Indian Statistical Institute Kolkata

M.Tech (Computer Science Dissertation)

A Survey On Perfectly Reliable and Secure Message Transmission

Author:
Shyam Sundar Das
Roll No:CS0919

Supervisor:
Prof. Rana Barua
Stat-Math Unit

Date: 18th July 2011

A dissertation submitted in partial fulfillment of the requirements for the award of M.Tech.(Computer Science) degree

Indian Statistical Institute

Certificate of Approval

This is to certify that the thesis entitled "A Survey on Perfectly Reliable and Secure Message Transmission Protocols" submitted by **Shyam Sundar Das** towards partial fulfilment for the degree of M.Tech. in Computer Science at Indian Statistical Institute, Kolkata, is a record of the work carried out by him under my supervision and guidance.

(Prof. Rana Barua)
Stat-Math Unit
Indian Statistical
Institute

Date:
Kolkata

Acknowledgements

I take this opportunity to thank **Prof. Rana Barua**, Stat-Math Unit, ISI, Kolkata for his valuable guidance and inspiration. His pleasant and encouraging words have always kept my spirits up.

I would also like to thank **Dr. Ashish Choudhury**, ISI Kolkata, for all his advice and encouragement. For his patience and for the way he helped me to think about problems with a broader perspective, I will always be grateful.

Finally, I would like to thank all of my class mates and friends for their support and motivation to complete this project.

Abstract

In the study of the cryptosystems, we are interested in that type of cryptosystems which provide perfect security against a computationally unbounded third party. When the *sender* **S** and the *receiver* **R** are connected by a single channel which can reveal information to a third party, it is not possible to design a cryptosystem which does not reveal any information about the message. This leads to a problem where the *sender* and the *receiver* are part of an unreliable, connected, distributed network. The distrust in the network is modeled by an entity called *adversary*, who has *unbounded computing power* and who can corrupt some of the nodes of the network (excluding **S** and **R**) in a variety of ways. **S** wants to send to **R** a message $m^{\mathbf{S}}$ that consists of ℓ elements, where $\ell \geq 1$, selected uniformly from a finite field Z_q. Our aim is to design a protocol, such that the receiver will recover the message correctly after communicating certain number of times with the sender. And the receiver should be able to recover the message in spite of the presence of the disrupting authority. The protocols which are able to do this is called *perfectly reliable message transmission (PRMT) protpcol*. Now, if the protocols are such that, the adversary does not get any information about the message, we call them as *perfectly secure message transmission (PSMT) protpcol*. Security against an adversary with infinite computing power is also known as *non-cryptographic* or *information theoretic* or *Shannon* security and this is the strongest notion of security.

RMT and SMT problem can be studied in various network models and adversarial settings. We may use the following parameters to describe different settings/models for studying RMT/SMT: Type of Underlying Network (*Undirected Graph, Directed Graph, Hypergraph*), Type of Communication (*Synchronous, Asynchronous*, Adversary capacity (*Threshold Static, Threshold Mobile, Non-threshold Static, Non-threshold Mobile*). Type of Faults (*Fail-stop, Passive, Byzantine, Mixed*).

Irrespective of the settings in which RMT/SMT is studied, we have to deal with some common issues. First one is that, we have to find out the necessary and sufficient structural conditions to be satisfied by the underlying network for the existence of any RMT/SMT protocol, tolerating a given type of adversary. Now, if a protocol exists in a network our aim will be to find an efficient and optimal protocol. In this dissertation, we look into the above issues in several network models and adversarial settings. More specifically, we survey some well known SMT schemes and study their properties.

Contents

Chapter 1

Background

1.1 Brief History of Cryptography

Cryptography is the science of devising methods that allow for information to be sent in a secure form in such a way that the only person able to retrieve this information is the intended recipient. It is one of the oldest fields of technical study. We can find records of *Cryptography*, going back at least 4000 years. Ciphering(encrypted message) has always been considered vital for diplomatic and military secrecy. Cryptography probably began in or around 2000 B.C. in Egypt. Probably these were not the serious attempts for secret communications, but rather to have been attempts at mystery, intrigue, or even amusement for literate onlookers. The ancient Greeks are said to have known of ciphers (e.g., the scytale transposition cipher claimed to have been used by the Spartan military). Herodotus tells us of secret messages physically concealed beneath wax on wooden tablets or as a tattoo on a slave's head concealed by regrown hair, though these are not proper examples of cryptography as the message, once known, is directly readable; this is known as *steganography*(it means hidden writing, e.g.invisible ink, using low-definition bits of sound/graphics files. Advantage is that, it does not reveal to enemy that you have information to hide). The cryptographic history of Mesopotamia was similar to that of Egypt, in that cuneiforms were used to encipher text. This technique was also used in Babylon and Assyria. In the Bible, a Hebrew ciphering method is used at times. In this method, the last letter of the alphabet is replaced by the first, and vice versa. This is called 'atbash'. The Greek writer Polybius invented the 5 x 5 Polybius Square, widely used in various cryptographic systems. Julius Caesar used a system of advancing each letter four places, commonly called a Caesar shift. During the Middle Ages, cryptography started to progress. All of the Western European governments used cryptography in one form or another, and codes started to become more popular. The earliest ciphers involved only vowel substitution (leaving consonants unchanged). By 1860 large codes were in common use for diplomatic communications, and cipher systems had become a rarity

for this application. Cipher systems prevailed, however, for military communications except for high-command communications because of the difficulty of protecting codebooks from capture or compromise in the field. The invention of telegraph and radio pushed forward the development of cryptographic protection of telecommunications: the speed and the volumes of data traffic became considerable and more vulnerable to interception and decryption. The radio espionage was closely following the development of new telecommunications technologies, but paradoxically, the telegraphic and radio exchange of information was mainly in clear or done in plain ciphers. It was not until the 20th century that mathematical theory and computer science have both been applied to cryptanalysis. As the science of cryptology becomes increasingly sophisticated, most nations have found it necessary to develop special governmental bureaus to handle diplomatic and military security for example, the National Security Agency in the United States. The widespread use of computers and data transmission in commerce and finance is making cryptography very important in these fields as well. Recent successes in applying certain aspects of computer science to cryptography seem to be leading to more versatile and more secure systems in which encryption is implemented with sophisticated digital electronics. Industry, however, have argued over who will have ultimate control over data encryption and, as a result, over government access to encrypted private transmissions.

1.2 Motivation For Modern Cryptography

There are many areas which contains sensitive information that have to be sent to another location. For example

1. *Military:* battle plan or information regarding weapons needs to be reached to the battle ground from military headquarter.
2. *Finance:* transactions using ATM card or online banking.
3. *Diplomacy:* sensitive negotiations between different governments need to be remain secret.

Basically in every circumstances, when we are trying to communicate some secret data at a distance and do not want anyone other than the intended receiving party to know the information, there will be application of cryptography.

1.3 Definition of a Cryptosystem

The fundamental objective of cryptography is to enable two parties, usually referred to as Alice and Bob, to communicate over an insecure channel in such a way that a third party (also popularly known as *adversary*) who is not an intended receiver, cannot understand what is being said. This channel could be a telephone line or computer network, for example. The information that Alice

wants to send to Bob, which we call *plaintext*, can be English text, numerical data, or anything at all - its structure is completely arbitrary. Alice encrypts the *plaintext*, using a predetermined key, and sends the resulting *ciphertext* over the channel. Oscar, upon seeing the *ciphertext* over the channel by eavesdropping, cannot determine what the *plaintext* was; but Bob, who knows the *decryption key*, can decrypt the *ciphertext* and reconstruct the *plaintext*.

Cryptosystem ([9]): Let us consider a five-tuple (P, C, K, E, D), where

1. P is the finite set of possible *plaintext*;
2. C is the finite set of possible *ciphertext*;
3. K is the finite set of possible keys;
4. E is the set of *encryption rule* $Enc_k : P \to C$;
5. D is the set of *decryption rule* $Dec_k : C \to P$.

Then, this five-tuple is a **cryptosystem** if for each $k \in K$, there exists an *encryption rule* $Enc_k \in E$ and a corresponding *decryption rule* $Dec_k \in D$ which satisfies $Dec_k(Enc_k(p)) = p, \forall p \in P$.

1.4 Classification of Cryptosystem

Depending upon the adversarial computing power and the encryption key used by the sender(S) & the decryption key used by the receiver (R), we can classify the cryptosystem in four groups. We show this classification in *Figure 1.1*.

1. The Cryptosystem is conditionally secure and the sender and the receiver use the same key for encryption and decryption. The Cryptosystem in which the sender and the receiver use the same key for encryption and decryption, we call them as *Symmetric Key Encryption*. Some examples of this type of cryptosystems are *AES* (*Advanced Encryption Standard*), *DES* (*Data Encryption Standard*) etc.

2. The Cryptosystem is conditionally secure and the sender and the receiver use the different keys for encryption and decryption. The Cryptosystem in which the sender and the receiver use the different keys for encryption and decryption, we call them as *Public Key Encryption*. Some examples of this type of cryptosystems are *RSA*, *ElGamal* etc.

3. The Cryptosystem is unconditionally secure and the sender and the receiver use the same key for encryption and decryption. One-time Pad which was first described by **Gelbert Vernan** in 1917, is the example of this type of cryptosystem. Unfortunately, there are major disadvantages to the One-time Pad. For the One-time Pad, we require $|K| \geq |P|$ which means the amount of key that must be communicated securely is at least as big as the amount of *plaintext*. This would not be a major problem if

the same key could be used to encrypt different messages; however, the security of unconditionally secure cryptosystems depends on the fact that each key is used for only one encryption. To overcome this problem, we require the forth type of cryptosystem where the sender and the receiver use the different keys for encryption and decryption.

4. The Cryptosystem is unconditionally secure and the sender and the receiver use the different keys for encryption and decryption. Some examples of this type of cryptosystem are *wire-tape channel model* by **Wyner** in 1975 and *Quantum mechanics* by **Bennett and Brassard** in 1984. But these are basically theoretical. In 1993 Dolev, Dwork, Waarts and Yung [1] proposed a **Network model**. In this model, they assumed that the sender and the receiver are the part of a network consisting of nodes and connections between these nodes. They also assume that some of these nodes can be controlled by the computationally unbounded adversary. First, they creates the node disjoint paths(which are known as channels) between the sender and the receiver. If a channel contains some node controlled by the adversary, then they consider that channel is under the control of adversary.

Figure 1.1: The Classification of Cryptosystem.

Adversarial power / key	*same key by S and R*	*different keys by S and R*
conditionally secure	*Symmetric Key Encryption*	*Public Key Encryption*
unconditionally secure	*One Time Pad*	??

If the sender and receiver is connected by only one channel, then it may not be possible to design a cryptosystem for this group. So, we will assume that there are more than one channel between sender and receiver and at least one channel is not under the control of adversary. This leads to a new type of message sending method called *reliable and secure message transmission.*

1.5 Introduction To Secure Message Transmission

It often occurs that two parties, a sender and a receiver, want to transmit a message to one another while guaranteeing the secrecy and reliability of the message against any eavesdropping third party. Communication that meets these two requirements is called secure message transmission(SMT). For simplicity, we assume that the communication channel is perfect in the sense that transmission over the channel is error-free, but there may be some adversarial party

that is able to control the communication over the channel. Perfect secrecy can be achieved in this setting by means of onetime-pad encryption, where the sender and receiver somehow at some point before or during the transmission of the message agree on a secret key that consists of at least as many bits as the message and use this key to encrypt the message during the transmission. However,in order to achieve perfect secrecy such a key can only be used once and therefore requires the storage of a lot of key-data in order to allow the transmission of a large number of messages.
If the power of the adversarial party is strengthened up to the point where it is able to modify data transmitted on the communication channel, things become much more difficult.
To get rid of these problems, Dolev, Dwork, Waarts and Yung [1] considered a multi-channel model. In this model, two parties are connected by $n > 1$ communication channels and an adversarial party is able to eavesdrop and modify data, on at most t channels, where $t < n$. In fact, this model can be seen as the natural abstraction of a typical communication network, where the channels are all the vertex-disjoint communication paths from the sending party to the intended receiving party.

This model has two important advantages. First of all, it is possible to prevent an adversary from totally blocking all communication, as at least one channel will always be out of reach. More importantly, when t is small enough compared to n, it is possible to achieve secure communication without using any initial secret key. This strongly separates this model from the traditional model with one communication channel where one always requires either a computational restriction on the adversary or sender and receiver both have the same secret key beforehand.
Throughout this report we will assume that message is an element or sequence of elements of some finite field Z_q.

1.6 RMT, SMT and Its Variants

We now give informal definition of different variants of RMT/SMT protocols [10].

1. An RMT protocol is called δ-reliable, for any $\delta = 2^{-\Omega(\kappa)}$, where κ is the error parameter, if at the end of the protocol, **R** correctly outputs **S**'s message, except with error probability δ. Moreover, this should hold, irrespective of the behavior of the adversary.

2. An SMT protocol is called ϵ-secure, for any $\epsilon = 2^{-\Omega(\kappa)}$, where κ is the error parameter, if at the end of the protocol, the adversary does not get any extra information about **S**'s message, except with probability ϵ.

3. A message transmission protocol is called (ϵ, δ)-secure, if it is ϵ-secure and δ-reliable, for some $\epsilon, \delta > 0$.

4. An RMT protocol is called *perfectly reliable*, also called as PRMT, if it is 0-reliable.

5. An RMT protocol is called *statistically reliable*, also called as SRMT, if it is δ-reliable, for some $\delta > 0$.

6. A message transmission protocol is called *perfectly secure*, also called as PSMT, if it is $(0, 0)$-secure.

7. A message transmission protocol is called *statistically secure*, also called as SSMT, if it is $(0, \delta)$-secure. Such protocols are also called as *almost perfectly secure* protocols.

1.7 Various Models for Studying SMT

There are various network settings and adversary models in which SMT problem can be studied and has been studied in the past. We can use the following parameters to describe the different models for studying SMT:

1. *Underlying network:*

   It can be *Undirected*, *Directed* or *Hypergraph*. The simplest of the network is undirected network model, where it is assumed that link between every two nodes is bidirectional and hence support both way communication. In directed network model, it is assumed that every communication link has a direction associated with it. *Hypergraphs* are the most general form of the network, where instead of edges, we have hyperedges. Each hyperedge will have a source node and a set of receiver. Any information sent by the source node will be received identically by all the receiver of the hyperedge.

2. *Type of communication:*

   Type of communication can be *synchronous* or *asynchronous*. In a synchronous network, there exists a global clock in the system and so the transmission delay along every edge of the network is bounded. On the other hand, in an asynchronous network, there is no global clock in the system. Thus each link in the network has arbitrary but finite delay. The difficulty in designing a protocol over asynchronous network comes from the fact that sender or receiver cannot distinguish between a slow sender and a corrupted sender. Thus, if a receiver node is expecting some message from a sender node and if no message arrives then the receiver cannot distinguish whether the sender node is corrupted and did not sent the message at all or the message is just delayed in the network.

3. *Adversary Capacity:*

   Based on the corruption capacity and adversarial behavior there can be four model- *Threshold* and *Non-Threshold*, *Static* and *Mobile*. In threshold adversary model, the number of channels that can be corrupted by

the adversary is bounded by a threshold. On the other, non-threshold adversary model is a generalization of threshold model. In non-threshold setting, the adversary is specified by an adversary structure, which is a set of all possible set of channels that can be potentially corrupted by the adversary. Moreover, each set in the adversary structure may have different size. If the adversary is static, then it controls the same set of nodes throughout the protocol execution. This is a valid assumption if the protocol is executed for a short period of time. On the other hand, if the adversary is mobile, then the adversary can corrupt different set of nodes during different instances of the protocol. This models a scenario when a protocol is executed for a long duration.

4. *Type of faults:*

   There are four type of faults- *failstop*, *passive*, *Byzatine* and *mixed*. The weakest type of corruption is the failstop corruption, where the adversary can simply stop the complete functioning of a node. Passive corruption means that the adversary has full access to the computation and communication of the node. However, the adversary cannot make the node to deviate from the protocol. The most powerful type of corruption is Byzantine corruption, where a node is completely under the control of the adversary and may behave arbitrarily during the protocol execution. Lastly, the adversary may simultaneously control different set of nodes in passive, fail-stop and active fashion; such a generalized adversary is called mixed adversary.

5. Type of security:

   Based on the security level achieved by a protocol, we may have *perfect security* or *statistical security*. In perfect security, adversary does not get any information about the message. But in statistical security the adversary can get information about the message with certain probability.

A taxonomy of settings in which RMT and SMT problem can be studied is presented in *Figure 1.2*.

Figure 1.2: The taxonomy of settings in which RMT/SMT can be studied.

Underlying Network	Type of Communication	Adversary Capacity	Type of Faults	Type of Security
Undirected Graph *Directed Graph* *Undirected Hypergraph* *Directed Hypergraph*	*Synchronous* *Asynchronous*	*Threshold Static* *Threshold Mobile* *Non-Threshold Static* *Non-Threshold Mobile*	*Byzantine* *Fail-Stop* *Passive* *Mixed*	*Perfect* *Statistical*

From the *Figure 1.1*, we can come up with various models and settings. For example:

1. Underlying network can be undirected & synchronous and the adversary may be threshold, static and Byzantine and the level of security may be perfect.

2. Underlying network can be undirected & synchronous and the adversary may be threshold, mobile and Byzantine and the level of security may be perfect.

3. Underlying network can be undirected & synchronous and the adversary may be non-threshold, mobile and Byzantine and the level of security may be perfect.

4. Underlying network can be directed & synchronous and the adversary may be threshold, static and Byzantine and the level of security may be perfect.

5. Underlying network can be directed & synchronous and the adversary may be threshold, mobile and Byzantine and the level of security may be perfect.

In this report, we will discus different protocols for the following model

1. **Underlying network**- *undirected graph*;

2. **Type of communication**- *synchronous*;

3. **Adversary Capacity**- *threshold and static*;

4. **Type of faults**- *Byzantine*;

5. **Type of security**- *perfect secrecy.*

Throughout this report we will use the notation $\mathcal{A}_{tb}^{static}$ for denoting a static adversary which controls t channels in Byzantine fashion.

In any model, the following 3 questions are fundamental.

1. **Possibility:** What are the necessary and sufficient structural conditions to be satisfied by the underlying network for the existence of any RMT/SMT protocol, tolerating a given type of adversary?

2. **Feasibility:** Once the existence of MT/SMT protocol in a network is ascertained, the next natural question is, does there exist an efficient protocol on the given network?

3. **Optimality:** Given a message of specific length, what is the minimum communication complexity (lower bound) needed by any RMT/SMT protocol to transmit the message and how to design a polynomial time RMT/SMT protocol whose total communication complexity matches the lower bound on the communication complexity (optimal protocol)?

The above taxonomy and a unified framework for a number of research problems were first discussed in [1]. Different techniques are used to resolve the above issues in different settings. For example, the techniques used in designing optimal RMT/SMT protocols in undirected networks are completely different from the ones used in directed networks.

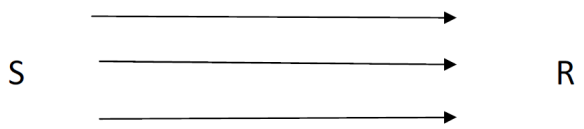
1.8 Parameters of RMT and SMT Protocols

Any RMT and SMT protocol has following 4 parameters:

1. *Round Complexity:* It is the number of rounds, denoted by r, taken by the protocol. A round is a communication from the sender to receiver or vice-versa.

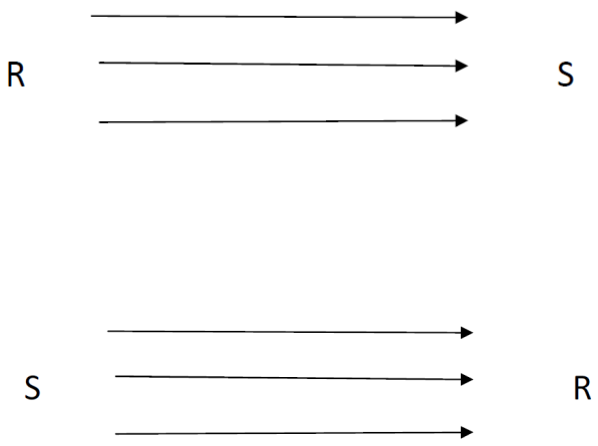
   During the protocol, if the communication is only from the sender to the receiver, we called that protocol as *one-way* protocol and the round complexity is one. This is shown in the *Figure 1.3*.

Figure 1.3: One-way communication.



   On the other hand, if the communication is both way, that is, the sender sends information to the receiver and the receiver also sends information to the sender, we called that protocol as *two-way* protocol and the round complexity is more than one. This is shown in the *Figure 1.4*.

Figure 1.4: Two-way communication.



2. *Communication Complexity:* In any SMT protocol computation and communication is done over a finite field Z_q. Communication complexity is the total number of field elements communicated by the sender and receiver during the protocol. We will mainly concern about this complexity.

3. *Computation Complexity:* It is the amount of computation done by the sender and receiver during the protocol.

4. *Connectivity of The Underlying Network:* For the existence of a SMT protocol, the underlying network should satisfy a minimum connectivity so that there exists enough number of node disjoint path (we will called them as channels) between the sender and the receiver.

1.9 The Known Results

In 1993, Dolev, Dwork, Waarts and Yung [1] started the line of research in *perfectly secure message transmission* ($PSMT$), listing a number of important initial results. One observation was about the number of channels required for the existence of a single-round PSMT protocol against $\mathcal{A}_{tb}^{static}$.

Theorem 1.9.1 ([1]) *A single-round PSMT protocol tolerating $\mathcal{A}_{tb}^{static}$ exists if and only if the number of channels $n \geq 3t + 1$.*

Dolev et al. furthermore discovered that as soon as we allow interaction in the message transmission protocol, i.e., multiple transmission round where the

parties can communicate feedback with regard to the data they received, it is possible to construct perfectly secure protocols under the weaker restriction.

Theorem 1.9.2 ([1]) *A multi-round (more than one round) PSMT protocol tolerating $\mathcal{A}_{tb}^{static}$ exist if and only if the number of channels $n \geq 2t+1$.*

Finally, Dolev et al. showed that no $PSMT$ protocols exist when $n \leq 2t$. Also in the literature there are some results for the the lower bound of the communication complexity of the $PSMT$ protocols. The lower bound on the communication complexity of single round PSMT tolerating $\mathcal{A}_{tb}^{static}$ was proved in two independent works in [2,3].

Theorem 1.9.3 ([2, 3]) *If the sender and receiver are connected by $n \geq 3t+1$ channels, then any single round PSMT protocol tolerating $\mathcal{A}_{tb}^{static}$ must communicate $\Omega(\frac{nl}{n-3t})$ field elements to securely send a message containing l field elements.*

In another interesting work, Srinathan et al.[2] proved the bound for multi round $PSMT$ protocol tolerating $\mathcal{A}_{tb}^{static}$. This result is stated in the following theorem:

Theorem 1.9.4 ([2]) *Let the sender and the receiver be connected by $n \geq 2t+1$ channels. Then any multi-round $PSMT$ protocol tolerating $\mathcal{A}_{tb}^{static}$ must communicate $\Omega(\frac{nl}{n-2t})$ field elements to securely send a message containing l field elements.*

The existing results on $PSMT$ are summarized in the *Figure 1.5*

Figure 1.5: Results for PSMT protocols Tolerating $\mathcal{A}_{tb}^{static}$.

no. of Rounds	lower bound on channels	lower bound on CC
$r = 1$	$n \geq 3t+1$	$\Omega(\frac{nl}{n-3t})$
$r \geq 2t+1$	$n \geq 2t+1$	$\Omega(\frac{nl}{n-2t})$

1.10 Conclusion

In this chapter, we have studied the RMT/SMT and the various models and settings where we can develop the protocols for RMT/SMT. We have discussed what are the necessary and sufficient conditions for existing a protocol on a model tolerating certain type of adversary. We have also described the different parameters for the protocols.

In the subsequent chapters, we will look into some existing well known $PSMT$ protocols tolerating $\mathcal{A}_{tb}^{static}$ and do their complete analysis.

Chapter 2

Perfectly Secure Message Transmission Protocols

Perfectly secure message transmission($PSMT$) protocols were first presented by Dolev, Dwork, Waarts and Yung [1] in 1993. They first proposed an $O(t)$-round protocol and finally they developed a 3-round protocol. Unfortunately, none of these protocols are optimal from the point of view of round complexity and communication complexity. In this chapter, we will discuss these protocols and analyze their properties.

2.1 An $O(t)$ Round Protocol

2.1.1 Informal Description

First, we will provide an informal description and working procedure of this protocol. It can have at most $2t + 1$ rounds. From the *Theorem 1.9.1*, we need at least $2t + 1$ channels for the existence of a multi round protocol against A_t^{static}. We assume that, we have $n = 2t + 1$ channels between the sender(S) and the receiver(R). The aim of this protocol is to establish a *secure random pad* p^s between the sender and the receiver. If we are able to do that, then we can Xor p^s with the message m and broadcast it. Since the receiver will have the same pad, it will get the message easily. And since p^s is secure , m will be secure.

To establish the random pad, the sender chooses a random t-degree polynomial $f^s(x)$ from $Z_q(x)$ (where $q > n$ and $Z_q(x)$ is the set of all polynomials whose coefficients are in the finite field Z_q) and computes $a_i^s = f^s(i)$ for $i = 1, 2, 3, ..., n$ and sends a_i^s over channel w_i. The sender also sets $p^s = f^s(0)$.

Due to the effect of the adversary A_t^{static}, there may be some changes in the values at the time of transmission over the channels. Suppose, the receiver receives a_i^r over each channel w_i. The flow of information is shown in *Figure 2.1*. Now, if the receiver can get back a t-degree polynomial from the receiving

values, then it can use the constant term of that polynomial as the required random pad. The receivers tries to interpolate the n values (i, a_i^r). If the receiver recovers a t-degree polynomial $f^r(x)$, it sets $f^r(0)$ as the required pad and broadcasts an "acknowledgement message" to the sender telling that it has received the correct pad. Otherwise, the receiver will broadcast $(a_1^r, a_2^r, ..., a_n^r)$. Notice that in this case, at least one channel has delivered wrong value. But the receiver can not identify the channel. That is why it broadcasts back everything to the sender. The sender, on comparing these values with the original values will identify all such corrupted channels w_i, for which $a_i^r \neq a_i^s$. The sender will remove these identified corrupted channels and repeat the same method on the remaining channels by decreasing the degree of the random polynomial $f^s(x)$ by the number of identified corrupted channels. There are at most t channels under the control of the adversary. Now, if the adversary changes only one value (which is the worst case otherwise receiver will reconstruct the polynomial) of a polynomial in a certain round, then in the next round only one corrupted channels will be identified. So, it may take at most $2t$ rounds to identify all the corrupted channels. So, this protocol can continue for at most $2t + 1$ rounds.

Figure 2.1: Data Flow Over the n channels During **Round I** of $O(t)$ Round Protocol.

channel	S Sends	R Receives
w_1	a_1^s	a_1^r
w_2	a_2^s	a_2^r
....	....	....
w_i	a_i^s	a_i^r
....	....	....
w_n	a_n^s	a_n^r

2.1.2 Formal Description of the Protocol

Formal description of the protocol is given in the *Figure 2.2*. We now prove the properties of the protocol.

Lemma 2.1.1 *The protocol in Figure 2.2 will require at most* $2t + 1$ *rounds to terminate.*

Proof The adversary can control at most t channels. Now, if the adversary changes only one value (which is the worst case otherwise, receiver will reconstruct the polynomial) of a polynomial sent by the sender in a certain round, then in the next round only one corrupted channels will be identified. So, it may take at most $2t$ rounds to identify all the corrupted channels as to identify one corrupted channels, we may require 2 rounds. And another round is required foe sending the message. So, this protocol can continue for at most $2t + 1$ rounds.

Figure 2.2: O(t) round PSMT Tolerating $\mathcal{A}_{tb}^{static}$

1. $W = \{w_1, w_2, ..., w_n\}$;
2. number of identified corrupted channels p=0;
3. b=1,j=1;

repeat
Round j: S to R:

If j=1, S starts from step 4.

1. S checks if $a_i^r = a_i^s$ or not and removes the channel w_i for which $a_i^r \neq a_i^s$ from the set W. Let the number of removed channels be k.
2. S computes p=p+k and relabels the channels in W as $\{w_1, w_2, ..., w_{n-p}\}$
3. S broadcasts the set W.
4. S chooses a random $(t-p)$ degree polynomial $f_b^s(x) \in Z_q(x)$.
5. S calculate $a_i^s = f_b^s(i)$ for $i = 1, 2, 3, ..., (n-p)$ and sets random pad $p_b^s = f_b^s(0)$.
6. S sends a_i^s over channel $w_i \in W$.

Round j+1: R to S:

Suppose, R receives a_i^r over channel w_i.

1. R tries to interpolate received $n-p$ values and do the followings
   (a) If R gets a $t-p$ degree polynomial $f_b^r(x)$, it sets $p_b^r = f_b^r(0)$ and broadcasts an "acknowledgement message".
   (b) Otherwise, R broadcasts $(a_1^r, a_2^r, ..., a_{n-p}^r)$.

j=j+2,b=b+1.
Until (R sends an "acknowledgement message")

Round j: S to R:

1. S broadcasts $z = p_b^s \oplus m$.

Message Recovery by R:

R will recover the message by computing $z \oplus p_b^r$.

Lemma 2.1.2 (Correctness) *In the protocol described in Figure 2.2, the receiver will always be able to correctly recover the message.*

Proof The receiver sends an "acknowledgement message" if it reconstructs a polynomial whose degree is same as the degree of the polynomial(it is known to the receiver as sender has broadcasted W) sent by the sender during that round. This should happen when adversary does not modify any values over the channels controlled by it. It can take at most $2t$ round to get such a polynomial. After getting the "acknowledgement message" the sender uses the constant term (i.e.p_b^s) of the last random polynomial chosen by it for xoring with the message. Since, the receiver will also reconstruct the same polynomial (so, $p_b^s = p_b^r$), it will recover the message correctly.

Lemma 2.1.3 (Perfect Secrecy) *In the protocol described in Figure 2.2, the message m will be information theoretically secure from $\mathcal{A}_t^{static}$.*

Proof To prove the perfect secrecy of the protocol, we have to show that the pad p_b^s remains secure. In the $(2i-1)$-th round(for $i \leq t$) adversary can get $(t-p)$ number of values on $f^s(x)$ of degree $(t-p)$. So, adversary will not be able to recover p^s because it requires $(t-p+1)$ values to recover a polynomial of degree $(t-p)$. In $2i$-th round, the receiver either broadcasts an "acknowledgement message" or broadcasts the received $(n-p)$values. In the first case, the pad remains secure as adversary will not be able to construct $f^s(x)$ and in the second case, the sender will randomly choose a polynomial of required degree independent of the previously chosen polynomials. So, this protocol provides perfect secrecy.

Lemma 2.1.4 (Communication Complexity) *The total communication complexity of the protocol described in Figure 2.2 is* $O(n^3)$.

Proof We will prove the total communication complexity in two steps. First, we will show that the communication complexity of each round can be $O(n^2)$. In the first round the sender sends one value over each channel. This implies communication complexity is $O(n)$ during the first round. In the second round communication complexity can be maximum $O(n^2)$ since the receiver may broadcasts n values over each channel or broadcasts an "acknowledgement message". Communication complexity of the third round is $O(n)$ if the sender broadcasts z and $O(n^2)$ if it broadcasts the set W. Since in the other subsequent rounds the sender and the receiver repeat the same process with less number of channels, the communication complexity of each round will be $O(n^2)$.

This protocol can continue for $2t+1 = O(n)$ rounds. So, the total communication complexity is $O(n^3)$.

Theorem 2.1.5 *The protocol in Figure 2.2 is a PSMT protocol. The Protocol requires at most* $2t+1$ *rounds and has a communication complexity of* $O(n^3)$.

Proof Proof of this theorem follows from *Lemma 2.1.1, Lemma 2.1.2, Lemma 2.1.3, Lemma 2.1.4.*

2.2 3-Round Protocol

The main drawback of the protocol in *Figure 2.2* is that the number of rounds of the protocol is $O(t)$. In this section, we will describe a protocol which overcomes this problem. This protocol successfully terminates after three rounds and it can also work against a mobile adversary.

2.2.1 Informal Description

To reduce the number of rounds of the previous protocol, sufficient number of random pads will be sent during the first round so that the receiver can recover at least one pad correctly. To do this, the sender chooses $nt+1$ random

t-degree polynomials $f_i^s(x) \in Z_q(x)$(we will call these polynomials as primary polynomials) and sets pads $p_i^s = f_i^s(0)$. Now, for each primary polynomial $f_i^s(x)$, the sender chooses another n degree t secondary polynomials $h_{ij}^s(x)$ satisfying $h_{ij}^s(0) = f_i^s(j)$ for $j = 1, 2...., n$. For each pad p_i^s, the sender sends a secondary polynomial $h_{ij}^s(x)$ over each channelw_j and $(a_{i1j}^s, a_{i2j}^s, ..., a_{ijj}^s, .., a_{inj}^s)$ over channelw_j, where $a_{ijk}^s = h_{ij}^s(k)$.

Suppose, for the i-th pad the receiver receives $h_{ij}^r(x)$ for $j = 1, 2...., n$ and a_{ijk}^r for $j = 1, 2, ...n, k = 1, 2...., n$. The flow of information for the i-th pad is shown in *Figure 2.3*. If, degree of some polynomials are not t, the receiver can ignore that channels. The receiver tries to interpolate $(j, h_{i,j}^r(0))$ for $i = 1, 2, ..., nt+1$, if it gets at least one degree t polynomial $g_i^r(x)$ for some i, then the receiver sets the constant term of that polynomial as the pad p_i^r. The adversary knows at most t values on $g_i^r(x)$. So the pad p_i^r remains secure and can be used for encryption of message. If one such pad exists, the receiver broadcasts an "acknowledgement message" and corresponding index i. Otherwise, the receiver constructs $nt+1$ *conflict graphs* $G_i = (V_i, E_i)$ corresponding to each pad, where the vertex set V_i contains n vertices v_i corresponding to each channel w_i and edge set E_i which is an order set, contains an directed edge (v_k, v_j) if $a_{ijk}^r \neq h_{ij}^r(k)$. The receiver then finds an index l such that the conflict graph $G_{l=}(V, E)$ is the subgraph of the union of the remaining nt graphs(union of the graphs has the same vertex set and there is a directed edge between two vertices if this edge exists in any one of the graphs). After this, the receiver broadcasts index l and all the all the received values except the values corresponding to pad l.

During the third round, the sender broadcasts $z = p_i^s \oplus m$ if it received an "acknowledgement message" and an index i in the second round. Otherwise, it compares the received values with corresponding original values and gets a list of faulty channels over which these two values are not same. The sender broadcasts the faulty list and $z = p_l^s \oplus m$. For the first case, the receiver already has the required pad. So, the receiver will get the message. In the second case, using the faulty list the receiver will reconstruct the polynomial $f_l^s(x)$ correctly and will get the message.

Figure 2.3: Data Flow for one pad Over the n channels During **Round I** of 3-Round Protocol.

channel	S Sends for pad p_i^s	R Receives
w_1	$h_{i1}^s(x), a_{i11}^s, a_{i21}^s, ..., a_{in1}^s$	$h_{i1}^r(x), a_{i11}^r, a_{i21}^r, ..., a_{in1}^r$
w_2	$h_{i2}^s(x) a_{i12}^s, a_{i22}^s, ..., a_{in2}^s$	$h_{i2}^r(x) a_{i12}^r, a_{i22}^r, ..., a_{in2}^r$
....	....	....
w_i	$h_{ii}^s(x) a_{i1i}^s, a_{i2i}^s, ..., a_{ini}^s$	$h_{ii}^r(x) a_{i1i}^r, a_{i2i}^r, ..., a_{ini}^r$
....	....	....
w_n	$h_{in}^s(x), a_{i1n}^s, a_{i2n}^s, ..., a_{inn}^s$	$h_{in}^r(x), a_{i1n}^r, a_{i2n}^r, ..., a_{inn}^r$

2.2.2 Description of the Protocol

Formal description of the protocol is given in the *Figure 2.4*. We now prove the properties of the protocol.

Figure 2.4: 3 round PSMT Tolerating $\mathcal{A}_t^{static}$

Round I: S to R:

1. For $i = 1, 2, ..., nt + 1$ Sender
   (a) chooses a random t-degree polynomial $f_i^s(x) \in Z_q(x)$.
   (b) chooses another n degree t polynomials $h_{i,j}^s(x)$ for $j = 1, 2....., n$ satisfying $h_{i,j}^s(0) = f_i^s(j)$.
2. Sender sets $p_i^s = f_i^s(0)$ for $i = 1, 2, ..., nt + 1$.
3. Computes $a_{i,j,k}^s = h_{i,j}^s(k)$ for $i = 1, 2, ..., nt + 1, j = 1, 2, ...n, k = 1, 2...., n$.
4. For each pad p_i^s, sender sends
   (a) $h_{i,j}^s(x)$ over channel w_j.
   (b) $(a_{i1j}^s, a_{i2j}^s, ..., a_{ijj}^s, .., a_{inj}^s)$ over channel w_j.

Round II: R to S:

Suppose, for i-th pad, R receives $h_{i,j}^r(x)$ and $(a_{i1j}^r, a_{i2j}^r, ..., a_{ijj}^r, .., a_{inj}^r)$ over channelw_j

1. For $i = 1, 2, ..., nt+1$ the receiver tries to interpolate $(j, h_{i,j}^r(0))$ for $j = 1, 2, ..., n$.
   (a) If it can recover at least one t-degree polynomial $g_i^r(x)$, then it broadcasts an "acknowledgement message" and the corresponding index i.
   (b) The receiver computes $p_i^r = g_i^r(0)$.
2. If R does not get any such $g_i^r(x)$, R constructs conflict graphs $G_i = (V_i, E_i$ for $i = 1, 2, ..., nt + 1$ where the vertex set V_i contains n vertices v_i corresponding to each channel w_i and edge set E_i which is an order set, contains an directed edge (v_k, v_j) if $a_{ijk}^r \neq h_{ij}^r(k)$. Then it finds an index l such that $G_l = (V, E)$ is the subgraph of the union of the remaining nt graphs. R then broadcasts the index l and all the received values except the values for the l-th pad.

Round III: S to R:

1. If the sender gets an "acknowledgement message" and index i, then it broadcasts $z = p_i^s \oplus m$.
2. Otherwise, the sender puts
   (a) a channel j in the "faulty list" if $h_{i,j}^r(x) \neq h_{i,j}^r(x)$ and
   (b) channel k in the "faulty list" if $a_{i,j,k}^s \neq a_{i,j,k}^r$.
3. The sender broadcasts the "faulty list" and $z = p_l^s \oplus m$.

Message Recovery by R:

1. If R has sent the "acknowledgement message" in second round, R has the correct pad. So, it will recover the message by computing $z \oplus p_i^r$.
2. Otherwise, R receives the "faulty list" reliably. Using the "faulty list" R will be able to reconstruct the polynomial $f_l^s(x)$ correctly and R gets the message by computing $z \oplus f_l^s(0)$.

Lemma 2.2.1 (Correctness) *In the protocol described in Figure 2.4, the receiver will always be able to correctly recover the message.*

Proof To prove the correctness of the protocol, we have to prove that the message will be recovered correctly. During the second round, if the receiver can interpolate $(j, h^r_{i,j}(0))$ for $j = 1, 2, ..., n$ to a degree t polynomial $g^r_i(x)$ for any i, then the constant term of this polynomial is same as the constant term of $f^s_i(x)$. Because if the adversary makes some changes in the values over some channels the degree of the reconstructed polynomial will not remain same as t. As, in this case, the sender will Xor the pad p^s_i with the message m, the receiver will recover the correct message by using $g^r_i(x)$.

Now we will show that if the receiver constructs the conflict graphs, then there will be one $G_i = (V, E)$ which will be subgraph of the union of the remain conflict graphs. If the adversary changes all the values over a channel controlled by it, there will be n conflicts. We know adversary can control at most t channels and no two uncorrupted channels can conflict each other. So, for t corrupted channels there can be nt conflicts. So, the conflict graphs can have at most nt possible edges. There are $nt+1$ conflict graphs, so by pigeon hole principle there will be a conflict graph $G_i = (V_i, E_i)$ which will be the subgraph of the union of the reaming conflict graphs. In the third round, the sender will identify all the corrupted channels and broadcasts the "faulty list". There are at least $t+1$ uncorrupted channel and $t + 1$ values on a t degree polynomial is sufficient for constructing it. So, removing the values corresponding to the fault channels, the receiver will recover the i-th pad correctly. So, the receiver will get back the message correctly.

Lemma 2.2.2 (Perfect Secrecy) *In the protocol described in Figure 2.4, the message m will be information theoretically secure from $\mathcal{A}^{static}_t$.*

Proof To prove the perfect secrecy of the protocol we have to show that the adversary does not get any information about the message. During the second round, if the receiver can interpolate $(j, h^r_{i,j}(0))$ for $j = 1, 2, ..., n$ by a degree t polynomial $g^r_i(x)$ for any i, then the constant term of this polynomial remains secure because adversary can get at most t values on this polynomial. Hence, the adversary will not be able to reconstruct the degree t polynomial. So, the adversary will not be able to recover the message as $g^r_i(0)$ will be used to xor with the message in third round.

Now, if the receiver constructs the conflict graphs, it will not broadcasts the values corresponding to the pad which will be used to mask the message. The adversary will not be able to recover this pad as it has at most t values for the t degree polynomial from which this pad is created. So, in both the cases, the message will remain secure from $\mathcal{A}^{static}_t$.

Lemma 2.2.3 (Communication Complexity) *The total communication complexity of the protocol described in Figure 2.4 is $O(n^4)$.*

Proof To send a polynomial of degree t, the sender need to send it's $t+1$ coefficients. In the first round, the sender has sent $(nt+1)n(t+1)+(nt+1)n^2$ values which is of $O(n^3)$. During second round, to broadcast $(nt)n(t+1)+(nt)n^2$ values the receiver have to send $O(n^4)$ values. In the third round, if sender broadcasts the "faulty list" along with the message, it will send $tn+n$ values. So, the communication complexity of the protocol is $O(n^4)$.

Theorem 2.2.4 *The protocol in Figure 2.4 is a PSMT protocol. The Protocol requires* 3 *rounds and has a communication complexity of* $O(n^4)$.

Proof Proof of this theorem follows from Lemma 2.2.1, Lemma 2.2.2, Lemma 2.2.3.

2.3 Conclusion

Now, we summarize the two protocols discussed in this chapter in *Figure 2.2* and *Figure 2.4* in *Figure 2.5*.

Figure 2.5: Summary of the above two protocols.

protocol	*no. of rounds*	*connectivity*	*CC*	ℓ
$O(t)$ *rounds protocol*	$r = 2t+1$	$n \geq 2t+1$	$O(n^3)$	1
3 *rounds protocol*	$r = 3$	$n \geq 2t+1$	$O(n^4)$	1

From the above figure, we can conclude that none of the protocols are optimal as their communication complexity and the number of sent messages do not satisfy the lower bound of the communication complexity of a multi-round protocol. Also, we can not consider them as efficient as the first protocol may require $2t+1$ rounds and communication complexity of the second protocol is very high. In the following chapter, we will present some efficient protocols.

Chapter 3

Efficient Perfectly Secure Message Transmission Protocols

In the previous chapter, we have described a 3-round protocol with communication complexity $O(n^4)$. Here, we will discuss two protocols which have less communication complexity in compare to the protocol in *section 2.3*. However, they are also not optimal. These two protocols are presented by Sayeed and Abu-Amara [4] in 1996. First of them is a 3-round protocol and second one is a 2-round protocol.

3.1 3-Round Protocol

For this protocol, we assume that we have $n = 2t + 1$ channels and t of these channels can be controlled by the adversary in Byzantine fashion.

3.1.1 Informal Description

As it is a 3 rounds protocol the sender(S) starts the protocol by choosing n random t degree polynomials $f_i^s(x)$ from $Z_q(x)$ and sets the random pads as $p_i^s = f_i^s(0)$. Over each channel w_i the sender sends $f_i^s(x)$ and values of the all polynomials at i ,that is, $a_{ji}^s = f_j^s(i)$ for $j = 1, 2, ..., n$. Due to the effect of the adversary there may be some changes in the values received by the receiver(R).

Suppose, R receives $f_i^r(x)$ and a_{ji}^r for $j = 1, 2, ..., n$ over each channel w_i. The flow of information is shown in *Figure 3.1*. Now, the receiver constructs an directed graph $G = (V, E)$ where V contains n vertices v_i corresponding to each channel w_i and E contains an directed edge (v_i, v_j) if $a_{ji}^r = f_j^r(i)$. The receiver counts the out-degree of each vertices. Since, there are at least $t+1$ uncorrupted

channels, if a polynomial is received correctly, then the corresponding vertex in G will have out-degree $t+1$ or more. The receiver removes the vertices and adjacent edges if the out-degree of the corresponding vertices is less than $t+1$. This process continues until all the vertices in the modified graphs have out-degree at least $t+1$. Now, the receiver creates two lists- $list_A$ and $list_B$. $list_B$ contains all the channels corresponding to the vertices which are present in the final graph and $list_A$ contains the remaining channels. Clearly, the channels in $list_A$ are faulty as they have delivered wrong polynomials (because if these polynomials were delivered correctly, they would have been consistent with the values over $t+1$ uncorrupted channels) and some of the channels in $list_B$ may be faulty because the adversary can change a polynomial in such a way that the changed polynomial will satisfy the values over t uncorrupted channels. So, if we consider the channel itself deliver the consistent value for the changed polynomial, it will satisfy $t+1$ channels. Corresponding to each channel w_i in $list_B$, the receiver crates $list_i$ and $value\_list_i$. $list_i$ contains the vertices v_j if the edge (v_i, v_j) is not present in the final graph and $value\_list_i$ contains the values on channel w_j for all v_j in $list_i$ which has been received corresponding to i-th polynomial. The receiver broadcasts the $list_A$ and all $list_i$ and $value\_list_i$.

After receiving all the values reliably, the sender creates a "faulty-channel-list" which contains all the channels of $list_A$. For each $value\_list_i$, the sender checks with the original values it had sent in first round. If the values are different, then the sender adds the channels on which the values are different in "faulty channel list". Otherwise, the sender puts channel w_i in "faulty-channel-list". The sender creates a "probable-correct-channel-list " which contains all the channels except the channels in "faulty channel list". The "probable-correct-channel-list" contains at least $t+1$ channels as at most t channels will be identified as faulty by the sender. Clearly, this list will contain all the uncorrupted channels. The property of any channel in "probable-correct-channel-list" is that it has delivered the polynomial and the values of the other polynomials corresponding to the channels in "probable-correct-channel-list" correctly. Now, the sender computes the exclusive-or of message m with all p_i^s(let this value be z) for the channels w_i in "probable-correct-channel-list". The sender broadcasts "probable-correct-channel-list" and z.

Upon receiving the "probable-correct-channel-list", the receiver will be able to correctly reconstruct the polynomials sent over these channels during the first round by considering their values over the channels of "probable-correct-channel-list". So, the receiver will be able to recover the message by xoring the constant term of the polynomial over the channels in "probable-correct-channel-list" with z.

3.1.2 Description of the Protocol

Formal description of the protocol is given in the Figure 3.2. We now prove the properties of the protocol.

Figure 3.1: Data Flow Over the n channels During **Round I** of 3-Round Protocol.

channel	S Sends	R Receives
w_1	$f_1^s(x), a_{11}^s, a_{21}^s, ..., a_{n1}^s$	$f_1^r(x), a_{11}^r, a_{21}^r, ..., a_{n1}^r$
w_2	$f_2^s(x) a_{12}^s, a_{22}^s, ..., a_{n2}^s$	$f_2^r(x) a_{12}^r, a_{22}^r, ..., a_{n2}^r$
....	....	....
w_i	$f_i^s(x) a_{1i}^s, a_{2i}^s, ..., a_{ni}^s$	$f_i^r(x) a_{1i}^r, a_{2i}^r, ..., a_{ni}^r$
....	....	....
w_n	$f_n^s(x), a_{1n}^s, a_{2n}^s, ..., a_{nn}^s$	$f_n^r(x), a_{1n}^r, a_{2n}^r, ..., a_{nn}^r$

Lemma 3.1.1 (Correctness) *In the protocol described in Figure 3.2, the receiver will always be able to correctly recover the message.*

Proof To prove the correctness of the protocol, we have to show that $p_i^r = p_i^s$ for all the pads over the channels in "probable-correct-channel-list". Because if they are same the message will be recovered correctly as $m = m \oplus x \oplus x$ for some $x \in Z_q$. To reconstruct a polynomial of degree t, we require $t+1$ points on the polynomial. As the "probable-correct-channel-list" contains at least $t+1$ channels, we will get at least $t+1$ values of the polynomials which are to be reconstructed. Also all these values are original correct values. So, all the reconstructed polynomials over the channels in "probable-correct-channel-list" are the same as the polynomial sent by the sender over the channels in "probable-correct-channel-list". So, for these channels $p_i^r = p_i^s$. Hence, the message will be recovered correctly.

Lemma 3.1.2 (Perfect Secrecy) *In the protocol described in Figure 3.2, the message m will be information theoretically secure from $\mathcal{A}_{tb}^{static}$.*

Proof To prove the perfect secrecy of this protocol, we need to prove that at least one pad p_i^s corresponding to a channel in "probable-correct-channel-list" remains secure against $\mathcal{A}_{tb}^{static}$. For the $t+1$ uncorrupted channels in "probable-correct-channel-list", the adversary get t values on the polynomials over these channels. These t values are not sufficient to construct a polynomial of degree t. Also the value-list(i) does not reveal any new information as these values are known to the adversary. So, all the $t+1$ pads corresponding to the uncorrupted channels remain secure. Hence, the adversary will not be able to recover the message.

Lemma 3.1.3 (Communication Complexity) *The total communication complexity of the protocol described in Figure 3.2 is $O(n^3)$.*

Proof To send a polynomial of degree t, we need to send it's $t+1$ coefficients. In the first round, the sender sends n polynomials and n values of each polynomial. So, the complexity of the first round is $n(t+1)+n^2 = O(n^2)$. During the second

Figure 3.2: 3-Round PSMT Tolerating $\mathcal{A}_{tb}^{static}$

Round I: S to R:

1. For $i = 1, 2, ..., n$, the sender randomly chooses polynomials $f_i^s(x)$ of degree t and sets the random pads as $p_i^s = f_i^s(0)$.
2. For $i = 1, 2, ..., n$, the sender sends polynomial $f_i^s(x)$ and $a_{ji}^s (j = 1, 2, .., n)$ over channel w_i.

Round II: R to S:
Suppose, R receives $f_i^r(x)$ and a_{ji}^r for $j = 1, 2, ..., n$ over channel w_i for $i = 1, 2, ...n$.

1. R constructs a directed graph $G = (V, E)$ where V contains n vertices v_i corresponding to each channel w_i and E contains an directed edge (v_i, v_j) if $a_{ji}^r = f_i^r(j)$.
2. R removes a vertex and it's adjacent edges if the out-degree of the vertex is less than $t + 1$. R recursively does this until all the reaming vertices have out-degree greater than $t + 1$. Let the final graph be H.
3. R creates two lists $list_A$ and $list_B$. $list_B$ contains all the channels corresponding to the vertices which are present in the graph H and $list_A$ contains the remaining channels.
4. For each channel w_i in $list_B$, the receiver crates $list_i$=$\{v_j|(v_i, v_j)$ not in H$\}$.
5. For each channel w_i in $list_B$ receiver crates $value\_list_i$=$\{a_{i,j}^r|v_j$ in list(i)$\}$.
6. The receiver broadcasts the $list_A$ and all the $list_i$ and $value\_list_i$.

Round III: S to R:

1. The sender creates "faulty-channel-list" containing all the channels of $list_A$.
2. For all $value\_list_i$, the sender checks if $a_{ij}^r = a_{ij}^s$ or not for all v_j in $list_i$.
   (a) if they are same R puts channel w_i in "faulty-channel-list".
   (b) otherwise, puts channel w_j in "faulty-channel-list".
3. The sender creates a "probable-correct-channel-list" containing all the channels except those are in "faulty-channel-list".
4. The sender computes Exclusive-or of the message m and all the p_i^s for the channels w_i in "probable-correct-channel-list". Let this value be z.
5. The sender broadcasts "probable-correct-channel-list" and z.

Message Recovery:

The receiver reconstructs the polynomials over the channels in "probable-correct-channel-list" and computes the corresponding pads p_i^r. The receiver then recovers the message by Exclusive-oring all these pads with z.

round the receiver broadcasts $list_A$ which can have at most t elements and there can have at most n $value\_list_i$ each of which can contain at most t elements. So, complexity of this round is $nt + n(t)n = O(n^3)$. In the third round, the sender broadcasts the "probable-correct-channel-list" and z. So, it may have to send total $n^2 + n$ values. So, the total communication complexity of this protocol is $O(n^3)$.

Theorem 3.1.4 *The protocol in Figure 3.2 is a PSMT protocol. The Protocol requires* 3 *rounds and has a communication complexity of* $O(n^3)$.

Proof Proof of this theorem follows from *Lemma 3.1.1, Lemma 3.1.2, Lemma 3.1.3.*

3.2 2-Round Protocol

3.2.1 Informal Description

This protocol is similar to the 3-Round protocol described in *section 3.1.* Only difference is that it is a two round protocol and the receiver starts the process. The receiver chooses n random t degree polynomials $f_i^r(x) \in Z_q(x)$ and sets the random pads as $p_i^r = f_i^r(0)$. Over each channel w_i the receiver sends $f_i^r(x)$ and values of the other polynomials at i, that is, $a_{ji}^r = f_j^r(i)$ for $j = 1, 2, ..., n$.

Due to the effect of the adversary there may be some change in the values received by the sender. Suppose, S receives $f_i^s(x)$ and a_{ji}^s for $j = 1, 2, ..., n$ over each channel w_i. The flow of information is shown in *Figure 3.3.* Now, the sender constructs an directed graph $G = (V, E)$ where V contains n vertices v_i corresponding to each channel w_i and E contains an directed edge (v_i, v_j) if $a_{ji}^r = f_i^r(j)$. The sender counts the out-degree of each vertices. Since, there are at least $t + 1$ uncorrupted channels, if a polynomial is received correctly, then the corresponding vertex in G will have out-degree $t + 1$ or more. The sender removes a vertex and it's adjacent edges if the out-degree of that vertex is less than $t + 1$. This process continues until all the vertices in the remaining graphs have out-degree $\geq t + 1$. Let the final graph be $H = (V^1, E^1)$. Now, the sender creates two lists: $list_A$ and $list_B$. $list_B$ contains all the channels corresponding to the vertices which are present in H and $list_A$ contains remaining channels. Clearly, the channels in $list_A$ are faulty as they have delivered wrong polynomials (because if these polynomials were delivered correctly, they would have been consistent with the values over $t + 1$ uncorrupted channels) and some of the channels in $list_B$ may be faulty because the adversary can change a polynomial in such a way that the changed polynomial will satisfy the values over t uncorrupted channels. So, if we consider the channel itself deliver the consistent value for the changed polynomial, it will satisfy $t+1$ channels. Corresponding to each channel w_i in list-B, the receiver crates $list_i$ and $value\_list_i$. $list_i$ contains the vertices v_j if the edge (v_i, v_j) is not present in the final graph and $value\_list_i$ contains the values on channel w_j for all v_j in $list_i$ which has been received corresponding to i-th polynomial.

Since, the sender have to send the message in this round, it creates a message carrying polynomial $F(x)$ which is of degree n. The coefficients of $F(x)$ are as follows- coefficient of x^0 is the message m, coefficient of x^i is zero if channel w_i is in $list_A$, coefficient of x^i is $p_i^s = f_i^r(0)$ for the channels in $list_B$. Let $b = min[t,$(number of channels in $list_B$-$(t + 1))]$. Here, 'b' is the number of channels which is in $list_B$ and may be faulty. The sender does not know the identity of these channels and uses the constant term of the polynomials

corresponding to these channels (which is in $list_B$ and may be faulty) as the coefficients of some x^i in $F(x)$. These coefficients will remain unknown to the receiver. So, total $b+1$ coefficients (b+ the constant term of $F(x)$) are unknown to the receiver. So, the sender broadcasts the $list_A$ and all the $list_i$ and $value\_list_i$ and $(F(1), F(2), ..., F(b+1))$.

Upon receiving these values reliably, the receiver creates a "faulty-channel-list" which contains all the channels of $list_A$. For each $value\_list_i$, the receiver checks with the corresponding original values it sent in first round. If the values are different, then the receiver puts that channel in "faulty-channel-list" otherwise, puts the channel w_i in "faulty-channel-list". The receiver now tries to reconstruct the message carrying polynomial $F(x)$. It takes the coefficient $F(x)$ as follows- coefficient of x^0 as unknown, coefficient of x^i as zero if channel w_i is in $list_A$, coefficient of x^i as p_i^r if channel w_i is absent in both $list_A$ and "faulty-channel-list", coefficient of x^i as unknown if channel w_i is not in $list_A$ but in "faulty-channel-list". There can be maximum $b+1$ unknown. So, the receiver will reconstruct the polynomial $F(x)$ using $(F(1), F(2), ..., F(b+1))$ and recover the message m.

Figure 3.3: Data Flow Over the n channels During **Round I** of 2-Round Protocol.

channel	R Sends	S Receives
w_1	$f_1^r(x), a_{11}^r, a_{21}^r, ..., a_{n1}^r$	$f_1^s(x), a_{11}^s, a_{21}^s, ..., a_{n1}^s$
w_2	$f_2^r(x), a_{12}^r, a_{22}^r, ..., a_{n2}^r$	$f_2^s(x), a_{12}^s, a_{22}^s, ..., a_{n2}^s$
....	....	....
w_i	$f_i^r(x), a_{1i}^r, a_{2i}^r, ..., a_{ni}^r$	$f_2^s(x), a_{1i}^s, a_{2i}^s, ..., a_{ni}^s$
....	....	....
w_n	$f_n^r(x), a_{1n}^r, a_{2n}^r, ..., a_{nn}^r$	$f_n^s(x), a_{1n}^s, a_{2n}^s, ..., a_{nn}^s$

3.2.2 Description of the Protocol

Formal description of the protocol is given in the Figure 3.2. We now prove the properties of the protocol.

Lemma 3.2.1 (Correctness) *In the protocol described in Figure 3.4, the receiver will always be able to correctly recover the message.*

Proof We will now prove that the message will be recovered by the receiver correctly. To prove it, we need to show that the receiver will be able to reconstruct the polynomial $F(x)$ correctly. To reconstruct $F(x)$, we have to know the coefficients. During the message recovery, the receiver will be able to identify all the channels over which the polynomials were delivered wrongly. Because either in the conflict graph the corresponding vertices have out-degree$\leq t$ (which

will be immediately included in $list_A$) or $list_i$ of these channels will contain at least one uncorrupted channel. So, when the receiver compares the values of $value\_list_i$ with the original values, it will identify the channels. The constant term of these polynomials (which are the coefficients of $F(x)$) are unknown to the adversary. We have seen in *section 3.2.1* the number of the channels which will be identified by R during the message recovery is b. So, there will be total $b+1$ unknowns and the receiver has $b+1$ values of $F(x)$. Using these values the receiver will reconstruct $F(x)$ correctly and get the message correctly.

Lemma 3.2.2 (Perfect Secrecy) *In the protocol described in Figure 3.4, the message m will be information theoretically secure from $\mathcal{A}_{tb}^{static}$.*

Proof To prove the perfect secrecy of the protocol, we need to show that the constant term of degree n polynomial $F(x)$ remains secure. The adversary can compute t coefficients for the t polynomials sent over the channels controlled by it. For the other polynomial it can get maximum t values on them. So, the adversary will not be able to reconstruct them. Also the adversary does not get any additional information from the $value\_list_i$. As the sender broadcasts $t+1$ values of $F(x)$, adversary can get these values. So, the adversary can get total $2t+1$ values on the polynomial $F(x)$ of degree $2t+1$. Therefore, it will not be able to reconstruct $F(x)$. Hence, the message will remain secure.

Lemma 3.2.3 (Communication Complexity) *The total communication complexity of the protocol described in Figure 3.4 is $O(n^3)$.*

Proof To send a polynomial of degree t, we need to send it's $t+1$ coefficients. In the first round, the receiver has sent n polynomials and n values of each polynomial. So, the complexity of the first round is $n(t+1)+n^2 = O(n^2)$. During the Second round, the sender broadcasts $list_A$ which can have at most t elements and there may be at most n $value\_list_i$ each of which can contain at most t elements. Also it broadcasts maximum $t+1$ values of the polynomial $F(x)$. Complexity of this round is $O(n^3)$. So, the total communication complexity of this protocol is $O(n^3)$.

Theorem 3.2.4 *The protocol in Figure 3.4 is a $PSMT$ protocol. The Protocol requires at most 3 rounds and has a communication complexity of $O(n^3)$.*

Proof Proof of this theorem follows from the *Lemma 3.2.1, Lemma 3.2.2, Lemma 3.3.3.*

3.3 conclusion

Now, we summarize the above two protocols in the *Figure 3.5.*
The three rounds protocol is a significant improvement on Dolev et al.'s three round protocol interms of communication and computation. This two rounds protocol is the first two-way SMT protocol where the cost of communication

and computation is polynomial in n. Though, the protocols of the *section 3.1* and *section 3.2* have less communication complexity in compare to the protocols described in the second chapter, these protocols are not communication optimal. In the next chapter we will mention some optimal $PSMT$ protocols tolerating $\mathcal{A}_{tb}^{static}$.

Figure 3.5: Summary of the protocols of *section 3.1* and *section 3.2*

protocol	*no. of rounds*	*connectivity*	*CC*	ℓ
3-*Round Protocol*	$r = 3$	$n \geq 2t + 1$	$O(n^3)$	1
2-*Round Protocol*	$r = 2$	$n \geq 2t + 1$	$O(n^3)$	1

Figure 3.4: 2-Round PSMT Tolerating $\mathcal{A}_{tb}^{static}$

$$W = \{w_1, w_2, ..., w_n\}$$

Round I: R to S:

1. For $i = 1, 2, ..., n$, the receiver randomly chooses polynomials $f_i^r(x) \in Z_q(x)$ of degree t and sets the random pads as $p_i^r = f_i^r(0)$.
2. For $i = 1, 2, ..., n$, the receiver sends polynomial $f_i^r(x)$ and $a_{ji}^r (j = 1, 2, .., n)$ over channel w_i.

Round II: S to R:

Suppose, S receives $f_i^s(x)$ and a_{ji}^s for $j = 1, 2, ..., n$ over channel w_i for $i = 1, 2, ...n$.

1. S constructs an directed graph $G = (V, E)$ where V contains n vertices v_i corresponding to each channel w_i and E contains an directed edge (v_i, v_j) if $a_{ji}^s = f_i^s(j)$.
2. S removes the vertex and it's adjacent edges if the out-degree of the vertex is less than $t + 1$. Recursively done this until all the remaining vertices have out-degree $\geq t + 1$. Let the final graph be $H = (V^1, E^1)$.
3. S creates two lists: $list_A$ and $list_B$. $list_B = \{w_i | v_i \in V^1\}$ and $list_A$ contains all the channels of W except those in $list_B$.
4. For each channel w_i in $list_B$, the sender crates $list_i = \{v_j | (v_i, v_j) \notin E^1\}$.
5. For each channel w_i in $list_B$, the sender crates $value\_list_i = \{a_{ij}^r | v_j \in \text{list-}(i)\}$.
6. Construct a message carrying polynomial $F(x) = \sum_{i=0}^{n} b_i x^i$ of degree n as follows-

$$b_i = \begin{cases} m & if i = 0 \\ 0 & if w_i \in list_A \\ f_i^s(0) & if w_i \in list_B \end{cases}$$

7. The sender broadcasts the $list_A$, all the $list_i$, $value\_list_i$ and $(F(1), F(2), ..., F(b + 1))$.

Message Recovery:

1. The receiver creates "faulty-channel-list"$(= list_D)$ containing all the channels of $list_A$.
2. For all $value\_list_i$, the receiver checks if $a_{ij}^r = a_{ij}^s$ or not for all v_j in $list_i$.
   (a) if they are same R puts channel w_i in "faulty-channel-list".
   (b) otherwise, puts channel w_j in "faulty-channel-list".
3. The receiver creates a "probable-correct-channel-list" containing all the channels of W except those are in "faulty-channel-list".
4. The receiver reconstructs the polynomial $F(x)$ as follows-

$$b_i = \begin{cases} unkunown & if i = 0 \\ 0 & if w_i \in list_A \\ f_i^r(0) & if w_i \in list_D \\ unkunown & if w_i \in list_D, w_i \notin list_A \end{cases}$$

The receiver has $b + 1$ points on $F(x)$ and there are $b + 1$ unknowns. So, the receiver will be able to recover the message.

Chapter 4

Communication Optimal $PSMT$ protocols

In the previous chapters, we have discussed some $PSMT$ protocols. Some of these protocols are efficient, but not communication optimal. Here, we will mention a two round and a three round communication optimal $PSMT$ protocol. Unfortunately, both of them are optimal only in the message size $l = \theta(n)$.

4.1 Optimal PSMT Protocols

4.1.1 Three Round Protocol

The first three rounds communication optimal $PSMT$ protocol tolerating $\mathcal{A}_{tb}^{static}$ was presented by A. Patra, A. Choudhary, K. Srinathan, and C. Pandu Rangan [8] in 2006. This $PSMT$ protocol securely sends a message containing $l = \theta(n)$ by communicating $O(n^2)$ field elements.

4.1.2 Two Rounds Protocol

Two rounds communication optimal $PSMT$ protocol was presented by K. Kurosawa and K. Suzuki [7] in 2008. In this paper, they show the first 2-round PSMT protocol for n = 2t+1 such that not only the communication complexity is $O(n^2)$ but also the computational costs of the sender and the receiver are both polynomial in n. Thus they solve the open problem raised by Agarwal, Cramer and de Haan[6] at CRYPTO 2006. The main disadvantage of this protocol is that it is optimal only if the message size $l = \theta(n)$.

Chapter 5

Conclusions and Open Problems

In the initial part of this report, we have discussed the various network settings and models in which the RMT/SMT protocols can be designed. In the second chapter, we have discussed two $PSMT$ protocols of [1] and analyze their properties. In the next chapter, we have presented two efficient $PSMT$ protocols of [4]. And in the fourth chapter we have mentioned some optimal $PSMT$ protocols.

Though the three round $PSMT$ protocol of [8] and the two round $PSMT$ protocol of [7] (mentioned in the fourth chapter) are communication optimal, they are optimal only if $l = \theta(n)$. This brings forth the following open problem:

Open Problem 1: *Let the sender and the receiver be connected by $n = 2t+1$ channels. Then does there exists an efficient, polynomial time multi-round $PSMT$ protocol which securely sends a message containing l field elements by communicating $O(nl)$ field elements, tolerating $\mathcal{A}_{tb}^{static}$, for any value of l.*

Another open problem is about the communication complexity of a two round $PSMT$ protocol for sending a single field elements.

Open Problem 2: *Let the sender and the receiver be connected by $n = 2t+1$ channels. Then does there exists an efficient, polynomial time two rounds $PSMT$ protocol which securely sends a message containing one field element by communicating less than $O(n^2)$ field elements, tolerating $\mathcal{A}_{tb}^{static}$.*

Bibliography

[1] D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly secure message transmission. JACM,40(1)17 − 47, 1993.

[2] K. Srinathan, N. R. Prasad, and C. Pandu Rangan. On the optimal communication complexity of multiphase protocols for perfect communication. In 2007 IEEE Symposium on Security and Privacy (S and P 2007), 20-23 May 2007, Oakland, California, USA, pages 311-320. IEEE Computer Society, 2007.

[3] M. Fitzi, M. K. Franklin, J. A. Garay, and S. H. Vardhan. Towards optimal and efficient perfectly secure message transmission. In S. P. Vadhan, editor, Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings, volume 4392 of Lecture Notes in Computer Science, pages 311-322. Springer, 2007 .

[4] H. Sayeed and H. Abu-Amara. Efficientcient Perfectly Secure Message Transmission in Synchronous Networks. Information and Communication, 126(1):5361, 1996.

[5] S. Srinathan, A. Narayanan, and C. Pandu Rangan. Optimal Perfectly Secure Message Transmission. In CRYPTO'04, pages 545561, Santa Barbara, California, 2004.

[6] S.Agarwal, R.Cramer and R.de Haan: Asymptotically Optimal Two-Round Perfectly Secure Message Transmission. CRYPTO 2006: pp.394408 (2006).

[7] K. Kurosawa and K. Suzuki. Truly efficient 2 round perfectly secure message transmission scheme. In Nigel P. Smart, editor, Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings, volume 4965 of Lecture Notes in Computer Science, pages 324-340. Springer, 2008.

[8] A. Patra, A. Choudhary, K. Srinathan, and C. Pandu Rangan. Constant phase bit optimal protocols for perfectly reliable and secure message transmission. In R. Barua and T. Lange, editors, Progress in Cryptology - INDOCRYPT 2006, 7th International Conference on Cryptology in India, Kolkata, India, December 11-13, 2006, Proceedings, volume 4329 of Lecture Notes in Computer Science, pages 221-235. Springer, 2006.

[9] D. R. Stinson. Cryptpgraphy Theory and Practice. Chapman and Hall/CRC.

[10] A. Choudhary. Protocols for Reliable and Secure Message Transmission. P.hd Thesis Paper.