secure Proactive Secret Sharing Scheme and Distributed Key Distribution Centers, Manuscript, May 2002.

15. D'Arco, P., and Stinson, D. R. In Proceedings of AsiaCrypt'02 on Unconditionally Secure Robust Distributed Key Distribution (Queensport, New Zealand, Dec 2002) vol. 2501, pp. 346-363.

16. Desmedt, Y., and Jajodia, S. Redistributing secret shares to new access structure and its applications, Technical Report ISSETR-97-01, George Mason University, July 1997.
17. Feldman, P. A practical scheme for non-interactive verifiable secret sharing. In Proceedings of 28th IEEE Symposium on Foundations of Computer Science (October 1987), pp. 427-437.
18. Harn, L. and Lin, C. Detection and identification of cheaters in secret reconstruction, Designs, Code and Cryptography, 52 1 (2009), pp. 15-24.
19. Herzberg, A., Jarecki, S., Krawczyk, H., and Yung, M. Proactive secret sharing or: How to cope with perpetual leakage. In Proceedings on Advances in Cryptology---Crypto'95, (Santa Barbara, CA, USA, Aug 1984) LNCS 963, pp. 339-352.
20. Lamport, L. Concurrent reading while writing. Communications of the ACM, 20, (1977), pp. 806-811.
21. Lin, C., Harn, L., and Yea, D. Information-theoretically secure strong verifiable secret sharing. In Proceedings of the Fifth International Conference on Information Conference on Security and Cryptography---Secrypt '09 (Milan, Italy, July 2009), pp. 233-238.
22. Liu, C.L. Introduction to Combinatorial Mathematics. McGraw-Hill, New York, 1968.
23. McEliece, R.J., and Sarwate, D.V. On sharing secrets and Reed-Solomon codes. Commun. ACM, 24, (1981), pp. 583-584.
24. Nikov, V., Nikova, S., Preneel, B., and Vandewalle, J. Applying general access structure to proactive secret sharing schemes. In Proceedings of the 23rd Symposium on Information Theory in the Benelux, (2002), pp. 197-206.
25. Ostrovsky, R., and Yung, M. How to withstand mobile virus attacks. In Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computer---PODC '91 (Montreal, Quebec, Canada, Aug 1991), pp. 51-59.

26. Pedersen, T.P. Non-interactive and information-theoretic secure verifiable secret sharing, In Proceedings on Advances in Cryptology --- Crypto '91 (Santa Barbara, CA, USA, Aug 1992) LNCS 576, pp. 129-140.

27. Rabin, T., and Ben-Or, M. Verifiable secret sharing and multiparty protocols with honest majority. In Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing---STOC'89 (New York, NY, USA, May 1989), pp. 73-85.

28. Rabin T. A simplified approach to threshold and proactive RSA. In Proceedings on Advances in Cryptology---Crypto'98 (Santa Barbara, CA, USA, Aug 1998) vol. 1462, pp. 89–104.

29. Shamir, A. How to share a secret. Commun. ACM, 22 11 (Nov. 1979), pp. 612-613.

30. Simmons, G. An introduction to shared secret schemes and their applications. Sandia Report SAND 88-2298, 1988.

31. Stinson, D.R., and Wei, R. Unconditionally secure proactive secret sharing scheme with combinatorial structures in selected areas in cryptography, In SAC '99, LNCS 1758, (1999), pp. 200-214.

32. Zhou, L. Towards building secure and fault-tolerant on-line services. Ph.D. thesis, Computer Science Department, Cornell University, Ithaca, New York, USA, May 2001.

33. Zhou, L., Schneider, F.B., and Van Renesse, R. APSS: Proactive secret sharing in asynchronous systems. Technical Report TR 2002-1877, Computer Science Department, Cornell University, Ithaca, New York, USA, Oct. 2002.

VITA

Priyanka Koneru was born in Challapalli, Andhra Pradesh, India on 21st April 1987. After completion of 12th standard at Sri Chaitanya College, she went to Koneru Lakshmaiah University to purse her Bachelor of Technology degree in Computer Science. She being highly ambitious moved to Kansas City, MO in Fall'08 to purse Master of Science in Computer Science at University of Missouri Kansas City and graduated in December 2010. During her graduate studies, she worked as Graduate Research Assistant at UMKC, and Web Developer Intern at Bushnell Corporation and Mobile Application Developer Intern at Sprint Nextel. She presented a paper on "Modeling and Analyzing Web-based Information Systems" at "Software Engineering Research and Practice (SERP'10)" held at Las Vegas, NV and the paper got published in the conference proceedings.