

# Perfectly Secure Message Transmission Tolerating Mixed Adversary<sup>☆</sup>

Arpita Patra<sup>\*,1,2</sup>, Ashish Choudhury<sup>1,3</sup>, Ashwinkumar B. V<sup>6</sup>,  
Kannan Srinathan<sup>4</sup>, C. Pandu Rangan<sup>1,5</sup>

---

## Abstract

In this paper, we study the issues related to the *possibility*, *feasibility* and *optimality* for *perfectly secure message transmission* (PSMT) in an undirected synchronous network, under the influence of a *mixed adversary* having *unbounded computing power*, who can corrupt some of the nodes in the network in *Byzantine*, *fail-stop* and *passive* fashion respectively. Specifically, we answer the following questions: (a) **POSSIBILITY**: Given a network and a mixed adversary, what is the necessary and sufficient condition for the existence of any PSMT protocol over the network tolerating the adversary? (b) **FEASIBILITY**: Once the existence of a protocol is ensured, then does there exist a polynomial time and efficient protocol on the given network? (c) **OPTIMALITY**: Given a message of specific length, what is the minimum communication complexity (lower bound) needed by any PSMT protocol to transmit the message and how to design a polynomial time protocol whose total communication complexity matches the lower bound on the communication complexity? We answer the above questions by considering two different types of mixed adversary, namely static mixed adversary and mobile mixed adversary. Intuitively, it is more difficult to tolerate a mobile mixed adversary (who can corrupt different set of nodes during different stages of the protocol) in comparison to its static counter part (who corrupts the same set of nodes throughout the protocol). However, surprisingly, we show that the *connectivity requirement in the network and lower bound on communication*

---

<sup>☆</sup>This is an extended, modified and elaborate version of [10]

\*Corresponding author

*Email addresses:* arpitapatra10@gmail.com, arpitapatra\_10@yahoo.co.in (Arpita Patra), partho\_31@yahoo.co.in, partho31@gmail.com (Ashish Choudhury), ashwinkumarbv@gmail.com (Ashwinkumar B. V), srinathan@iiit.ac.in (Kannan Srinathan), prangan55@gmail.com, prangan55@yahoo.com (C. Pandu Rangan)

<sup>1</sup>Department of Computer Science and Engineering, IIT Madras, Chennai India 600036.

<sup>2</sup>Financial Support from Microsoft Research India Acknowledged.

<sup>3</sup>Financial Support from Infosys Technology India Acknowledged.

<sup>4</sup>Center for Security, Theory and Algorithmic Research, IIIT Hyderabad, India.

<sup>5</sup>Work Supported by Project No. CSE/05/06/076/DITX/CPAN on Protocols for Secure Computation and Communication, Sponsored by Department of Information Technology, Govt. of India.

<sup>6</sup>Dept. of Computer Science and Engineering, Cornell University, Ithaca. The work was done when the author was an undergraduate student at IIT Madras.

*complexity of PSMT protocols is same against both static and mobile mixed adversary.* To design our protocols against static and mobile mixed adversary, we use several new techniques, which are of independent interest.

*Key words:* Information Theoretic Security, Static and Mobile Mixed Adversary, Undirected Graphs, Synchronous Networks.

---

## 1. Introduction

Consider the following problem: a *sender*  $\mathbf{S}$  and a *receiver*  $\mathbf{R}$  are part of an unreliable distributed synchronous network and are connected through intermediate nodes. The distrust in the network is modelled by an entity called *adversary*, who has *unbounded computing power* and who can corrupt some of the intermediate nodes in a variety of ways.  $\mathbf{S}$  wishes to send to  $\mathbf{R}$  a message  $m$  that consists of  $\ell \geq 1$  field elements, selected uniformly from a finite field  $\mathbb{F}$ . The challenge is to design a protocol, such that after interacting with  $\mathbf{S}$  as per the protocol,  $\mathbf{R}$  should output  $m$  without any error (*perfect reliability*) and at the same time, adversary should not get any information about  $m$  what so ever (*perfect security*). Moreover, this should happen irrespective of the behavior of the adversary. This problem is known as *perfectly secure message transmission* (PSMT). Security against such a powerful adversary is also known as *non-cryptographic* or *information theoretic* or *Shannon* security. Notice that since adversary has unbounded computing power, we cannot solve PSMT problem by using classical cryptographic primitives such as public key cryptography, digital signatures, authentication schemes, etc as the security of all these primitives holds good only against an adversary having bounded computing power.

**Why to Study PSMT:** PSMT is one of the fundamental problems in secure distributed computing. There are two motivations to study PSMT problem. Many fundamental fault tolerant distributed computing primitives, such as secure multiparty computation (MPC) [56, 21, 6, 8, 42, 3, 4, 5], Byzantine Agreement (BA) [41, 15, 13, 7, 26], Verifiable Secret Sharing (VSS) [9, 6, 42, 20], etc assume that there exists a direct and secure link between every two nodes in the network. This implies that the underlying network graph is a complete graph, which is an unrealistic assumption. In the networks, where  $\mathbf{S}$  and  $\mathbf{R}$  are not adjacent, PSMT protocols help to simulate a virtual secure link between  $\mathbf{S}$  and  $\mathbf{R}$ . This way, we can simulate a virtual complete network, over which the above fault tolerant primitives can be executed.

The second motivation to study PSMT is to achieve information theoretic security. The security of all existing public key cryptosystems, digital signatures are based on the unproven hardness assumptions of certain number theoretic problems. However, the increase in computing speed and advent of new computing paradigms like Quantum computing [46] may render these assumptions very weak or useless in practice. But these factors have no effect on information theoretic security which is the strongest notion of security. Thus in a scenario,

when existing public key cryptosystems, digital signatures can not provide satisfactory security, PSMT protocols may help to provide effective alternative.

**A Taxonomy for PSMT Protocols:** The PSMT problem was first proposed and solved by Dolev et.al [14]. Dolev et.al considered an undirected synchronous network and assumed that the adversary can corrupt  $t_b$  nodes in the network in Byzantine fashion. Roughly speaking, if a node is Byzantine corrupted, then the adversary can not only listen all the information possessed by that node, but also can force the node to deviate from the protocol in any arbitrary manner. The work of Dolev et.al is followed by several other works, which considered PSMT problem in several network settings and adversarial model. For example, the underlying network model may be undirected graph [14, 45, 50, 1, 16, 36, 25], directed graph [12, 32, 37, 34] or hypergraph [19, 52]. The communication in the network could be synchronous [14, 45, 51, 25] or asynchronous [44, 11, 49]. The faults in the network could be passive, fail-stop, Byzantine or sometimes mixed/hybrid faults [10]. The number of faulty nodes in the network may be bounded by a fixed constant (threshold adversary) [14, 25] or the potential sets of faulty nodes may be described by a collection of subsets of nodes (non-threshold adversary) [23, 49, 40]. The adversary may be static [14, 23, 25, 12] or mobile [54, 39, 10, 35]. The protocol may allow a negligible error probability in reliability [18, 12, 55, 24, 2, 51, 33, 48] or may not allow any error in reliability [14, 45, 50, 1, 16, 36, 25]. We may use the following parameters to describe different settings/models for studying PSMT:

1. Type of Underlying Network — *Undirected Graph, Directed Graph, Hypergraph.*
2. Type of Communication — *Synchronous, Asynchronous.*
3. Adversary capacity — *Threshold Static, Threshold Mobile, Non-threshold Static, Non-threshold Mobile.*
4. Type of Faults — *Fail-stop, Passive, Byzantine, Mixed.*

For example, one may ask: what is the necessary and sufficient condition for PSMT over an *undirected synchronous* network thwarting a *threshold static mixed* adversary? In this way, hundreds of different models/settings can be formulated and almost all of them are used in practice.

Any PSMT protocol is analyzed by the following four parameters:

1. Connectivity of the underlying network, denoted by  $n$ ,
2. Number of phases, denoted by  $r$ , taken by the protocol. Here a phase is a communication from  $\mathbf{S}$  to  $\mathbf{R}$  or vice-versa,
3. Communication complexity denoted by  $c$ , which is the total number of field elements communicated by  $\mathbf{S}$  and  $\mathbf{R}$  in the protocol and
4. Amount of computation done by  $\mathbf{S}$  and  $\mathbf{R}$  in the protocol.

Irrespective of the settings in which PSMT is studied, the following issues are common:

- (i) POSSIBILITY: What is the necessary and sufficient condition for the existence of any PSMT protocol in a given network, tolerating a given type of adversary?

- (ii) FEASIBILITY: Once the existence of a PSMT protocol is ensured then does there exist a polynomial time efficient protocol on the given network?
- (iii) OPTIMALITY: Given a message of specific length, what is the minimum communication complexity (lower bound) needed by any PSMT protocol to transmit the message and how to design a polynomial time PSMT protocol whose total communication complexity matches the lower bound on the communication complexity?

This taxonomy and a unified framework for a number of research problems were first discussed in [47]. Different techniques are used to resolve the above issues in different settings. For example, the techniques used in designing optimal PSMT protocols in undirected networks are completely different from the ones used in directed networks.

### 1.1. Our Motivation

The issue of POSSIBILITY, FEASIBILITY and OPTIMALITY in the context of PSMT in undirected synchronous networks has been completely resolved in [14, 45, 50, 1, 16, 53, 25, 35]. However, all these works assume that the adversary can corrupt the nodes *only in Byzantine fashion*. In a typical large network, certain nodes may be strongly protected and few others may be moderately/weakly protected. An adversary may only be able to fail-stop(/eavesdrop in) a strongly protected node, while he may affect a weakly protected node in Byzantine fashion <sup>7</sup>. Thus, we may capture the abilities of an adversary in a more realistic manner, using three parameters  $t_b, t_f, t_p$  where  $t_b, t_f, t_p$  are the number of nodes under the influence of adversary in Byzantine, fail-stop and passive fashion respectively. Also it is better to grade different kinds of disruption done by adversary and consider them separately rather than treating every kind of fault as Byzantine fault as this is an “overkill” and causes an over estimation of the resources required for PSMT. A formal justification of the later statement will appear in the subsequent sections.

Motivated by the need to study mixed adversary in the context of secure message transmission, the authors in [38] have recently studied the issues related to POSSIBILITY, FEASIBILITY and OPTIMALITY in the context of reliable and secure message transmission in the presence of a mixed adversary, who can corrupt disjoint sets of  $t_b, t_f$  and  $t_p$  nodes in Byzantine, fail-stop and passive fashion respectively. However, the results presented in [38] holds only for the protocols, having a *negligible error probability in reliability*; i.e., the reliability

---

<sup>7</sup>Informally, a fail-stop corrupted node can crash at any time during the protocol execution. Moreover, once crashed it cannot become alive again. However, as long as it is alive, it will honestly follow the protocol, without leaking any information to the adversary. If a node is eavesdropped or passively corrupted then it honestly follows the protocol, but leaks full information to the adversary

is *not perfect reliability*<sup>8</sup>. As far our knowledge is concerned, nobody has ever addressed the issue of POSSIBILITY, FEASIBILITY and OPTIMALITY in the context of PSMT in undirected synchronous networks, tolerating mixed adversary. In this paper, we completely resolve the above three issues in *undirected synchronous network tolerating threshold mixed adversary*. The mixed adversary has unbounded computing power and controls disjoint sets of  $t_b$ ,  $t_f$  and  $t_p$  nodes in Byzantine, fail-stop and passive fashion respectively.

We consider two types of mixed adversary, namely *threshold static mixed adversary* and *threshold mobile mixed adversary*, denoted by  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  and  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ , respectively. Informally,  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  controls the *same* set of  $t_b$ ,  $t_f$  and  $t_p$  nodes in Byzantine, fail-stop and passive fashion respectively throughout the protocol. On the other hand,  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  may control *potentially different* set of  $t_b$ ,  $t_f$  and  $t_p$  nodes in Byzantine, fail-stop and passive fashion respectively, during different phases of the protocol. Hence if a node is corrupted by  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  in Byzantine/fail-stop/passive fashion in  $i^{th}$  phase, then it is healed at the end of that phase. So a node controlled by  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  in  $i^{th}$  phase will be honest in  $(i + 1)^{th}$  phase, unless the adversary chooses to corrupt the *same* node in  $(i + 1)^{th}$  phase as well.

**Why to Study Mobile Adversary:** If a protocol is executed for a very short duration, then it is appropriate to model the adversary as static, who corrupts the same set of nodes throughout the protocol. However, in many practical scenarios, a protocol may be executed for a longer duration, where  $\mathbf{S}$  and  $\mathbf{R}$  may interact for a long time. In such scenarios, some of the faults which are done in the earlier stages, may be identified and fixed and in the mean time, a hacker may attack some other nodes. Evidently, in such cases the mobile adversary models the fault behavior more appropriately than static adversary.

## 1.2. Organization of the Paper

Since, in this paper, we deal with both static and mobile adversary, for ease of understanding and to avoid confusion, we divide the paper into two halves. The first half deals with  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , while the second half deals with  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . At the end of each section, we give a comprehensive comparison of our results with the existing results. In the sequel, we present an elaborate literature survey, our contribution, our network and adversary model for static adversary and finally our results related to static adversary. Subsequently, the same format will be followed for mobile adversary.

---

<sup>8</sup>The authors in [38] have termed this problem as *unconditionally secure message transmission* (USMT), which is same as PSMT, except that  $\mathbf{R}$  may output an incorrect message at the end of the protocol with negligible error probability.

## 2. Network Model and Definitions Used for Static Adversary

We now describe the network model and adversary settings used for studying PSMT against static adversary. The underlying network is a connected synchronous network represented by an undirected graph, where  $\mathbf{S}$  and  $\mathbf{R}$  are two *non-adjacent* nodes of the graph (for if  $\mathbf{S}$  and  $\mathbf{R}$  are adjacent then PSMT can be solved trivially). All the edges in the network are reliable and secure but the nodes can be corrupted. We assume that there exists an adversary  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , who has *unbounded computing power* and controls *disjoint* sets of  $t_b, t_f$  and  $t_p$  nodes in Byzantine, fail-stop and passive fashion respectively. We would like to caution the reader on different ways in which the terminologies Byzantine, fail-stop and passive are used in the literature and there are minor and subtle variations among definitions of these terminologies used by different authors. To keep the discussion self contained, we give below the definitions of the above terminologies as used in this paper. The definitions are same as used in [17].

**Definition 1** (FAIL-STOP CORRUPTION [17]). *A node  $P$  is said to be fail-stop corrupted if the adversary can crash  $P$  at will at any time during the execution of the protocol. But as long as  $P$  is alive,  $P$  will honestly follow the protocol and the adversary will have no access to any information or internal state of  $P$ .*

**Definition 2** (PASSIVE CORRUPTION [17]). *A node  $P$  is said to be passively corrupted if the adversary has full access to the information and internal state of  $P$ . But  $P$  will honestly follow the protocol execution.*

**Definition 3** (BYZANTINE CORRUPTION [17]). *A node  $P$  is said to be Byzantine corrupted if the adversary fully controls the actions of  $P$ . The adversary will have full access to the computation and communication of  $P$  and can force  $P$  to deviate from the protocol in any arbitrary manner.*

Following the approach of Dolev et. al. [14], we abstract away the network and concentrate on solving PSMT problem for a single pair of processors, the *sender*  $\mathbf{S}$  and the *receiver*  $\mathbf{R}$ , connected by  $n$  parallel and synchronous bi-directional node disjoint paths/channels  $w_1, w_2, \dots, w_n$ , also called as *wires*. The reason for such an abstraction is as follows: suppose some intermediate node between  $\mathbf{S}$  and  $\mathbf{R}$  is under the control of the adversary. Then all the paths between  $\mathbf{S}$  and  $\mathbf{R}$  which passes through that node are also compromised. Hence, all the paths between  $\mathbf{S}$  and  $\mathbf{R}$  passing through that node can be modelled by a single wire between  $\mathbf{S}$  and  $\mathbf{R}$ . In the worst case, the adversary can compromise an entire wire in certain fashion by controlling a single node on the wire. Hence,  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  having unbounded computing power can corrupt up to  $t_b, t_f$  and  $t_p$  wires in Byzantine, fail-stop and passive fashion respectively. Moreover, as a worst case assumption, we assume that the wires that are under the control of the adversary in Byzantine, fail-stop and passive fashion are mutually disjoint.

A wire which is controlled in a fail-stop fashion may fail to deliver any information, but if it delivers the information then it will be correct. Moreover, the

adversary will have no idea about the information that has passed through a wire which is controlled in fail-stop fashion. A wire which is passively controlled will always deliver correct information. However, the adversary will completely know the information that has passed through a passively controlled wire. A Byzantine corrupted wire may deliver correct information or it may deliver incorrect/changed information. However, in any case, the adversary will completely know the actual information that was sent through a Byzantine corrupted wire.

Following the approach of [14], we assume that any PSMT protocol operates as a sequence of *phases*, where a phase is a send from  $\mathbf{S}$  to  $\mathbf{R}$  or vice-versa. Moreover, since the network is synchronous, there exists a global clock and hence the transmission delay of each wire is fixed. The static mixed adversary  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  controls the **same** set of  $t_b, t_f$  and  $t_p$  wires among  $n$  wires, in Byzantine, fail-stop and passive fashion respectively, in different phases of any PSMT protocol. The static Byzantine adversary  $\mathcal{A}_{t_b}^{static}$  is a special type of  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  with  $t_f = t_p = 0$ , who controls at most  $t_b$  wires in Byzantine fashion.

Since Byzantine corrupted wires can also be eavesdropped, the maximum number of wires which can be eavesdropped by  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  in any PSMT protocol is bounded by  $t_b + t_p$ . We assume that the adversary is a *centralized* adversary and can collectively pool the data from the wires under its control and use it according to his own choice in any manner. The set of wires which  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  controls is decided before the execution of the protocol. Before the execution of the protocol, neither  $\mathbf{S}$  nor  $\mathbf{R}$  knows in advance which wires are going to be influenced by  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . However, the total number of wires that can be under the control of  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  in a certain fashion (Byzantine/failstop/passive) throughout the protocol is bounded by a threshold. Also once a wire is under the control of  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  in some fashion, it will remain corrupted in the same fashion throughout the protocol.

Throughout the paper we use  $m$  to denote the message that  $\mathbf{S}$  wants to send to  $\mathbf{R}$ , where  $m$  is a sequence of  $\ell \geq 1$  field elements, selected from uniform distribution over a finite field  $\mathbb{F}$ . Moreover, we assume that all computation and communication in our protocols are done over  $\mathbb{F}$ . The only restriction on  $\mathbb{F}$  is that  $|\mathbb{F}| \geq n$ . We use  $|m|$  to denote the number of field elements in  $m$ . We say that a wire is **corrupted**, if the information sent over the wire is changed. A wire which is not under the control of the adversary is said to be **honest** or **uncorrupted**. We now give the following definitions:

**Definition 4 (BROADCAST).** *If some information is sent over all the wires then it is said to be “broadcast”. If  $x$  is “broadcast” over at least  $2t_b + t_f + 1$  wires, then at most  $t_f$  wires may crash and fail to deliver  $x$ , where as at most  $t_b$  wires may deliver incorrect  $x$ . But at least  $t_b + 1$  wires will deliver correct  $x$ . So receiver will be able to correctly recover  $x$  by taking majority among the received values. Moreover, this is true irrespective of whether the adversary is static or mobile.*

**Definition 5 (PERFECTLY RELIABLE MESSAGE TRANSMISSION (PRMT) [14]).** *Let  $\mathcal{N} = (V, E)$  be an undirected graph representing a synchronous network. Let*

$\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  be a static mixed adversary and let  $\mathbf{S}$  and  $\mathbf{R}$  be two specific nodes in  $V$ . We define the VIEW of a node in  $V$  at any point of the execution of a protocol  $\Pi$ , to be the information the node can get from its local input (if any) to the protocol, all the messages that it had earlier sent or received, the protocol code executed by the node and its random coins. The VIEW of the adversary  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  at any point of the execution of  $\Pi$  is defined as all the information that the adversary can get from the VIEWS of all the nodes under the control of the adversary. For every message  $m$ , adversary  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  and a message transmission protocol  $\Pi$ , let  $\Gamma(\mathcal{A}_{(t_b, t_f, t_p)}^{static}, m, \Pi)$  denote the probability distribution on the VIEW of the adversary  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  at the end of the execution of  $\Pi$  when the message sent is  $m$ . The protocol  $\Pi$  is said to facilitate PRMT between  $\mathbf{S}$  and  $\mathbf{R}$ , if after interacting in phases as per the protocol, the following conditions hold:

PERFECT RELIABILITY:  $\mathbf{R}$  should correctly output  $m' = m$  without any error.

**Definition 6** (PERFECTLY SECURE MESSAGE TRANSMISSION (PSMT) [14]). A protocol  $\Pi$  is said to facilitate perfectly secure message transmission (PSMT) if it satisfies PERFECT RELIABILITY condition of PRMT. In addition to this, the protocol should also satisfy PERFECT SECRECY property which is as follows:

PERFECT SECRECY: The message is hidden from the adversary in information theoretic sense. More formally,  $\Gamma(\mathcal{A}_{(t_b, t_f, t_p)}^{static}, m, \Pi) \equiv \Gamma(\mathcal{A}_{(t_b, t_f, t_p)}^{static}, m', \Pi)$ , for all possible messages  $m'$ . That is, the above two distributions are identical irrespective of the message transmitted.

**Definition 7** (COMMUNICATION OPTIMAL PSMT PROTOCOL (OPSMT)). Let  $\mathcal{N}$  be an  $n$ - $(\mathbf{S}, \mathbf{R})$ -connected network under the influence of  $\mathcal{A}_{(t_b, t_f, t_p)}^{static} / \mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . Moreover, let  $\Omega(b)$  be the lower bound on the communication complexity of any  $r$  ( $r \geq 1$ ) phase PSMT protocol over such a network, to securely send a message  $m$  containing  $\ell$  ( $\ell \geq 1$ ) field elements against  $\mathcal{A}_{(t_b, t_f, t_p)}^{static} / \mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . If  $\Pi$  is an  $r$  phase PSMT protocol over such a network, which sends  $m$  by communicating  $\mathcal{O}(b)$  field elements against  $\mathcal{A}_{(t_b, t_f, t_p)}^{static} / \mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ , then  $\Pi$  is said to be a communication optimal PSMT (OPSMT) protocol against  $\mathcal{A}_{(t_b, t_f, t_p)}^{static} / \mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ .

### 3. Existing Literature and Our Contributions — Static Adversary

PSMT dates back to Dolev et. al. [14], who presented the first ever characterization (POSSIBILITY) for PSMT in an undirected synchronous network tolerating threshold static Byzantine adversary  $\mathcal{A}_{t_b}^{static}$ , who corrupts the same set of  $t_b$  nodes throughout the protocol in Byzantine fashion<sup>9</sup>. Dolev et. al. [14]

<sup>9</sup>Actually, Dolev et.al [14] have considered "Containment Model", where the adversary can Byzantine corrupt  $t_b$  nodes and passively corrupt  $t_p$  nodes, such that either the set of

abstracted the underlying network in terms of node disjoint paths/channels and concentrated on solving PSMT problem for a single pair of processors, the *sender*  $\mathbf{S}$  and the *receiver*  $\mathbf{R}$ , connected by  $n$  parallel and synchronous bi-directional channels  $w_1, w_2, \dots, w_n$ , also known as *wires*.<sup>10</sup> Using the wired abstraction, Dolev et.al assumed that any PSMT protocol executes in *phases*, where a phase is a send from  $\mathbf{S}$  to  $\mathbf{R}$  or vice-versa. Thus in a single phase PSMT, only  $\mathbf{S}$  communicates to  $\mathbf{R}$  and the protocol terminates. On the other hand, in a two phase PSMT, the first phase is from  $\mathbf{R}$  to  $\mathbf{S}$ , while the second phase is from  $\mathbf{S}$  to  $\mathbf{R}$ . In [14], it is shown that single phase PSMT tolerating  $\mathcal{A}_{t_b}^{static}$  is possible iff  $n \geq 3t_b + 1$ , where as two or more phase PSMT tolerating  $\mathcal{A}_{t_b}^{static}$  is possible iff  $n \geq 2t_b + 1$ .

Dolev et. al. [14] presented single phase and multi phase PSMT protocols tolerating  $\mathcal{A}_{t_b}^{static}$  with  $n = 3t_b + 1$  and  $n = 2t_b + 1$  respectively. The protocols of [14] were significantly improved in [45]. For the first time in the literature, a non-trivial lower bound on the communication complexity of PSMT protocols tolerating  $\mathcal{A}_{t_b}^{static}$  was derived in [50]. In [50], it is shown that *any two phase* PSMT over  $n \geq 2t_b + 1$  wires, must communicate  $\Omega\left(\frac{n\ell}{n-2t_b}\right)$  field elements to securely send a message containing  $\ell$  field elements against  $\mathcal{A}_{t_b}^{static}$ . The same bound was further extended in [53], where it is shown that *any three or more phase* PSMT over  $n \geq 2t_b + 1$  wires must communicate  $\Omega\left(\frac{n\ell}{n-2t_b}\right)$  field elements to securely send a message containing  $\ell$  field elements against  $\mathcal{A}_{t_b}^{static}$ . Moreover, in [53, 47], it is shown that any single phase PSMT over  $n \geq 3t_b + 1$  wires must communicate  $\Omega\left(\frac{n\ell}{n-3t_b}\right)$  field elements to securely send a message containing  $\ell$  field elements against  $\mathcal{A}_{t_b}^{static}$ . The same lower bound on single phase PSMT tolerating  $\mathcal{A}_{t_b}^{static}$  was independently derived in [16].

A single phase (multi-phase) PSMT protocol is called *communication optimal*, if the communication complexity of the protocol satisfies the communication complexity lower bound for single phase (multi-phase) PSMT i.e. a single phase (multi-phase) communication optimal protocol communicates  $\mathcal{O}\left(\frac{n\ell}{n-3t_b}\right)$  field elements ( $\mathcal{O}\left(\frac{n\ell}{n-2t_b}\right)$  field elements) for sending a message of size  $\ell$  with  $\ell \geq 1$ . The lower bound on communication complexity of single phase PSMT tolerating  $\mathcal{A}_{t_b}^{static}$  as given in [53, 16] was shown to be tight in [53, 16] by providing single phase *communication optimal* PSMT protocols over  $n \geq 3t_b + 1$  wires. In [50], the authors tried to prove the tightness of the lower bound on communication complexity of two phase PSMT by designing a *communication optimal* two phase PSMT over  $n = 2t_b + 1$  wires. However, in [1], the two

---

Byzantine corrupted nodes is a subset of the set of passively corrupted nodes or vice-versa. However, in this paper, we assume that the set of Byzantine corrupted nodes is disjoint from the set of passively corrupted nodes.

<sup>10</sup>The approach of abstracting the network as a collection of  $n$  wires is justified using Menger's theorem [29] which states that a graph is  $c - (\mathbf{S}, \mathbf{R})$ -connected iff  $\mathbf{S}$  and  $\mathbf{R}$  are connected by at least  $c$  vertex disjoint paths.

phase PSMT protocol of [50] is shown to be incorrect. Moreover, the authors in [1] have presented a two phase *communication optimal* PSMT protocol over  $n = 2t_b + 1$  wires. Though this protocol proves the tightness of the lower bound on the communication complexity of two phase PSMT tolerating  $\mathcal{A}_{t_b}^{static}$  (as given in [50]), unfortunately it has the following limitations:

1. Both  $\mathbf{S}$  and  $\mathbf{R}$  need to perform *exponential computation* (exponential in  $n$ ) and
2. The message size  $\ell$  is also exponential in  $n$ .

Subsequently, in [36], a three phase polynomial time *communication optimal* PSMT with  $n = 2t_b + 1$  is presented, where the message size  $\ell$  is polynomial in  $n$  and both  $\mathbf{S}$  and  $\mathbf{R}$  perform polynomial computation. Though this significantly improves the two phase PSMT protocol of [1] (in terms of computation and message length), designing a two phase polynomial time *communication optimal* PSMT with  $n = 2t_b + 1$  (where  $\ell$  is polynomial in  $n$ ) remained an interesting and challenging open problem. Recently, in [25], the authors have resolved this problem by designing a two phase polynomial time communication optimal PSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ .

We now summarize the existing results for PSMT in undirected synchronous networks tolerating  $\mathcal{A}_{t_b}^{static}$  in Table 1 and Table 2.

Number of Phase(s) ( $r$ )	Connectivity Requirement between $\mathbf{S}$ and $\mathbf{R}$ ( $n$ )	Lower Bound on Communication Complexity
$r = 1$	$n \geq 3t_b + 1$ [14]	$\Omega\left(\frac{n\ell}{n-3t_b}\right)$ [16, 53, 47]
$r \geq 2$	$n \geq 2t_b + 1$ [14]	$\Omega\left(\frac{n\ell}{n-2t_b}\right)$ [50, 53, 47]

Table 1: Connectivity Requirement and Lower Bound on Communication Complexity for PSMT in Undirected Synchronous Networks Tolerating  $\mathcal{A}_{t_b}^{static}$ . Here  $\ell$  denotes the message size in terms of field elements.

### 3.1. Our Contribution in PSMT over Undirected Synchronous Networks Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$

As mentioned earlier, any PSMT protocol is analyzed by (a) the connectivity requirement of the network, (b) the number of phases required by the protocol, (c) the total number of field elements communicated by  $\mathbf{S}$  and  $\mathbf{R}$  throughout the protocol and (d) the computation done by  $\mathbf{S}$  and  $\mathbf{R}$ . The *trade-offs* among these parameters are well studied in the literature in the context of PSMT over undirected synchronous network tolerating  $\mathcal{A}_{t_b}^{static}$  (see Table 1 and Table 2). In this paper, we investigate this trade-off for PSMT in the presence of threshold static *mixed* adversary  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , which according to our knowledge is the *first* attempt in the literature. So we present characterization, lower bound on communication complexity and protocols that matches the lower bound for PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . In summary, we show the following:

Number of Phases ( $r$ )	Communication Complexity in Terms of Field Elements	Remarks
$r = 1$	$\mathcal{O}\left(\frac{n\ell}{n-3t_b}\right)$ [53, 16, 47]	$\ell = \Omega(n)$
$r \geq 2$	$\mathcal{O}\left(\frac{n\ell}{n-2t_b}\right)$ [50] $\mathcal{O}\left(\frac{n\ell}{n-2t_b}\right)$ [1] $\mathcal{O}\left(\frac{n\ell}{n-2t_b}\right)$ [36] $\mathcal{O}\left(\frac{n\ell}{n-2t_b}\right)$ [25]	<ul style="list-style-type: none"> <li>• <math>r = 2</math>. Protocol shown to be incorrect in [1].</li> <li>• <math>r = 2</math>. <math>\ell</math> is exponential; Exponential computation and communication complexity [1].</li> <li>• <math>r = 3</math>. <math>\ell \geq n^2</math>; Polynomial computation and communication complexity [36].</li> <li>• <math>r = 2</math>. <math>\ell \geq n^2</math>; Polynomial computation and communication complexity [25].</li> </ul>

Table 2: PSMT Protocols with Optimum Communication Complexity Tolerating  $\mathcal{A}_{t_b}^{static}$ . Here  $\ell$  is the message size in terms of field elements and  $n$  is the corresponding connectivity requirement from Table 1.

1. Single phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  is possible iff there exists  $n \geq 3t_b + t_f + t_p + 1$  wires between  $\mathbf{S}$  and  $\mathbf{R}$ .
2. Any single phase PSMT protocol over  $n \geq 3t_b + t_f + t_p + 1$  wires tolerating  $\mathcal{A}_{(t_b, t_o, t_f, t_p)}^{static}$  must communicate  $\Omega\left(\frac{n\ell}{n-(3t_b+t_f+t_p)}\right)$  field elements to securely transmit a message containing  $\ell$  field elements. Moreover, we show that this bound is tight by designing a polynomial time single phase PSMT over  $n = 3t_b + t_f + t_p + 1$  wires, which sends a message of  $\ell$  field elements (where  $\ell$  is polynomial in  $n$ ) by communicating  $\mathcal{O}\left(\frac{n\ell}{n-(3t_b+t_f+t_p)}\right) = \mathcal{O}(n\ell)$  field elements.
3. Any two or more phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  is possible iff there exists  $n \geq 2t_b + t_f + t_p + 1$  wires between  $\mathbf{S}$  and  $\mathbf{R}$ .
4. Any two or more phase PSMT protocol over  $n \geq 2t_b + t_f + t_p + 1$  wires tolerating  $\mathcal{A}_{(t_b, t_o, t_f, t_p)}^{static}$  must communicate  $\Omega\left(\frac{n\ell}{n-(2t_b+t_f+t_p)}\right)$  field elements to securely transmit a message containing  $\ell$  field elements. Moreover, we show that this bound is tight by designing a *four phase* polynomial time PSMT protocol over  $n = 2t_b + t_f + t_p + 1$  wires, which sends a message of  $\ell$  field elements (where  $\ell$  is polynomial in  $n$ ) by communicating  $\mathcal{O}\left(\frac{n\ell}{n-(2t_b+t_f+t_p)}\right) = \mathcal{O}(n\ell)$  field elements. Finally, we present a *three phase* PSMT protocol with the same communication complexity (i.e  $\mathcal{O}(n\ell)$ ).

### 3.2. Techniques Used for Designing PSMT Protocols Against $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$

In order to design our four phase PSMT protocol against  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , we use the notion of Reed-Solomon (RS) codes and their properties from coding theory [27, 28]. Though the existing optimal PSMT protocols tolerating  $\mathcal{A}_{t_b}^{static}$  also use RS codes and their properties, they cannot be extended in a straight forward

manner for designing optimal PSMT protocols tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  (a formal discussion on this is given in Section 6.3). Furthermore, we design a *three* phase communication optimal PSMT by extending the techniques proposed in [25] for Byzantine adversary to mixed adversary.

#### 4. Coding Theory Preliminaries

In our protocols, we have used Reed-Solomon (RS) codes, which are used to reliably send message over a noisy channel. Informally, to send a message over a noisy channel, the message is encoded using RS encoding and the resultant codeword, which can be viewed as a tuple of values, is sent over the channel. During the transmission of the codeword, the values at some locations could get erased and values at few other locations could be changed arbitrarily. We can view the later type of error as Byzantine errors. At the receiving end, the shortened and possibly changed codeword is decoded to get back the original message. The redundancy which is incorporated in the codeword by the encoding function allows to correctly decode and obtain the original message.

Let  $Ch_{(t_b, t_f)}$  denote a noisy channel, where at most  $t_f$  and  $t_b$  locations of a codeword can be arbitrarily erased and changed respectively during the transmission.

**Definition 8 (REED-SOLOMON (RS) CODES [27]).** For message block  $M = (m_1, m_2, \dots, m_k)$  over  $\mathbb{F}$ , define ReedSolomon polynomial as  $P_M(x) = m_1 + m_2x + m_3x^2 + \dots + m_kx^{k-1}$ . Let  $\alpha_1, \alpha_2, \dots, \alpha_N$ , where  $N > k$ , denote a sequence of distinct and fixed elements from  $\mathbb{F}$ . Then the vector  $C = (c_1, c_2, \dots, c_N)$  where  $c_i = P_M(\alpha_i)$ ,  $1 \leq i \leq N$  is called the Reed-Solomon (RS) codeword of length/size  $N$  for the message block  $M$ . We denote the length/size of vector  $C$  by  $|C|$ .

We now define *error correction* and *error detection*.

**Definition 9 (ERROR CORRECTION AND DETECTION [43]).** Let  $\mathcal{C}$  be a subset of  $\mathbb{F}^N$  that contains all possible  $N$  length RS codeword over  $\mathbb{F}$ . Let  $C \in \mathcal{C}$  be the transmitted codeword and let  $C' \in \mathbb{F}^N$  be the received vector. By a Byzantine error, we mean the event of changing an entry in codeword  $C$ . The error locations are the indices of the entries in which  $C$  and  $C'$  differs.

The task of error correction is to find the error locations and error values in the received vector  $C'$ . On the other hand, error detection means an indication by the decoder that errors have occurred, without attempting to correct them.

Suppose a sender has generated a RS codeword  $C$  of size  $|C| = N$ , for a message block  $M$  of size  $k$  and sends the codeword  $C$  through  $Ch_{(t_b, t_f)}$ . Let  $C'$  be the received vector of size  $N' \geq N - t_f$ , which is different from  $C$  at most at  $t_b$  locations. The next theorem summarizes a known result related to Reed-Solomon codes.

**Theorem 1 (SINGLETON BOUND [27]).** The receiver can reconstruct the message  $M$  from  $C'$  iff  $N \geq 2t_b + t_f + k$ .

RS-DECODING ALGORITHM [31]: Berlekamp-Welch algorithm is one of the most simple and efficient RS decoding algorithms existing in the literature. The description of this algorithm can be found in any standard Coding theory book, such as [43, 30, 22]. However, the descriptions of the algorithm, as given in these sources, are in terms of several field and algebraic operations, which is specific to coding theory community. Since the main topic of this paper is secure message transmission, where the decoding algorithm is only used as a black-box, in order to avoid too much digression, we take the simple description of the decoding algorithm from [31].

Suppose sender has a message of size  $k$  field elements, which he wants to send reliably over  $Ch_{(t_b, t_f)}$  using RS codes. In order to do so, the sender has to encode the message into an RS codeword of size  $N = 2t_b + t_f + k$  (see Theorem 1). So the sender constructs *ReedSolomon* polynomial  $P(x)$  of degree  $k - 1$  and constructs the  $N$  length RS codeword  $C = (c_1, \dots, c_N)$ , where  $c_i = P(\alpha_i)$ , for  $i = 1, \dots, N$ . Finally, the sender sends the codeword  $C$  to the receiver over  $Ch_{(t_b, t_f)}$ .

We now assume the worst case, where exactly  $t_f$  locations get erased in the codeword  $C$  during its transmission. Although any subset of  $t_f$  locations might get erased, we make a simplifying assumption that the last  $t_f$  locations have been erased. Thus, the receiver will receive a shortened  $N'$  length vector  $C' = (c'_1, \dots, c'_{N'})$ , where  $N' = N - t_f = 2t_b + k$ . Let  $R(x)$  denote the polynomial of smallest degree passing through the points  $(\alpha_1, c'_1), \dots, (\alpha_{N'}, c'_{N'})$ . It is easy to see that  $R(x)$  will differ from  $P(x)$  for at most  $t_b$  values of  $\alpha_j$ . *Notice that the received values  $R(\alpha_j)$ 's may not lie on a  $k - 1$  degree polynomial, due to the presence of  $t_b$  corrupted values.* In order to get the original message, the receiver has to construct the polynomial  $P(x)$  from these  $N'$  values of  $R(x)$ . The question is, how the receiver can do so?

Our first observation is that if the receiver can find a polynomial  $P'(x)$  of degree  $k - 1$  that agrees with  $R(x)$  at  $k + t_b$  points, then  $P'(x) = P(x)$ . This is because out of the  $k + t_b$  points, at most  $t_b$  could be corrupted. Therefore,  $P'(x) = P(x)$  for at least  $k$  points. But a polynomial of degree  $k - 1$  is uniquely defined by its values at  $k$  points.

Now the question is: how the receiver can find such a polynomial? The receiver could try to guess where the  $t_b$  errors occurred, but this would take too much time (in fact, it would require exponential time). A very clever polynomial-time algorithm for this problem was invented by Berlekamp and Welch. The main idea is to describe the received polynomial  $R(x)$  (constituted by  $R(\alpha_j)$  values), which because of the errors may not be a  $k - 1$  degree polynomial, as a ratio of polynomials. Let  $e_1, \dots, e_{t_b}$  be the  $t_b$  positions at which errors occurred. Define the *error locator polynomial*  $E(x) = (x - e_1)(x - e_2) \dots (x - e_{t_b})$ . It is easy to see that  $E(x)$  is zero at exactly the  $t_b$  points at which errors occurred. Now observe that the following relation holds:

$$P(x)E(x) = R(x)E(x), \text{ for } x = \alpha_1, \dots, \alpha'_{N'}. \quad (1)$$

The above equation is true for all  $x$  points at which *no errors* occurred, as  $P(x) = R(x)$  at those points. On the other hand, at all  $x$  points where error

occurred,  $E(x) = 0$ . So both the sides of the above equation will be zero at those points.

Now let  $Q(x) = P(x)E(x)$ . Then  $Q(x)$  is a polynomial of degree  $t_b + k - 1$  and is therefore specified by  $t_b + k$  coefficients, which are unknown.  $E(x)$  is a polynomial of degree  $t_b$  and is described by  $t_b + 1$  coefficients. Note that the coefficient of  $x^{t_b}$  in  $E(x)$  is 1. So, there are only  $t_b$  unknown coefficients of  $E(x)$ . Thus, there are total  $(t_b + k) + (t_b) = 2t_b + k = N'$  unknowns here. Moreover, we have  $N'$  received values of  $R(x)$ . So from Equation 1, by equating  $Q(x) = R(x)E(x)$ , we can form a system of  $N'$  linear equations in  $N'$  unknowns and solve them. Once the system of equations are solved, we get  $Q(x)$  and  $E(x)$ . From the  $E(x)$  polynomial, we get the locations at which errors occurred. We can then find  $P(x)$  by computing the quotient  $Q(x)/E(x)$ .

If  $N' = k + 2t_b$  and if indeed at most  $t_b$  errors occurred, then the decoding algorithm will correctly output the message. For a complete proof of this fact, see [30, 43, 22].

We now demonstrate the working of the above algorithm with few examples. In these examples, for the ease of presentation, we make the following assumptions:

1. Instead of performing the computations over  $\mathbb{F}$ , we perform the computations over the set of whole numbers. However, the same examples will also work if we perform all the computations over a sufficiently large  $\mathbb{F}$ .
2. Instead of using  $\alpha_1, \dots, \alpha_N$  from  $\mathbb{F}$  for computing RS codeword of length  $N$ , we use  $1, \dots, N$  from the set of whole numbers.

**Remark 1.** *In all the following examples, we will specify  $k$ , the degree of the polynomial used for encoding as a function of  $t_b$ . This is because in all our PRMT and PSMT protocols,  $k$  will be indeed selected as a function of  $t_b$ . In fact, each of the following examples represents one of the possible cases, which may arise in the context of our PRMT and PSMT protocol. During the description of our PRMT and PSMT protocols, we will show how these examples are related with various such cases.*

**Example 1.** *Let  $t_b = 1, t_f = 0, k = t_b + 1 = 2$  and  $N = k + 2t_b + t_f = 4$ . Let  $m = (1, 2)$ . So the ReedSolomon polynomial is  $P(x) = 1 + 2x$ . The four length RS codeword will be  $C = (P(1), \dots, P(4)) = (3, 5, 7, 9)$ . Suppose during the transmission of the codeword, the third location gets corrupted and the receiver receives the vector  $(3, 5, 8, 9)$ . Let  $R(x)$  be the minimum degree polynomial passing through the points  $(1, 3), (2, 5), (3, 8)$  and  $(4, 9)$ . It is easy to see that  $R(x)$  is not a polynomial of degree one. The goal of the decoding algorithm will be to find a polynomial of degree  $k - 1 = 1$ , passing through  $k + t_b = 3$  of the  $R(j)$ 's. It is easy to see that there is exactly one such polynomial, namely the one passing through the points  $(1, 3), (2, 5)$  and  $(4, 9)$ . Since  $t_b = 1$ , the error locator polynomial is*

$$E(x) = (x - e_1)$$

Now  $Q(x) = P(x)E(x)$  will be of degree two. So let

$$Q(x) = Ax^2 + Bx + F$$

For  $x = 1, \dots, 4$ , it holds that

$$Q(x) = R(x)E(x)$$

The above relation implies that

$$\begin{aligned} Ax^2 + Bx + F &= R(x)(x - e_1) \\ \implies Ax^2 + Bx + F + e_1R(x) &= xR(x) \end{aligned}$$

Substituting  $x = 1, \dots, 4$  in the above relation, we get the following system of equations:

$$\begin{aligned} A + B + F + 3e_1 &= 3 \\ 4A + 2B + F + 5e_1 &= 10 \\ 9A + 3B + F + 8e_1 &= 24 \\ 16A + 4B + F + 9e_1 &= 36 \end{aligned}$$

Solving the above system of linear equations, we get  $A = 2, B = -5, F = -3$  and  $e_1 = 3$ . Thus  $Q(x) = 2x^2 - 5x - 3$  and  $E(x) = (x - 3)$ . This implies that error has occurred in the third location. Finally, the algorithm computes  $P(x) = Q(x)/E(x) = 1 + 2x$ . Thus the recovered message is  $(1, 2)$ .

In the above example, the value of  $N, t_b, t_f$  and  $k$  satisfies the inequality given by Theorem 1. However, if this is not the case, then anything can happen. We illustrate all possible cases with few examples. These examples will also illustrate the cases which will arise, when we use RS encoding and decoding in the context of secure message transmission. We will then give the formal description of the RS decoding algorithm, along with its properties.

**Example 2.** Let  $t_b = 2, k = t_b + 1 = 3, t_f = 0$  and  $N = 2t_b + 1 = 5$ . Let  $m = (1, 2, 3)$ . So  $P(x) = 1 + 2x + 3x^2$  and the RS codeword of size five is  $(P(1), \dots, P(5)) = (6, 17, 34, 57, 86)$ . Suppose during the transmission of codeword, only one error occurs, instead of  $t_b = 2$  errors. Let the error occurs at the first location and let the received vector be  $(4, 17, 34, 57, 86)$ .

Although only one error has occurred in the received vector (instead of two), the receiver has no information about this fact. The receiver will think that at most two errors are present in the received vector and would try to correct them. However, from Theorem 1, we require  $N' = 7$  in order to correct two errors. Furthermore, with  $N' = 5$  and  $k = 3$ , the decoding algorithm will correctly output the original message, only if one error would have occurred in the received vector, which is the case in this example.

If the receiver applies RS decoding algorithm, assuming the number of errors to be corrected is one, then the algorithm will proceed as follows: the decoding algorithm will try to find a polynomial of degree  $k - 1 = 2$ , passing through  $k + 1 = 4$  of the received points. It is easy to see that the only polynomial passing through four of the received points in this case is the original polynomial  $P(x)$ . Since the algorithm is assuming the number of errors to be one, the error locator polynomial will be

$$E(x) = (x - e_1)$$

Also,  $Q(x) = P(x)E(x)$  will be a polynomial of degree three. So let

$$Q(x) = Ax^3 + Bx^2 + Cx + D$$

By substituting  $x = 1, \dots, 5$  in the equation

$$Q(x) = R(x)E(x),$$

we get the following system of linear equations:

$$\begin{aligned} A + B + C + D + 4e_1 &= 4 \\ 8A + 4B + 2C + D + 17e_1 &= 34 \\ 27A + 9B + 3C + D + 34e_1 &= 102 \\ 64A + 16B + 4C + D + 57e_1 &= 228 \\ 125A + 25B + 5C + D + 86e_1 &= 430 \end{aligned}$$

By solving the above system of equations, we get  $A = 3, B = -1, C = -1, D = -1$  and  $e_1 = 1$ . Thus  $E(x) = (x - 1)$  and  $Q(x) = 3x^3 - x^2 - x - 1$ , indicating that error has occurred in the first location. Moreover,  $P(x) = Q(x)/E(x) = 1 + 2x + 3x^2$ . Thus in this case, the receiver will correctly recover the message.

In the above example, the receiver could recover the original message only because the number of *actual* errors that are present in the received vector is *same* as the number of errors that the receiver *guessed* and tried to correct. But receiver will not be sure whether the recovered message is correct. Because the decoding algorithm tried to correct only one error, where as two errors could be present in the received vector. If the receiver is sure that exactly one error could be present in the received vector, then he is certain that the output of the algorithm is correct. However, since he is unsure about the exact number of errors in the received vector, the receiver cannot take any guarantee of the output polynomial. In fact, if two errors occur in the transmitted codeword, then the decoding algorithm could end up outputting an incorrect message, as illustrated in the following example:

**Example 3.** Suppose in the previous example, exactly  $t_b = 2$  errors occur in the transmitted codeword. Namely, the errors occur in the third and fourth location and let the received vector be  $(6, 17, 28, 39, 86)$ . Notice that the errors

are introduced in the codeword in such a way that the two corrupted points, namely (3, 28) and (4, 39), along with the first two correct points, namely (1, 6) and (2, 17) lie on the polynomial  $0x^2 + 11x - 5$ . This is possible because the original polynomial  $P(x) = 3x^2 + 2x + 1$  is of degree two and two polynomials of degree two can have same value at two points.

Now if the receiver assumes that only one error is present in the received vector and tries to correct it, then the decoding algorithm will proceed as follows: the decoding algorithm will try to find a polynomial of degree  $k - 1 = 2$ , passing through  $k + 1 = 4$  of the received points. In this case, there is only one such polynomial, namely  $0x^2 + 11x - 5$ , passing through the points (1, 6), (2, 17), (3, 28) and (4, 86) and hence the decoding algorithm will output this polynomial. Out of the received five points, only three points, namely (1, 6), (2, 17) and (5, 86) lie on the original polynomial  $P(x)$ .

Since the decoding algorithm assumes the number of errors to be one, the error locator polynomial will be  $(x - e_1)$  and  $Q(x) = Ax^3 + Bx^2 + Cx + D$ . After substituting  $x = 1, \dots, 5$  in the relation  $Q(x) = R(x)(x - e_1)$ , we get the following system of equations:

$$\begin{aligned} A + B + C + D + 6e_1 &= 6 \\ 8A + 4B + 2C + D + 17e_1 &= 34 \\ 27A + 9B + 3C + D + 28e_1 &= 84 \\ 64A + 16B + 4C + D + 39e_1 &= 156 \\ 125A + 25B + 5C + D + 86e_1 &= 430 \end{aligned}$$

Solving the above system of equations, we get  $Q(x) = 11x^2 - 60x + 25$  and  $E(x) = (x - 5)$ . This will give  $P(x) = Q(x)/E(x) = 11x - 5$ . Thus the decoding algorithm outputs an incorrect polynomial. Moreover, the algorithm has output fifth location as the error location, even though the fifth location in the received vector represents a correct point on original polynomial  $P(x)$ .

In the above algorithm, the decoding algorithm outputs an incorrect message due to the following reason: the sender sent the codeword (6, 17, 34, 57, 86), corresponding to the polynomial  $3x^2 + 2x + 1$ . From Theorem 1, receiver will be able to recover the message *only if one Byzantine error occur during the transmission of the codeword*. However, during the transmission of the codeword, *two errors occur instead of one*. The received vector is (6, 17, 28, 39, 86). The errors are introduced in such a way that the received vector (6, 17, 28, 39, 86) has a *distance*<sup>11</sup> of *one* from the codeword (6, 17, 28, 39, 50), corresponding to the polynomial  $0x^2 + 11x - 5$ . Since the decoding algorithm tried to correct one error, it will work as if the transmitted codeword was (6, 17, 28, 39, 50), which is received as (6, 17, 28, 39, 86), due to the introduction of error at *fifth* location.

---

<sup>11</sup>The distance between two vectors is the number of locations at which the two vectors have different components.

So it will output fifth location as the error location, even *though it is not an error location*. Moreover, the algorithm will output an incorrect message.

The above example illustrates one of the cases, which occurs, when the actual number of errors in the transmitted codeword is more than the number of errors, which the RS decoding algorithm can correct (as given by Theorem 1). However, there may be another case. The *actual* number of errors in the transmitted codeword could be more than the number of errors, which the RS decoding algorithm can correct (as given by Theorem 1), such that the decoding algorithm fails to output any *meaningful* polynomial. If this is the case, then the decoding algorithm can simply declare that actual number of errors in the received vector is *more* than the number of errors that the decoding algorithm tried to correct. This case is illustrated by the following example:

**Example 4.** Suppose  $t_b = 2, t_f = 0, N = 2t_b + 1 = 5$  and  $k = t_b + 1 = 3$ . Let  $m = (1, 2, 0)$ . So  $P(x) = 1 + 2x$  and the transmitted codeword is  $(3, 5, 7, 9, 11)$ . Suppose two errors are arbitrarily introduced in the first two locations and let the received vector be  $(1, 2, 7, 9, 11)$ . From Theorem 1, the receiver can recover the original message from the received vector if only one error occurs in the received vector.

If the RS decoding algorithm tries to correct one error in the received vector, then the algorithm will proceed as follows: the algorithm will try to find a polynomial of degree two passing through four of the received points. However, in this case, the errors are introduced in such a way that there exist no polynomial of degree two passing through four of the received points. So the algorithm will not output any meaningful polynomial.

Since the decoding algorithm assumes the number of errors to be one, the error locator polynomial will be  $(x - e_1)$  and  $Q(x) = Ax^3 + Bx^2 + Cx + D$ . After substituting  $x = 1, \dots, 5$  in the relation  $Q(x) = R(x)(x - e_1)$ , we get the following system of equations:

$$\begin{aligned} A + B + C + D + e_1 &= 1 \\ 2A + 4B + 2C + D + 2e_1 &= 4 \\ 27A + 9B + 3C + D + 7e_1 &= 21 \\ 64A + 16B + 4C + D + 9e_1 &= 36 \\ 125A + 25B + 5C + D + 11e_1 &= 55 \end{aligned}$$

Solving the above system of equations, we get  $Q(x) = -\frac{1}{8}x^3 + \frac{7}{2}x^2 - \frac{79}{8}x + 5$  and  $E(x) = (x - \frac{5}{2})$ . Thus the decoding algorithm outputs  $Q(x)$  and  $E(x)$ , which are not meaningful. The error location as pointed out is not an integer. Moreover,  $Q(x)$  does not divide  $E(x)$ . So the algorithm can declare that more than one error are present in the received vector.

We now give the summary of the four examples, which illustrates four properties of the RS decoding algorithm, which will be further used in the context of our PSMT protocols. We will formalize these properties at the end of this section.

1. In Example 1, the receiver knows that at most  $t_b$  errors could be present in the received vector. Moreover, the value of  $k, N', t_b$  and  $t_f$  satisfies the inequality given in Theorem 1. Hence the decoding algorithm correctly outputs the message by finding the  $t_b$  errors. Moreover, the receiver is sure that the output polynomial is correct.
2. In Example 2, the receiver knows that at most  $t_b$  errors could be present in the received vector. However, only  $\frac{t_b}{2}$  errors are introduced in the received vector. By substituting  $N' = 2t_b + 1, k = t_b + 1$  and  $t_f = 0$  in the inequality of Theorem 1, we find that RS decoding algorithm can correctly output the message only if  $\frac{t_b}{2}$  errors are present in the received vector. Since only  $\frac{t_b}{2}$  errors are introduced in the received vector, the RS decoding algorithm when applied to correct  $\frac{t_b}{2}$  errors, correctly output the original message. However, *receiver has no way of knowing that the recovered message is correct as he does not know that indeed  $\frac{t_b}{2}$  errors are present in the received vector.*
3. In Example 3, more than  $\frac{t_b}{2}$  errors are introduced in the received vector. However, from Theorem 1, RS decoding algorithm can correctly output the message only if  $\frac{t_b}{2}$  errors are present in the received vector. But the actual number of errors in the received vector is more than what can be corrected. Moreover, the errors are introduced in such a way that the received vector has a distance of  $\frac{t_b}{2}$  from another valid codeword  $\hat{C}$  (different from the original codeword which was actually sent by the sender). Since the decoding algorithm is applied to correct  $\frac{t_b}{2}$  errors, the algorithm will output incorrect message, corresponding to  $\hat{C}$ . Moreover, the decoding algorithm outputs at least one correct location in the received vector as the error location. Furthermore, the receiver has no way of knowing that the recovered message is incorrect.
4. In Example 4, more than  $\frac{t_b}{2}$  errors are introduced in the received vector. From Theorem 1, RS decoding algorithm can correctly output the message only if  $\frac{t_b}{2}$  errors are present in the received vector. But the actual number of errors in the received vector is more than what can be corrected. However, the errors are introduced in such a way that the received vector has a distance of more than  $\frac{t_b}{2}$  from all possible valid codewords. Since the decoding algorithm is applied to correct  $\frac{t_b}{2}$  errors, it fails to output any *meaningful* polynomial. In this case, the receiver simply declares that more than  $\frac{t_b}{2}$  errors are present in the received vector.

In all the previous examples, we have only considered the *error correcting capability* of RS codes as given by Theorem 1. However, as mentioned at the beginning of the section, we can use RS codes to either correct errors or detect errors or simultaneously do the both. The following theorem gives the number of errors which can be corrected and detected by RS codes.

**Theorem 2 ([27, 12]).** *Let  $C$  be an  $N$  length RS codeword, corresponding to a message of size  $k$  field elements and let  $C$  be transmitted over  $Ch_{(t_b, t_f)}$ . Let  $C'$  be the received vector of size  $N'$ , where  $N' \geq (N - t_f)$ . Then RS decoding*

can correct upto  $c$  Byzantine errors in  $C'$  and simultaneously detect additional  $d$  Byzantine errors in  $C'$  iff  $N' - k \geq 2c + d$ , such that  $(c + d) \leq t_b$ .

Notice that Theorem 1 is a special case of Theorem 2 because we obtain the former by substituting  $d = 0$  and  $c = t_b$  in the later. Theorem 2 states that if we use RS decoding algorithm *only for correcting* errors (i.e.,  $d = 0$ ), then it can correct at most  $\frac{(N' - k)}{2}$ . Thus if at most  $\frac{(N' - k)}{2}$  errors are present in the received vector, then the decoding algorithm will correctly find them and recovers the original message.

On the other hand, if we use RS decoding algorithm *only for detecting* errors (i.e.,  $c = 0$ ), then it can detect at most  $(N' - k)$  errors. Thus if at most  $(N' - k)$  errors are present in the received vector, then the decoding algorithm will sense it and will output an error, indicating that at most  $(N' - k)$  errors are present in the received vector. However, unlike error correction, error detection will not output the locations at which errors are present.

If RS decoding algorithm is used with non-zero values of  $c$  and  $d$  (provided they satisfy the inequalities given in Theorem 2), then the algorithm can simultaneously correct and detect errors. In this case, the algorithm will first try to correct  $c$  errors. If the number of errors that are present in the received vector is indeed  $c$ , then the algorithm will correct all of them. Moreover, the algorithm will not detect any additional error and will correctly output the message. On the other hand, if more than  $c$  errors but at most  $(c + d)$  errors are present in the received vector, then the algorithm will detect the additional  $d$  errors (other than the  $c$  errors, which it tried to correct) and will output an error, indicating that more than  $c$  errors are present in the received vector. We illustrate the case of simultaneous correction and detection using RS decoding, with the help of following example. This example also illustrate the cases, which will arise in the context of our PSMT protocols, when we use RS decoding for simultaneous detection and correction.

**Example 5.** Let  $t_b = 2, t_f = 0, N = 2t_b + 1 = 5, k = \frac{t_b}{2} + 1 = 2, c = \frac{t_b}{2} = 1$  and  $d = \frac{t_b}{2} = 1$ . Let  $m = (1, 2)$ . So  $P(x) = 1 + 2x$  and the transmitted RS codeword is  $C = (3, 5, 7, 9, 11)$ . Since  $t_f = 0, N' = N = 5$ . Substituting the value of  $N', K, c$  and  $d$  in the inequality of Theorem 2, we find that RS decoding algorithm will be able to correct one error and detect one additional error in the received vector.

Suppose exactly one error occurs in the received vector, say in the first location. The RS decoding algorithm, when applied to correct  $c = 1$  error, will correctly identify the error. This is because the algorithm will try to find a polynomial of degree  $k - 1 = 1$ , passing through  $k + c = k + \frac{t_b}{2} = 3$  of the received points. In this case, there is only one polynomial of degree one, passing through three of the received points, namely the original polynomial  $P(x)$ . So the algorithm will correctly output the polynomial  $P(x)$ . Moreover, the receiver will be sure that the output polynomial is correct because in this case,  $(c + d) = t_b$  and the maximum number of errors that could be present in the received vector is also  $t_b$ . Since the algorithm has not detected any additional error (other than  $c$  errors), it implies that the output polynomial is indeed correct.

On the other hand, suppose that more than  $c = \frac{t_b}{2} = 1$  errors are present in the received vector; i.e., suppose two errors are present in the received vector. Moreover, the errors are introduced in the first two locations and let the received vector be  $(5, 6, 7, 9, 11)$ . Notice that here the errors are introduced in such a way that first  $t_b + (k - 1) = 3$  points in the vector, namely  $(1, 5)$ ,  $(2, 6)$  and  $(3, 7)$  lie on polynomial  $x + 4$ . On the other hand, the last  $N' - t_b = 3$  points in the vector, namely  $(3, 7)$ ,  $(4, 9)$  and  $(5, 11)$  lie on polynomial  $2x + 1$ . If we apply the RS decoding algorithm to correct  $c = 1$  error, then the decoding algorithm will try to find a polynomial of degree  $k - 1 = 1$ , passing through  $k + c = k + \frac{t_b}{2} = 3$  of the received points. In this case, there are two polynomials of degree one, passing through three of the received points. So the decoding algorithm will output an error. More specifically,  $E(x) = (x - e_1)$  and  $Q(x) = P(x)E(x) = Ax^2 + Bx + F$ . By substituting  $x = 1, \dots, 5$  in the relation  $Q(x) = R(x)E(x)$ , we get the following system of linear equations:

$$\begin{aligned} A + B + F + 5e_1 &= 5 \\ 4A + 2B + 1F + 6e_1 &= 12 \\ 9A + 3B + 1F + 7e_1 &= 21 \\ 16A + 4B + 1F + 9e_1 &= 36 \\ 25A + 5B + 1F + 11e_1 &= 55 \end{aligned}$$

However, the above system of equations does not have any solution and hence the algorithm will not output any polynomial. This will indicate to the receiver that more than  $c$  errors are present in the received vector, which are detected by the algorithm.

In the above example, we have considered the case, when  $c + d = t_b$ . If  $c + d < t_b$ , then again anything can happen. For example, the errors could be introduced in such a way that the received vector could have a distance of  $c$  from another valid codeword (other than the one sent by the sender, as in Example 3). In this case, the algorithm will output the incorrect polynomial corresponding to the other codeword. Moreover, the receiver will have no way of knowing that the output polynomial is incorrect, as  $(c + d) < t_b$ . On the other hand, the errors could be introduced in such a way that the received vector has a distance of more than  $c$  from all valid codewords (as in Example 4). In this case, the algorithm will fail to output any polynomial, indicating to the receiver that more than  $c$  errors are present in the received vector.

We now give the formal description of Berlekamp-Welch RS decoding algorithm. In the algorithm, all the computations are performed over  $\mathbb{F}$ . The algorithm takes the following inputs:

1. An  $N'$  length vector  $C'$ , received over  $Ch_{(t_b, t_f)}$ . Let  $i_1, \dots, i_{N'} \in \{1, \dots, N\}$  denote the indices of the components of the received vector. This implies that the components of the received vector at the remaining indices in the set  $\{1, \dots, N\} - \{i_1, \dots, i_{N'}\}$  are erased. Here  $N$  is the length of the

original RS codeword and  $N' \geq (N - t_f)$ . We denote the values in the received vector as  $R(\alpha_{i_1}), \dots, R(\alpha_{i_{N'}})$ .

2. Parameter  $k$ , where  $k - 1$  is the degree of the original polynomial  $P(x)$ , used for encoding the message.
3. Parameters  $c \geq 0$  and  $d \geq 0$ , subject to the condition that  $N' - k \geq 2c + d$  and  $(c + d) \leq t_b$ . Here  $c$  is the number of errors that the algorithm tries to correct and  $d$  is the number of additional errors that the algorithm tries to detect.

The algorithm is formally given in Table 3.

<b>Algorithm RS-DEC</b> ( $N', C', c, d, k$ )
<b>Goal: To Find a Polynomial of Degree <math>k - 1</math> Passing Through <math>k + c</math> Received <math>R(\alpha_j)</math>'s</b>
<ol style="list-style-type: none"> <li>1. Let the <i>error locator</i> polynomial <math>E(x) = (x - e_1) \dots (x - e_c)</math>. The coefficient of <math>x^c</math> in <math>E(x)</math> will be one.</li> <li>2. Let <math>Q(x) = P(x)E(x) = a_{c+k-1}x^{c+k-1} + a_{c+k-2}x^{c+k-2} + \dots + a_0</math> be the polynomial of degree <math>c + k - 1</math>.</li> <li>3. Form a system of <math>N'</math> equations, involving <math>2c + k \leq N'</math> unknowns <math>e_1, \dots, e_c, a_{c+k-1}, \dots, a_0</math>, by substituting <math>x = \alpha_{i_1}, \dots, \alpha_{i_{N'}}</math> in the relation <math>Q(x) = R(x)E(x)</math>.</li> <li>4. Solve the above system of equations. Now there are the following cases: <ol style="list-style-type: none"> <li>(a) If the above system of equations fails to give any solution, then output an error. In this case, the receiver concludes that more than <math>c</math> errors are present in the received vector.</li> <li>(b) If the above system of equations gives a solution, such that value of at least one of the unknowns <math>e_1, \dots, e_c, a_{c+k-1}, \dots, a_0</math> is outside the field <math>\mathbb{F}</math>, then output an error. In this case, the receiver concludes that more than <math>c</math> errors are present in the received vector.</li> <li>(c) If the above system of equations gives a solution, such that value of at least two distinct unknowns in the set <math>\{e_1, \dots, e_c\}</math> are same, then output an error. In this case, the receiver concludes that more than <math>c</math> errors are present in the received vector.</li> <li>(d) If the above system of equations gives a solution, such that value of all the <math>2c + k</math> unknowns are from the field <math>\mathbb{F}</math> and each of unknowns in <math>\{e_1, \dots, e_c\}</math> have distinct values, then do the following: <ol style="list-style-type: none"> <li>i. Compute <math>P(x) = Q(x)/E(x)</math>. Let <math>P(x) = b_0 + b_1x + \dots + b_{k-1}x^{k-1}</math>.</li> <li>ii. Output <math>(b_0, \dots, b_{k-1})</math> as the message. In addition, output an <i>error list</i>, denoted by <i>Error_List</i>. The <i>Error_List</i> indicates the values which are identified to be corrupted in <math>C'</math>. The <i>Error_List</i> will contain <math>c</math> pairs. For <math>j = 1, \dots, c</math>, the <math>j^{\text{th}}</math> entry of <i>Error_List</i> is of the form <math>(e_j, C'_{i_{e_j}})</math>, where <math>C'_{i_{e_j}}</math> denotes the <math>i_{e_j}^{\text{th}}</math> entry in <math>C'</math>.</li> </ol> </li> </ol> </li> </ol>

Table 3: Protocol for RS Decoding

**Definition 10 (GOOD/BAD ERROR LIST).** We call an error list generated by RS-DEC algorithm as “good” if each of the values in the error list, pointed as a corrupted value, is indeed corrupted. Otherwise we call the error list as “bad”. When an error list is “bad”, it points a correct value in  $C'$  as corrupted.

We now state few important properties of RS decoding, which will be used in the context of our PSMT protocols. We have already illustrated all these properties with examples and hence we will not give formal proof of these properties. For a complete formal proof, we refer [22, 43]. In all these properties, we assume that  $t'_b \leq t_b$  is the actual number of errors that are present in the received vector. The receiver has no information about  $t'_b$ , except that  $t'_b \leq t_b$ .

**Property 1.** *If  $c + d = t_b$  and  $t'_b \leq c$ , then the algorithm will correct all these errors and will detect no additional errors. So the algorithm will output  $P(x)$ , which is the original/correct  $k - 1$  degree polynomial and `Error_List`, which is a "good" error list (of cardinality at most  $c$ ). Moreover, the receiver is certain that the output polynomial  $P(x)$  is correct and the error list `Error_List` is "good". This property is illustrated in Example 5.*

**Property 2.** *If  $c + d = t_b$  and  $t'_b > c$ , then the algorithm will fail to output any message, thus indicating to the receiver that more than  $c$  errors are present in the received vector. This is because even though the actual number of errors  $t'_b$  is more than  $c$  (which is the number of errors which the algorithm tried to correct), the algorithm has the capability to detect  $(t_b - c) \geq (t'_b - c)$  additional errors. However, the algorithm can only detect the additional errors, but will not be able to correct them. So the algorithm will not output any message. In this case, the receiver concludes that more than  $c$  errors are present in  $C'$ . This property is illustrated in Example 5.*

**Property 3.** *If  $c + d < t_b$  and  $t'_b \leq c$ , then the algorithm will correct all the  $t'_b$  errors and will correctly output  $P(x)$ . Moreover, `Error_List` will be a "good" error list (of cardinality at most  $c$ ). However, receiver will not be sure/certain that the output polynomial  $P(x)$  is correct and the error list `Error_List` is "good". This is because the extra detection capability of the algorithm in this case is less than  $t_b - c$  and the value of  $t'_b$  is unknown to the receiver. This property is illustrated in Example 2.*

**Property 4.** *If  $c + d < t_b$  and  $t'_b > c$ , such that the received vector has a distance of  $c$  from another valid codeword (different from the one, which was originally sent by the sender), then the algorithm will output the incorrect  $P'(x) \neq P(x)$ , corresponding to the other codeword. Moreover, the `Error_List` will be "bad" of cardinality at most  $c$ . Furthermore, receiver will not be sure/certain that the output polynomial  $P'(x)$  is correct and the error list `Error_List` is "bad". This is because the extra detection capability of the algorithm in this case is less than  $t_b - c$  and the value of  $t'_b$  is unknown to the receiver. This property is illustrated in Example 3.*

**Property 5.** *If  $c + d < t_b$  and  $t'_b > c$ , such that the received vector has a distance of more than  $c$  from all valid codewords, then the algorithm will output error. In this case, the receiver is certain that more than  $c$  errors are present in the received vector. This property is illustrated in Example 4.*

#### 4.1. PRMT Protocol Based on RS Codes

Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $N \geq 2t_b + t_f + 1$  wires,  $w_1, \dots, w_N$ . We now design a single phase protocol called PRMT-Mixed, which allows  $\mathbf{S}$  to *reliably* send a message  $m$  containing  $\ell \geq$  field elements to  $\mathbf{R}$ , tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . The protocol is based on the properties of RS codes and will be later used in designing OPSMT protocols against  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  and  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ .

<b>Protocol PRMT-Mixed</b> ( $m, \ell, N, t_b, t_f, k$ )
<ul style="list-style-type: none"> <li>• <math>\mathbf{S}</math> breaks up <math>m</math> into blocks <math>\mathbf{B}_1, \mathbf{B}_2, \dots, \mathbf{B}_{\ell/k}</math>, each consisting of <math>k</math> field elements, where <math>k = N - 2t_b - t_f</math>. If <math>\ell</math> is not an exact multiple of <math>k</math>, a default padding is used to make <math>\ell \bmod k = 0</math>.</li> <li>• For <math>j = 1, \dots, \ell/k</math>, <math>\mathbf{S}</math> computes <math>N</math> length RS codeword <math>(c_{j1}c_{j2} \dots c_{jN})</math>, corresponding to block <math>\mathbf{B}_j</math>. For <math>i = 1, \dots, N</math>, <math>\mathbf{S}</math> sends <math>c_{ji}</math> along wire <math>w_i</math>, for <math>j = 1, \dots, \ell/k</math>.</li> <li>• Let <math>\mathbf{R}</math> receive information over wires <math>w_{i_1}, \dots, w_{i_{N'}}</math>, where <math>\{w_{i_1}, \dots, w_{i_{N'}}\} \subseteq \{w_1, \dots, w_N\}</math> and <math>N' \geq N - t_f</math>. For <math>j = 1, \dots, \ell/k</math>, let <math>\mathbf{R}</math> receive <math>c'_{ji_1}, \dots, c'_{ji_{N'}}</math> along wire <math>w_{i_1}, \dots, w_{i_{N'}}</math> respectively. Let <math>C'_j = [c'_{ji_1}, \dots, c'_{ji_{N'}}]</math>.</li> <li>• For <math>j = 1, \dots, \ell/k</math>, <math>\mathbf{R}</math> executes <math>RS - DEC(N', C'_j, t_b, 0, k)</math> and recovers <math>B_j</math>. <math>\mathbf{R}</math> then concatenates all <math>\mathbf{B}_j</math>'s to recover the message <math>m</math>.</li> </ul>

Table 4: Single Phase Reliable Message Transmission Tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$

We now prove the properties of protocol PRMT-Mixed.

**Lemma 1 (Correctness).** *Protocol PRMT-Mixed correctly delivers  $m$ .*

PROOF: In order to show that  $\mathbf{R}$  will correctly receive  $m$ , we show that  $\mathbf{R}$  will recover each  $\mathbf{B}_j$  correctly. In the protocol, each  $\mathbf{B}_j$  is of size  $k = N - 2t_b - t_f$  and is RS encoded into a codeword of length  $N \geq 2t_b + t_f + 1$ . Corresponding to each  $\mathbf{B}_j$ ,  $\mathbf{R}$  receives an  $N' \geq N - t_f$  length vector, which differs from the original codeword at most at  $t_b$  locations. So by putting  $N' \geq N - t_f, k = N - 2t_b - t_f, c = t_b$  and  $d = 0$  in the inequality of Theorem 2, we find that  $\mathbf{R}$  will be able to correct all the  $t_b$  errors in  $C'_j$  by applying  $RS - DEC$  to  $C'_j$ . Thus  $\mathbf{R}$  correctly recovers  $\mathbf{B}_j$ .  $\square$

**Lemma 2 (Communication Complexity).** *The communication complexity of protocol PRMT-Mixed is  $\mathcal{O}\left(\frac{N\ell}{N-2t_b-t_f}\right)$ .*

PROOF: Corresponding to each block of size  $k$ ,  $\mathbf{S}$  sends an RS codeword of length  $N$ . So the communication complexity of the protocol is  $\mathcal{O}\left(\frac{N\ell}{k}\right) = \mathcal{O}\left(\frac{N\ell}{N-2t_b-t_f}\right)$ .  $\square$

Protocol PRMT-Mixed has another important property. *Consider the following scenario: In protocol PRMT-Mixed,  $\mathbf{S}$  knows that  $\mathbf{R}$  has the knowledge of the exact identity of  $\alpha \leq t_b$  wires that are Byzantine corrupted.  $\mathbf{S}$  does not know the exact identity of those  $\alpha$  wires. If this is the case, then the following theorem holds:*

**Theorem 3.** *Suppose  $\mathbf{S}$  knows that  $\mathbf{R}$  has the knowledge of the exact identity of  $\alpha \leq t_b$  wires that are Byzantine corrupted. Then in protocol PRMT-Mixed,  $\mathbf{S}$  can reliably send  $m$  using block size  $k \leq (N - 2t_b - t_f) + \alpha$ . Moreover, the communication complexity of the protocol will be  $\mathcal{O}\left(\frac{N\ell}{(N-2t_b-t_f)+\alpha}\right)$*

*Proof:* Since  $\mathbf{R}$  is aware of the exact identity of  $\alpha$  Byzantine corrupted wires,  $\mathbf{R}$  can simply ignore the values received over these wires. So the length of each received vector  $C'_j$  will be  $N'$ , where  $N' \geq N - t_f - \alpha$ . Moreover  $C'_j$  will now differ from original codeword at most at  $t_b - \alpha$  locations. So by putting  $N' \geq N - t_f - \alpha$ ,  $k = (N - 2t_b - t_f) + \alpha$ ,  $c = t_b - \alpha$  and  $d = 0$  in the inequality of Theorem 2, we find that by applying  $RS - DEC$  to  $C'_j$ ,  $\mathbf{R}$  will be able to correct all the  $t_b - \alpha$  errors in  $C'_j$  and hence correctly recover  $\mathbf{B}_j$ .

Since  $k \leq (N - 2t_b - t_f) + \alpha$ , the communication complexity of the protocol will be  $\mathcal{O}\left(\frac{N\ell}{(N-2t_b-t_f)+\alpha}\right)$ .  $\square$

## 5. Single Phase PSMT Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$

In this section, we provide the necessary and sufficient condition for the existence of single phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . We then derive the lower bound on the communication complexity of single phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . Finally, we show that our lower bound is *asymptotically* tight by designing a single phase OPSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ .

### 5.1. Necessary and Sufficient Condition for the Existence of Single Phase PSMT Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$

We first recall the existing characterization of single phase PSMT tolerating  $\mathcal{A}_{t_b}^{static}$  from [14].

**Theorem 4 ([14]).** *Single phase PSMT tolerating  $\mathcal{A}_{t_b}^{static}$  is possible iff the network is  $(3t_b + 1)$ - $(\mathbf{S}, \mathbf{R})$ -connected<sup>12</sup>.*

The necessary and sufficient condition for the existence of single phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  is given by the following theorem:

**Theorem 5.** *Single phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  is possible iff there exists  $n \geq 3t_b + t_f + t_p + 1$  wires between  $\mathbf{S}$  and  $\mathbf{R}$ .*

**PROOF:** (NECESSITY): In order to prove the necessity part, we follow the strategy of [14] that is used to prove the necessity of Theorem 4. We first argue that

---

<sup>12</sup>The actual expression is  $(t_a + \max(t_a, t_e) + 1)$ - $(\mathbf{S}, \mathbf{R})$ -connected, where the adversary can corrupt up to  $t_a$  nodes actively in Byzantine fashion and  $t_e$  nodes passively in the containment model, where the set of actively corrupted nodes is a subset of the set of passively corrupted nodes or vice-versa. However, as mentioned earlier, we do not deal with such a model in this paper.

in any single phase PSMT protocol to send  $m$  against  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , the information sent over any set of  $n - (2t_b + t_f)$  wires have full information about  $m$  (see Lemma 3 presented in the sequel). Now in any single phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , the adversary can eavesdrop information over  $t_b + t_p$  wires. This implies that in any single phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ ,  $n - (2t_b + t_f) > (t_b + t_p)$  should hold. Otherwise, adversary will get information about  $m$ , thus violating the perfect secrecy property. This further implies that in any single phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ ,  $n > 3t_b + t_f + t_p$  should hold. We now proceed to prove Lemma 3.

**Lemma 3.** *Let  $\Pi$  be a single phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  that sends  $m$ , where  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n$  wires. Then the information sent over any set of  $n - (2t_b + t_f)$  wires in  $\Pi$  will have full information about  $m$ .*

PROOF: The proof is by contradiction. Suppose in  $\Pi$ , the information sent over any set of  $n - (2t_b + t_f)$  wires have no information about  $m$ . This implies that for two distinct messages  $m_1$  and  $m_2$ , with  $m_1 \neq m_2$ , the information sent over any set of  $n - (2t_b + t_f)$  wires may be same (and thus do not distinguish the messages). Let the  $n$  wires be denoted by  $w_1, \dots, w_n$ . Let  $E_1$  and  $E_2$  be two different executions of  $\Pi$  to send  $m_1$  and  $m_2$  respectively. We now define the following notations:

1.  $\alpha_1$  and  $\alpha_2$  denote the collective information sent by  $\mathbf{S}$  over wires  $w_1, \dots, w_{t_b}$  in  $E_1$  and  $E_2$  respectively.
2.  $\beta_1$  and  $\beta_2$  denote the collective information sent by  $\mathbf{S}$  over wires  $w_{t_b+1}, \dots, w_{2t_b}$  in  $E_1$  and  $E_2$  respectively.
3.  $\Gamma_1$  and  $\Gamma_2$  denote the collective information sent by  $\mathbf{S}$  over wires  $w_{2t_b+1}, \dots, w_{2t_b+t_f}$  in  $E_1$  and  $E_2$  respectively.

Since the information sent over any set of  $n - (2t_b + t_f)$  wires may be same, we can assume that the collective information sent by  $\mathbf{S}$  over wires  $w_{2t_b+t_f+1}, \dots, w_n$  in  $E_1$  and  $E_2$  are same. Let it be denoted by  $\delta$ . Thus, in  $E_1$ , the total information sent by  $\mathbf{S}$  over the  $n$  wires can be represented as  $(\alpha_1, \beta_1, \gamma_1, \delta)$ . Similarly, the total information sent by  $\mathbf{S}$  over the  $n$  wires in  $E_2$  can be represented as  $(\alpha_2, \beta_2, \gamma_2, \delta)$ . Now consider the following adversary strategies:

1.  $\mathbf{S}$  wants to send  $m_1$  and sends  $(\alpha_1, \beta_1, \gamma_1, \delta)$  to  $\mathbf{R}$  over the  $n$  wires. The adversary controls the wires  $w_{2t_b+1}, \dots, w_{2t_b+t_f}$  in fail-stop fashion and blocks them. Moreover, the adversary does no Byzantine corruption. Thus view of  $\mathbf{R}$  is  $(\alpha_1, \beta_1, \perp, \delta)$ . According to the perfect reliability property of  $\Pi$ ,  $\mathbf{R}$  will output  $m_1$ .
2.  $\mathbf{S}$  wants to send  $m_2$  and sends  $(\alpha_2, \beta_2, \gamma_2, \delta)$  to  $\mathbf{R}$  over the  $n$  wires. The adversary controls the wires  $w_{2t_b+1}, \dots, w_{2t_b+t_f}$  in fail-stop fashion and blocks them. Moreover, the adversary does no Byzantine corruption. Thus view of  $\mathbf{R}$  is  $(\alpha_2, \beta_2, \perp, \delta)$ . According to the perfect reliability property of  $\Pi$ ,  $\mathbf{R}$  will output  $m_2$ .

3. **S** wants to send  $m_1$  and sends  $(\alpha_1, \beta_1, \gamma_1, \delta)$  to **R** over the  $n$  wires. The adversary controls the wires  $w_{2t_b+1}, \dots, w_{2t_b+t_f}$  in fail-stop fashion and blocks them. Moreover, the adversary controls the wires  $w_1, \dots, w_{t_b}$  in Byzantine fashion and changes  $\alpha_1$  to  $\alpha_2$ . Thus view of **R** will be  $(\alpha_2, \beta_1, \perp, \delta)$ . Now except with probability  $\frac{1}{2}$ , **R** cannot distinguish between whether  $w_1, \dots, w_{t_b}$  is Byzantine corrupted and  $\alpha_1$  is changed to  $\alpha_2$  or  $w_{t_b+1}, \dots, w_{2t_b}$  is Byzantine corrupted and  $\beta_2$  is changed to  $\beta_1$ . Thus, on receiving  $(\alpha_2, \beta_1, \perp, \delta)$ , **R** may output  $m_1$  or  $m_2$  with probability  $\frac{1}{2}$ . But this violates the perfect reliability property of  $\Pi$ , which is a contradiction.

From the above discussion, we conclude that the information sent over any set of  $n - (2t_b + t_f)$  wires in  $\Pi$  will have full information about  $m$ .  $\square$

Now in any single phase PSMT,  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  can eavesdrop at most  $t_b + t_p$  wires. From the above lemma, the information sent over any set of  $n - (2t_b + t_f)$  wires in any single phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  will have full information about the message. From these two facts, we can conclude that in any single phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ ,  $n - (2t_b + t_f) > (t_b + t_p)$  should hold, otherwise it will violate the perfect secrecy property. Thus, in any single phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ ,  $n > (3t_b + t_f + t_p)$ . This completes the necessity proof of Theorem 5.

**Sufficiency:** In order to prove the sufficiency of the condition of Theorem 5, we design a single phase OPSMT protocol with  $n = 3t_b + t_f + t_p + 1$  tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  in Section 5.3. This completes the proof of Theorem 5.  $\square$

**Comparison 1 (Significance of Theorem 5 Over Theorem 4).** *The significance of Theorem 5 over Theorem 4 is established by the following two facts:*

1. *Theorem 5 generalizes Theorem 4 as we get the later by substituting  $t_f = t_p = 0$  in the former.*
2. *Theorem 5 shows availability of more fault tolerance in comparison to Theorem 4. For a clean interpretation of this statement, consider a network with five wires between **S** and **R**. From Theorem 4, the network can tolerate only one Byzantine corruption. However, from Theorem 5, the network can tolerate one Byzantine corruption, along with one additional fault, which can be either passive or fail-stop type. This example clearly justifies the need to study PSMT in the context of mixed adversary. Had we treated the passive or fail-stop corruption as Byzantine corruption, we would require seven wires between **S** and **R** (from Theorem 4), which is much more than what is actually required.*

## 5.2. Lower Bound on Communication Complexity of Single Phase PSMT Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$

In [53, 16], the authors have derived the lower bound on the communication complexity of any single phase PSMT tolerating  $\mathcal{A}_{t_b}^{static}$ . We now derive the lower bound on the communication complexity of any single phase PSMT

tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , by extending the arguments used in [16] for deriving the lower bound against  $\mathcal{A}_{t_b}^{static}$ .

**Theorem 6.** *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n \geq 3t_b + t_f + t_p + 1$  wires. Then any single phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  over the  $n$  wires must communicate  $\Omega\left(\frac{n\ell}{n-(3t_b+t_f)}\right)$  field elements to securely send a message  $m$  containing  $\ell \geq 1$  field elements.*

PROOF: Let  $\Pi$  be any single phase PSMT over  $n \geq 3t_b + t_f + t_p + 1$  wires, tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , which sends a message  $m$  containing  $\ell \geq 1$  field elements from  $\mathbb{F}$ . We now define the following notations:

1.  $\mathcal{M}$  denotes the message space from where  $\mathbf{S}$  selects the message to be sent. In our context,  $\mathcal{M} = \mathbb{F}^\ell$ .
2.  $\mathbf{T}_i^m$  denotes the set of all possible transmissions that can occur on wire  $w_i \in \{w_1, \dots, w_n\}$ , when  $\mathbf{S}$  transmits message  $m \in \mathcal{M}$  using protocol  $\Pi$ .
3. For  $j \geq i$ ,  $\mathbf{M}_{i,j}^m \subseteq \mathbf{T}_i^m \times \mathbf{T}_{i+1}^m \times \dots \times \mathbf{T}_j^m$  denotes the set of all possible transmissions that can occur over the wires  $\{w_i, w_{i+1}, \dots, w_j\}$ , when  $\mathbf{S}$  transmits message  $m \in \mathcal{M}$  using protocol  $\Pi$ .
4.  $\mathbf{M}_{i,j} = \bigcup_{m \in \mathcal{M}} \mathbf{M}_{i,j}^m$  and  $\mathbf{T}_i = \bigcup_{m \in \mathcal{M}} \mathbf{T}_i^m$ . We call  $\mathbf{T}_i$  as the *capacity* of wire  $w_i$  and  $\mathbf{M}_{i,j}$  as the *capacity* of the set of wires  $\{w_i, w_{i+1}, \dots, w_j\}$ .

In protocol  $\Pi$ , one element from the set  $\mathbf{T}_i$  is transmitted over each wire  $w_i$ , for  $i = 1, \dots, n$ . Moreover, each element of the set  $\mathbf{T}_i$  can be represented by  $\log |\mathbf{T}_i|$  bits. Thus, the lower bound on the communication complexity of  $\Pi$  is  $\sum_{i=1}^n \log |\mathbf{T}_i|$  bits. In the sequel, we try to estimate  $\mathbf{T}_i$ .

Since  $\Pi$  is a single phase PSMT protocol, it implies that the transmissions on any set of  $t_b + t_p$  wires is *independent* of the message. Thus, for any two messages  $m_1, m_2 \in \mathcal{M}$ , it must hold that

$$\mathbf{M}_{2t_b+t_f+1, 3t_b+t_f+t_p}^{m_1} = \mathbf{M}_{2t_b+t_f+1, 3t_b+t_f+t_p}^{m_2}.$$

Notice that the above relation must hold for any selection of  $t_b + t_p$  wires. We focussed on the set  $\{w_{2t_b+t_f+1}, \dots, w_{3t_b+t_f+t_p}\}$  just for simplicity. From Lemma 3, the transmission over any set of  $n - (2t_b + t_f)$  wires in  $\Pi$  has *full* information about  $m$  and uniquely determine  $m$ . Thus it must also hold that

$$\mathbf{M}_{2t_b+t_f+1, n}^{m_1} \cap \mathbf{M}_{2t_b+t_f+1, n}^{m_2} = \emptyset.$$

We again stress that the above relation must hold for any selection of  $n - (2t_b + t_f)$  wires. We focussed on the set  $\{w_{2t_b+t_f+1}, \dots, w_n\}$  just for simplicity. As mentioned earlier,  $\mathbf{M}_{2t_b+t_f+1, 3t_b+t_f+t_p}^m$  will be same for all messages  $m$ . Thus, in order that the above relation holds, it must hold that  $\mathbf{M}_{3t_b+t_f+t_p+1, n}^m$  is *unique* for every message  $m$ . This implies that

$$|\mathbf{M}_{3t_b+t_f+t_p+1, n}| = |\mathcal{M}|.$$

From the definition of  $\mathbf{T}_i$  and  $\mathbf{M}_{i,j}$ , we get

$$\prod_{i=3t_b+t_f+t_p+1}^n |\mathbf{T}_i| \geq |\mathbf{M}_{3t_b+t_f+t_p+1,n}| \geq |\mathcal{M}|.$$

Let  $g = n - (3t_b + t_f + t_p)$ . The above inequality holds for any selection of  $g$  wires  $\mathcal{D} \subset \{w_1, \dots, w_n\}$ , where  $|\mathcal{D}| = g$ ; i.e.,  $\prod_{w_i \in \mathcal{D}} |\mathbf{T}_i| \geq |\mathcal{M}|$ . In particular, it holds for every selection  $\mathcal{D}_k = \{w_{kg+1 \bmod n}, w_{kg+2 \bmod n}, \dots, w_{kg+g \bmod n}\}$ , with  $k \in \{0, \dots, n-1\}$ .

If we consider all the  $\mathcal{D}_k$  sets collectively, then each wire is counted exactly  $g$  times in the collection. Thus, the product of the capacities of all  $\mathcal{D}_k$  yields the capacity of the full wire set to the  $g$ -th power, and since each  $\mathcal{D}_k$  has capacity at least  $|\mathcal{M}|$ , we get

$$|\mathcal{M}|^n \leq \prod_{k=0}^{n-1} \prod_{w_j \in \mathcal{D}_k} |\mathbf{T}_j| = (\prod_{i=1}^n |\mathbf{T}_i|)^g,$$

and therefore

$$n \log(|\mathcal{M}|) \leq g \sum_{i=1}^n \log(|\mathbf{T}_i|).$$

As  $\log(|\mathcal{M}|) = \ell \log(|\mathbb{F}|)$ , from the above inequality, we get

$$\sum_{i=1}^n \log(|\mathbf{T}_i|) \geq \left( \frac{n\ell \log(|\mathbb{F}|)}{g} \right) \geq \left( \frac{n\ell \log(|\mathbb{F}|)}{n - (3t_b + t_f + t_p)} \right).$$

As mentioned earlier,  $\sum_{i=1}^n \log(|\mathbf{T}_i|)$  denotes the lower bound on the communication complexity of protocol  $\Pi$  in bits. From the above inequality, we find that the lower bound on the communication complexity of protocol  $\Pi$  is  $\left( \frac{n\ell \log(|\mathbb{F}|)}{n - (3t_b + t_f + t_p)} \right)$  bits. Now each field element from  $\mathbb{F}$  can be represented by  $\log(|\mathbb{F}|)$  bits. Thus the lower bound on the communication complexity of protocol  $\Pi$  is  $\left( \frac{n\ell}{n - (3t_b + t_f + t_p)} \right)$  field elements. This completes the derivation of lower bound on the communication complexity of single phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ .  $\square$

### 5.3. Single Phase OPSMT Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$

We now design a polynomial time single phase OPSMT called 1-OPSMT over  $n = 3t_b + t_f + t_p + 1$  wires tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . The protocol shows the sufficiency of the condition given in Theorem 5 and the tightness of the lower bound derived in Theorem 6 as well. The protocol sends a message  $m$ , which is a single field element from  $\mathbb{F}$ , by incurring a communication complexity of  $\mathcal{O}(n)$  field elements. To send a message  $m$  containing  $\ell > 1$  field elements, the protocol can be concurrently repeated for each element of  $m$ . This requires a total communication cost of  $\mathcal{O}(n\ell)$  field elements. The protocol is given in Table 5.

We now prove the properties of protocol 1-OPSMT.

**Lemma 4 (Correctness).** *In protocol 1-OPSMT,  $\mathbf{R}$  will correctly receive  $m$ .*

<b>Protocol 1-OPSMT</b> ( $m, n, t_b, t_f$ )	
<b>Phase I: S to R</b>	
<b>Computation by S :</b>	
<ol style="list-style-type: none"> <li>1. <b>S</b> selects a polynomial <math>f(x)</math> of degree <math>t_b + t_p</math>, uniformly and randomly from <math>\mathbb{F}</math>, such that <math>f(0) = m</math>.</li> <li>2. Corresponding to <math>f(x)</math>, <b>S</b> computes an <math>n</math> length RS codeword <math>C = [c_1, \dots, c_n]</math>; i.e., <math>c_i = f(\alpha_i)</math>, for <math>i = 1, \dots, n</math>.</li> </ol>	
<b>Communication by S :</b>	
<ol style="list-style-type: none"> <li>1. For <math>1 \leq i \leq n</math>, <b>S</b> sends <math>c_i</math> to <b>R</b> over wire <math>w_i</math>.</li> </ol>	
<b>Message Recovery by R :</b>	
<ol style="list-style-type: none"> <li>1. Let <b>R</b> receive information over wires <math>\{w_{i_1}, \dots, w_{i_{n'}}\} \subseteq \{w_1, \dots, w_n\}</math>, where <math>n' \geq (n - t_f)</math>.</li> <li>2. Let <b>R</b> receive <math>c'_{i_1}, \dots, c'_{i_{n'}}</math> through <math>w_{i_1}, \dots, w_{i_{n'}}</math> respectively.</li> <li>3. Let <math>C' = [c'_{i_1}, \dots, c'_{i_{n'}}]</math>. <b>R</b> executes <math>RS - DEC(n', C', t_b, 0, t_b + t_p + 1)</math> to get <math>f(x)</math>, recovers <math>m = f(0)</math> and terminates the protocol.</li> </ol>	

Table 5: Single Phase OPSMT Tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ ,  $|m| = 1, n = 3t_b + t_f + t_p + 1$

PROOF: In order to prove the lemma, we show that **R** will successfully get back  $f(x)$ . In the protocol,  $f(x)$  is RS encoded into  $n$  length RS codeword  $C = [c_1, \dots, c_n]$ . Now the  $i^{th}$  component of  $C$  is sent over wire  $w_i$ . **R** receives a shortened vector  $C' = [c'_{i_1}, \dots, c'_{i_{n'}}]$  of length  $n'$ , corresponding to  $C$ , where  $C$  and  $C'$  may differ at most at  $t_b$  locations (apart from the locations that got erased). By substituting  $c = t_b, d = 0, N' = n' \geq n - t_f$  and  $k = t_b + t_p + 1$  in the inequality of Theorem 2, we find that  $RS - DEC$  can correct all the  $t_b$  errors in  $C'$  and hence **R** can get back  $f(x)$  successfully.  $\square$

**Lemma 5 (Secrecy).** *In protocol 1-OPSMT,  $m$  will be information theoretically secure from  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ .*

PROOF: Since  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  can eavesdrop at most  $t_b + t_p$  wires,  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  will get at most  $t_b + t_p$  distinct points on  $f(x)$ . However, the degree of  $f(x)$  is  $t_b + t_p$ . Thus  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  falls short by one point to uniquely interpolate  $f(x)$ . This implies information theoretic security on  $m = f(0)$ .  $\square$

**Lemma 6 (Communication Complexity).** *The communication complexity of protocol 1-OPSMT is  $\mathcal{O}(n)$ .*

PROOF: In the protocol, **S** sends a single field element along each wire. Thus, the communication complexity of the protocol is  $\mathcal{O}(n)$ .  $\square$

**Theorem 7.** *Protocol 1-OPSMT is a single phase OPSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ .*

PROOF: From Theorem 6, any single phase PSMT over  $n = 3t_b + t_f + t_p + 1$  wires against  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , must communicate  $\Omega(n)$  field elements to securely send

a message containing  $\ell = 1$  field elements. From Lemma 6, the communication complexity of protocol 1-OPSMT is  $\mathcal{O}(n)$ . Thus, protocol 1-OPSMT is a single phase OPSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ .  $\square$

**Theorem 8.** *Suppose  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n = 3t_b + t_f + t_p + 1$  wires. Then there exists a polynomial time single phase OPSMT protocol to securely send a message  $m$  containing  $\ell \geq 1$  field elements, tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ .*

PROOF: To securely send  $m$ ,  $\mathbf{S}$  can concurrently execute  $\ell$  instances of protocol 1-OPSMT to send each element of  $m$  individually. This incurs a communication complexity of  $\mathcal{O}(n\ell)$ . From Theorem 6, any single phase PSMT over  $n = 3t_b + t_f + t_p + 1$  wires, must communicate  $\Omega(n\ell)$  field elements to securely send a message containing  $\ell$  field elements. Thus, the resultant protocol is a single phase OPSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , which sends a message containing  $\ell \geq 1$  field elements.  $\square$

## 6. Multi Phase PSMT Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$

In this section, we present the necessary and sufficient condition for the existence of any multi phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . We then prove the lower bound on the communication complexity of multi phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . Finally, we show that our lower bound is *asymptotically tight* by designing a multi phase OPSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ .

### 6.1. Necessary and Sufficient Condition for the Existence of Multi Phase PSMT Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$

We first recall the existing characterization of multi phase PSMT from [14].

**Theorem 9 ([14]).** *Multi phase PSMT between  $\mathbf{S}$  and  $\mathbf{R}$  in an undirected synchronous network  $\mathcal{N}$  tolerating  $\mathcal{A}_{t_b}^{static}$  is possible iff the network is  $(2t_b + 1)$ - $(\mathbf{S}, \mathbf{R})$ -connected.*

Now the necessary and sufficient condition for the existence of any multi phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  is given by the following theorem:

**Theorem 10.** *Any  $r \geq 2$  phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  is possible iff there exists  $n \geq 2t_b + t_f + t_p + 1$  wires between  $\mathbf{S}$  and  $\mathbf{R}$ .*

PROOF: (NECESSITY): In order to prove the necessity part, we show that in any  $r \geq 2$  phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , the information sent over any set of  $n - (t_b + t_f)$  is dependent on message  $m$  (proved in Lemma 7 given in the sequel). Now in any  $r \geq 2$  phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , the adversary can eavesdrop information over  $t_b + t_p$  wires. This implies that in any  $r \geq 2$  phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ ,  $n - (t_b + t_f) > (t_b + t_p)$  should hold. Otherwise, adversary will get information about  $m$ , which will clearly violate the perfect secrecy property of the protocol. This further implies that in any  $r \geq 2$  phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ ,  $n > 2t_b + t_f + t_p$  should hold. We now proceed to prove Lemma 7.

**Lemma 7.** *Let  $\Pi$  be any  $r \geq 2$  phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  that sends  $m$ , where  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n$  wires. Then the information sent over any set of  $n - (t_b + t_f)$  wires in  $\Pi$  is dependent on message  $m$ .*

PROOF: The proof is by contradiction. Suppose in  $\Pi$ , the information sent over any set of  $n - (t_b + t_f)$  wires is independent of  $m$ . This implies that for two distinct messages  $m_1$  and  $m_2$ , with  $m_1 \neq m_2$ , the information sent over any set of  $n - (t_b + t_f)$  wires may be same (i.e the information does not distinguish  $m_1$  from  $m_2$  and vice versa). Let  $E_1$  and  $E_2$  be two different executions of  $\Pi$  for sending  $m_1$  and  $m_2$  respectively. Moreover, without loss of generality, we assume that in  $\Pi$ ,  $\mathbf{S}$  communicates to  $\mathbf{R}$  in odd phases, while  $\mathbf{R}$  communicates to  $\mathbf{S}$  in even phases. Now suppose that in  $E_1$  and  $E_2$ , the adversary controls wires  $w_{t_b+1}, \dots, w_{t_b+t_f}$  in fail-stop fashion and does no Byzantine corruption. Thus the adversary allows no communication over  $w_{t_b+1}, \dots, w_{t_b+t_f}$  in  $E_1$  and  $E_2$ . We now define the following notations:

1.  $\alpha_1^i$  and  $\alpha_2^i$  denote the collective information sent by  $\mathbf{S}$  to  $\mathbf{R}$  over wires  $w_1, \dots, w_{t_b}$  in  $E_1$  and  $E_2$  respectively in **odd phase**  $i$ .
2.  $\alpha_1^i$  and  $\alpha_2^i$  denote the collective information sent by  $\mathbf{R}$  to  $\mathbf{S}$  over wires  $w_1, \dots, w_{t_b}$  in  $E_1$  and  $E_2$  respectively in **even phase**  $i$ .
3.  $\Gamma^i$  denotes the collective information sent by  $\mathbf{S}$  to  $\mathbf{R}$  over wires  $w_{t_b+t_f+1}, \dots, w_n$  in both  $E_1$  and  $E_2$  in **odd phase**  $i$ .  $\Gamma^i$  is assumed to be same in both  $E_1$  and  $E_2$  as the information sent over any set of  $n - (t_b + t_f)$  wires is independent of the message (according to the our assumption).
4.  $\Gamma^i$  denotes the collective information sent by  $\mathbf{R}$  to  $\mathbf{S}$  over wires  $w_{t_b+t_f+1}, \dots, w_n$  in both  $E_1$  and  $E_2$  in **even phase**  $i$ .  $\Gamma^i$  is same in both  $E_1$  and  $E_2$  due to the same reason as above.

Thus, if the adversary behaves in the above fashion, then in  $E_1$ , the view of  $\mathbf{R}$  at the end of **odd phase**  $i$  will be  $(\alpha_1^i, \perp, \Gamma^i)$  and the view of  $\mathbf{S}$  at the end of **even phase**  $i$  will be  $(\alpha_1^i, \perp, \Gamma^i)$ . Finally,  $\mathbf{R}$  will output  $m_1$ . Similarly, in  $E_2$ , the view of  $\mathbf{R}$  at the end of **odd phase**  $i$  will be  $(\alpha_2^i, \perp, \Gamma^i)$  and the view of  $\mathbf{S}$  at the end of **even phase**  $i$  will be  $(\alpha_2^i, \perp, \Gamma^i)$ . Finally,  $\mathbf{R}$  will output  $m_2$ .

Now consider another execution  $E_3$  that sends  $m_1$ . In  $E_3$ , the adversary's behavior is as follows: The adversary controls wires  $w_1, \dots, w_{t_b}$  in Byzantine fashion and wires  $w_{t_b+1}, \dots, w_{t_b+t_f}$  in fail-stop fashion respectively. In **odd phase**  $i$ , the adversary blocks the communication over wires  $w_{t_b+1}, \dots, w_{t_b+t_f}$ . Moreover, the adversary changes  $\alpha_1^i$  sent by  $\mathbf{S}$  over wires  $w_1, \dots, w_{t_b}$  into  $\alpha_2^i$ . Thus, at the end of **odd phase**  $i$ , the view of  $\mathbf{R}$  will be  $(\alpha_2^i, \perp, \Gamma^i)$  and he will think, as if he is in execution  $E_2$  and will accordingly send  $\alpha_2^i$  over wires  $w_1, \dots, w_{t_b}$  and  $\Gamma^i$  over wires  $w_{t_b+t_f+1}, \dots, w_n$  to  $\mathbf{S}$  in **even phase**  $i$ . But now, the adversary changes  $\alpha_2^i$  into  $\alpha_1^i$ . Thus at the end of **even phase**  $i$ , the view of  $\mathbf{S}$  will be  $(\alpha_1^i, \perp, \Gamma^i)$  and he will think that he is in execution  $E_1$ . Thus, in **odd phases**, the adversary controls the wires in such a way that  $\mathbf{R}$  believes that he is in  $E_2$ . On the other hand, in **even phases**, the adversary controls the wires in such a way that  $\mathbf{S}$  believes that he is in  $E_1$ . Finally, at the end of

the protocol,  $\mathbf{R}$  will output  $m_2$ , while  $\mathbf{S}$  wanted to send  $m_1$ . This is a violation of perfect reliability property of  $\Pi$ , which is a contradiction.

From the above discussion, we conclude that the information sent over any set of  $n - (t_b + t_f)$  wires in  $\Pi$  is dependent on  $m$ .  $\square$

Now in any multi phase PSMT,  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  can eavesdrop at most  $t_b + t_p$  wires. From the above lemma, the information sent over any set of  $n - (t_b + t_f)$  wires in any multi phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  is dependent on the message. From these two facts, we can conclude that in any multi phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ ,  $n - (t_b + t_f) > (t_b + t_p)$  should hold, otherwise it will violate the perfect secrecy property of PSMT protocol. Thus, in any multi phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ ,  $n > (2t_b + t_f + t_p)$  should hold. This completes the necessity proof of Theorem 10.

**Sufficiency:** In order to prove the sufficiency of the condition of Theorem 10, we design a four phase OPSMT protocol with  $n = 2t_b + t_f + t_p + 1$  tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  in Section 6.4.4. This completes the proof of Theorem 10.  $\square$

**Comparison 2 (Significance of Theorem 10 Over Theorem 9).** *The significance of Theorem 10 over Theorem 9 is established by the following two facts:*

1. *Theorem 10 generalizes Theorem 9, as we get the later by substituting  $t_f = t_p = 0$  in the former.*
2. *Theorem 10 shows the possibility of tolerating more faults in comparison to Theorem 9. For a better understanding of the above statement, consider a network with four wires between  $\mathbf{S}$  and  $\mathbf{R}$ . From Theorem 9, the network can tolerate one Byzantine corruption. However, from Theorem 10, the network can tolerate one Byzantine corruption, along with one additional fault, which can be either passive or fail-stop type. This example clearly justifies the need to study PSMT in the context of mixed adversary. Had we treated the passive or fail-stop corruption as Byzantine corruption, we would have required five wires between  $\mathbf{S}$  and  $\mathbf{R}$  (Theorem 9), which is clearly more than what is actually needed.*

## 6.2. Lower Bound on Communication Complexity of Multi Phase PSMT Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$

We now derive the lower bound on the communication complexity of Multi phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . For this, we use similar arguments used for deriving the lower bound on the communication complexity of single phase PSMT, along with few other arguments.

**Theorem 11.** *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n \geq 2t_b + t_f + t_p + 1$  wires. Then any multiple phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  over the  $n$  wires must communicate  $\Omega\left(\frac{n\ell}{n - (2t_b + t_f)}\right)$  field elements to securely send a message  $m$  containing  $\ell \geq 1$  field elements.*

To prove the theorem, we first observe the communication pattern of any multi phase PSMT protocol and show that the communication complexity of any multi phase PSMT protocol is not less than the communication complexity of a special type of single phase PSMT protocol. We then derive the lower bound on the communication complexity of this special type of single phase PSMT. More specifically, the proof of Theorem 11 follows from Lemma 8 and Lemma 9, which are proved below.

**Lemma 8.** *The communication complexity of any multi phase PSMT protocol to send a message  $m$  tolerating an adversary corrupting up to  $B$  ( $\leq t_b$ ),  $F$  ( $\leq t_f$ ) and  $P$  ( $\leq t_p$ ) wires in Byzantine, fail-stop and passive manner respectively is not less than the communication complexity of distributing  $n$  shares for the message  $m$  such that any set of  $n - (B + F)$  shares has full information about the message while any set of  $(P + B)$  shares has no information about the message.*

To prove this lemma, we begin with a definition of a weaker version of single phase PSMT called PSMT with Error Detection (PSMTED). We then prove the equivalence of the communication complexity of PSMTED protocol to send a message  $m$  and the share complexity (i.e., the length of the shares) of distributing  $n$  shares for  $m$  such that any set of  $n - (F + B)$  shares has full information about  $m$  while any set of  $(P + B)$  shares has no information about  $m$  (Claim 1). Finally, we will show that the communication complexity of any multi phase PSMT protocol is at least equal to the communication complexity of single phase protocol PSMTED (Claim 3). These two equivalences will prove the desired equivalence as stated in this lemma.

**Definition 11.** *A single phase PSMT protocol, tolerating an adversary corrupting up to  $B$  ( $\leq t_b$ ),  $F$  ( $\leq t_f$ ) and  $P$  ( $\leq t_p$ ) wires in Byzantine, fail-stop and passive fashion respectively is called  $(B, F, P)$ -PSMTED, if it satisfies the following properties:*

1. *If the adversary eavesdrops  $P + B$  wires and does no other type of corruption, then  $\mathbf{R}$  correctly and securely receives the message sent by  $\mathbf{S}$ .*
2. *If the adversary maliciously changes the information over  $B$  wires ( $B \leq t_b$ ), then  $\mathbf{R}$  detects it and aborts.*
3. *If adversary blocks at most  $F + B$  wires and does no malicious corruption, then  $\mathbf{R}$  recovers the message correctly.*
4. *The adversary obtains no information about the transmitted message in information theoretic sense.*

Observe that PSMTED is a strictly weaker version of PSMT, as a PSMT protocol not only detects errors but also corrects them and recovers the message. We next show that the properties of PSMTED protocol for sending message  $m$  is equivalent to the property of the problem of distributing  $n$  shares for  $m$  such that any set of  $n - (F + B)$  shares has full information about  $m$  while any set of  $P + B$  shares has no information about  $m$ .

**Claim 1.** *Let  $\Pi$  be a PSMTED protocol executed over  $n$  wires between  $\mathbf{S}$  and  $\mathbf{R}$ . In an execution of  $\Pi$  for sending a message  $m$ , the data  $s_i, 1 \leq i \leq n$  sent by the  $\mathbf{S}$  along the wires  $w_i, 1 \leq i \leq n$ , forms  $n$  shares for  $m$  such that any set of  $n - (F + B)$  shares has full information about  $m$  while any set of  $P + B$  shares has no information about  $m$ .*

PROOF: The fact that any set of  $P + B$  shares has no information about  $m$  follows directly from property 1 and 4 of definition of PSMTED. We now show that any set of  $n - (F + B)$  shares has full information about  $m$ . The proof is by contradiction. For a set of wires  $A$ , let  $Message(m, A)$  denote the set of messages sent along the wires in  $A$  during the execution of PSMTED to send  $m$ . Now for any set of wires  $C$  with  $|C| \geq n - (F + B)$ ,  $Message(m, C)$  should uniquely determine the message  $m$ . Suppose not, then there exists another message  $m'$  such that  $Message(m, C) = Message(m', C)$ . Now by definition, the fail-stop controlled wires as well as the Byzantine controlled wires can block all the messages sent along those wires. So assume  $\mathbf{R}$  does not receive any information over  $F + B$  wires that are not in  $C$ . Thus for two different executions of PSMTED to send two distinct message  $m$  and  $m'$ , there exists an adversary strategy such that view of  $\mathbf{R}$  at the end of two executions is exactly same. This is a contradiction to the property 3 of PSMTED protocol  $\Pi$ , which must output the correct message if at most  $F + B$  wires block the communication and no malicious corruptions take place.  $\square$

The above claim shows that the communication complexity of PSMTED protocol to send  $m$  is same as the share complexity (sum of the length of all shares) of distributing  $n$  shares for  $m$  such that any set of  $n - (F + B)$  shares has full information about  $m$  while any set of  $P$  shares has no information about the message. Now we step forward to show that the communication complexity of PSMTED protocol is the lower bound on the communication complexity of any multi phase PSMT protocol.

Before we take a closer look at the execution of any multi phase PSMT protocol, we model  $\mathbf{S}$  and  $\mathbf{R}$  as polynomial time Turing machines with access to a random tape. The number of random bits used by  $\mathbf{S}$  and  $\mathbf{R}$  are bounded by a polynomial  $q(n)$ . Let  $r_1, r_2 \in \{0, 1\}^{q(n)}$  denote the contents of the random tapes of  $\mathbf{S}$  and  $\mathbf{R}$  respectively. The message  $m$  is an element from the set  $\{0, 1\}^{p(n)}$ , where  $p(n)$  is a polynomial. A transcript for an execution of a multi phase PSMT protocol  $\Pi$  is the concatenation of all the messages sent by  $\mathbf{S}$  and  $\mathbf{R}$  along all the wires.

**Definition 12.** *A passive transcript  $\mathcal{T}(\Pi, m, r_1, r_2)$  is a transcript for the execution of the multi phase PSMT protocol  $\Pi$  with  $m$  as the message to be sent,  $r_1, r_2$  as the contents of the random tapes of sender  $\mathbf{S}$  and the receiver  $\mathbf{R}$  and the adversary remaining passive throughout the execution of  $\Pi$ , doing no other type of corruption. Let  $\mathcal{T}(\Pi, m, r_1, r_2, w_i)$  denote the passive transcript restricted to messages exchanged along the wire  $w_i$ . When  $\Pi, m, r_1, r_2$  are obvious from the context, we drop them and denote the passive transcript restricted to a wire  $w_i$  by  $\mathcal{T}_{w_i}$ . Similarly,  $\mathcal{T}_\Delta$  denote the passive transcript restricted to the set of wires in  $\Delta$ .*

Given  $(m, r_1, r_2)$  it is possible for  $\mathbf{S}$  to compute the passive transcript  $\mathcal{T}(\Pi, m, r_1, r_2)$  by simulating  $\mathbf{R}$  with random tape  $r_2$ . Similarly given  $(m, r_1, r_2)$   $\mathbf{R}$  can compute passive transcript  $\mathcal{T}(\Pi, m, r_1, r_2)$  by simulating  $\mathbf{S}$  with random tape  $r_1$ . Note that although  $\mathbf{S}$  and  $\mathbf{R}$  require both  $r_1, r_2$  to generate the passive transcript,  $\mathbf{R}$  requires only  $r_2$  in order to obtain the message  $m$  from the passive transcript  $\mathcal{T}(\Pi, m, r_1, r_2)$ . This is clear since  $\mathbf{R}$  does not have access to  $r_1$  during the execution of  $\Pi$  but still can retrieve the message  $m$  from the messages exchanged.

We next define a special type of passive transcript and prove its properties.

**Definition 13.** A passive transcript  $\mathcal{T}_\Delta$ , with  $n - F \leq |\Delta| \leq n$  is said to be a valid fault-free transcript with respect to  $\mathbf{R}$ , if there exists random string  $r_2$  and message  $m$ , such that PSMT protocol  $\Pi$  at  $\mathbf{R}$ , with  $r_2$  as the contents of the random tape and  $\mathcal{T}_\Delta$  as the messages exchanged, terminates by outputting the message  $m$ .

**Definition 14.** Two transcripts  $\mathcal{T}_\Delta$  and  $\mathcal{T}'_\Delta$ , where  $n - F \leq |\Delta| \leq n$  are said to be adversely close if the two transcripts differ only on a set of wires  $A$  such that  $|A| \leq B + (|\Delta| - (n - F))$ . Formally  $|\{w_i | \mathcal{T}_{w_i} \neq \mathcal{T}'_{w_i}\}| \leq B + (|\Delta| - (n - F))$ .

We next claim an important property of valid fault free transcripts.

**Claim 2.** No two valid fault-free transcripts  $\mathcal{T}_\Delta(\Pi, m, r_1, r_2)$  and  $\mathcal{T}_\Delta(\Pi, m', r'_1, r'_2)$  with two different message inputs  $m, m'$ , can be adversely close to each other, where  $n - F \leq \Delta \leq n$ , irrespective of the value of  $r_1, r'_1, r_2$  and  $r'_2$ .

PROOF: Suppose there exists  $r_1, r'_1, r_2, r'_2$  and two different messages  $m, m'$ , such that the valid fault-free transcripts  $\mathcal{T}_\Delta(\Pi, m, r_1, r_2)$  and  $\mathcal{T}_\Delta(\Pi, m', r'_1, r'_2)$  are adversely close. This implies that there is a set of wires  $A$ , where  $|A| \leq B + (|\Delta| - (n - F))$ , such that the two transcripts differ only on messages sent along the wires in  $A$ . Without loss of generality, assume that the last  $B + (|\Delta| - (n - F))$  wires belong to  $A$ , with  $A = X \circ Y$ , where  $|X| = B$  and  $|Y| = (|\Delta| - (n - F))$ . If such transcripts exist, then adversary can also generate  $\mathcal{T}_\Delta(\Pi, m, r_1, r_2)$  by simulating  $\mathbf{S}$  with message  $m$  and random coin  $r_1$  and simulating  $\mathbf{R}$  with random coin  $r_2$ . In a similar way, he can simulate  $\mathbf{S}$  and  $\mathbf{R}$  and generate  $\mathcal{T}_\Delta(\Pi, m', r'_1, r'_2)$ .

Now consider the following adversary behavior: in each execution of  $\Pi$ , irrespective of the random coins of  $\mathbf{S}$ ,  $\mathbf{R}$  and irrespective of the message selected by  $\mathbf{S}$ , adversary guesses that random coins of  $\mathbf{S}$  and  $\mathbf{R}$  are  $r_1$  and  $r'_2$  respectively. Moreover, adversary also guesses that  $\mathbf{S}$  wants to send  $m$ . Now irrespective of whether adversary's guess is correct or not, adversary blocks the messages over the wires in  $Y$  and tries to change the messages along wires in  $X$  such that the view of  $\mathbf{S}$  becomes  $\mathcal{T}_{\Delta-Y}(\Pi, m, r_1, r_2)$  while the view of  $\mathbf{R}$  becomes  $\mathcal{T}_{\Delta-Y}(\Pi, m', r'_1, r'_2)$ .

Notice that if either  $\mathbf{S}$  or  $\mathbf{R}$  (or both) behaves differently, as opposed to adversary's guess then adversary will not be able to generate the above views at  $\mathbf{S}$  and  $\mathbf{R}$ 's end and will be caught. But in an execution of  $\Pi$ , where  $\mathbf{S}$  indeed

wants to send  $m$  using randomness  $r_1$ , while  $\mathbf{R}$  is using randomness  $r'_2$ , adversary will be successful in causing  $\mathcal{T}_{\Delta-Y}(\Pi, m, r_1, r_2)$  and  $\mathcal{T}_{\Delta-Y}(\Pi, m', r'_1, r'_2)$  to be  $\mathbf{S}$  and  $\mathbf{R}$ 's view respectively, at the end of the protocol. In such an execution,  $\mathbf{R}$  will end up outputting  $m' \neq m$ , which violates the perfect reliability property of PSMT. This shows a contradiction.  $\square$

Till now, we have shown that a transcript over at least  $n - F$  honest wires allows  $\mathbf{R}$  to output  $m$  correctly. We now show how to reduce a multi phase PSMT protocol into a single phase PSMTED protocol.

**Protocol PSMTED**

1.  $\mathbf{S}$  computes the passive transcript  $\mathcal{T}(\Pi, m, r_1, r_2)$  for some random  $r_1$  and  $r_2$  and sends  $\mathcal{T}(\Pi, m, r_1, r_2, w_i)$  to  $\mathbf{R}$  along  $w_i$ .
2. If  $\mathbf{R}$  does not receive information through at least  $n - F$  wires then  $\mathbf{R}$  outputs ERROR and terminate.
3. Let  $\mathbf{R}$  receive information over the set of wires  $\Delta$  where  $n - F \leq |\Delta| \leq n$ .  $\mathbf{R}$  concatenates the values received along these wires to obtain a transcript  $\mathcal{T}_\Delta$  (which may be corrupted along  $t_b$  wires) and does the following:
  - For each  $m \in \{0, 1\}^{p(n)}$  and  $r_2 \in \{0, 1\}^{q(n)}$ ,  $\mathbf{R}$  checks whether  $\mathcal{T}_\Delta$  is a valid transcript with random tape contents  $r_2$  for message  $m$ . If yes, then  $\mathbf{R}$  outputs  $m$  and terminate.
4. If the above checking fails for all  $m \in \{0, 1\}^{p(n)}$  and  $r_2 \in \{0, 1\}^{q(n)}$ , then  $\mathbf{R}$  outputs ERROR and terminate.

**Claim 3.** *The Communication complexity of any multi phase PSMT protocol  $\Pi$  to send  $m$  is at least equal to the communication complexity of PSMTED protocol. Moreover protocol PSMTED satisfies the properties given in Definition 11.*

PROOF: Let  $\Pi$  be any multi phase PSMT protocol and  $\Pi^{passive}$  denotes an execution of  $\Pi$  where the adversary does only eavesdropping and does no other type of corruption during the complete execution. It is easy to see that the communication complexity of  $\Pi^{passive}$  is trivially a lower bound on the communication complexity of any multi phase PSMT protocol (where the adversary may do other types of corruptions, in addition to eavesdropping). We now show that the communication complexity of  $\Pi^{passive}$  is same as the communication complexity of PSMTED protocol. Once we do this, then the communication complexity of PSMTED protocol is a trivial lower bound on the communication complexity of any multi phase PSMT protocol.

In PSMTED,  $\mathbf{S}$  assumes its random tape contains  $r_1$  and random tape of  $\mathbf{R}$  contains  $r_2$ .  $\mathbf{S}$  also assumes that in  $\Pi$ , the adversary will only do eavesdropping and no other type of corruption and generates the passive transcript  $\mathcal{T}(\Pi, m, r_1, r_2)$ . As explained earlier,  $\mathbf{S}$  can do so by simulating  $\mathbf{R}$ , assuming the content of  $\mathbf{R}$ 's random tape to be  $r_2$ . However, note that  $\mathbf{R}$  neither knows  $m$ , nor  $r_1, r_2$ , which  $\mathbf{S}$  has used for generating  $\mathcal{T}$ .  $\mathbf{S}$  then communicates  $\mathcal{T}$  to  $\mathbf{R}$ , by sending the components of  $\mathcal{T}$  restricted to wire  $w_i$ , along  $w_i$ . It is easy

to see that the cost of communicating such a transcript by **PSMTED** is same as the communication complexity of  $\Pi^{passive}$ .

The messages sent along wire  $w_i$  in **PSMTED** protocol is the concatenation of the messages that would have been exchanged between **S** and **R** along  $w_i$  in  $\Pi^{passive}$ . Since  $\Pi^{passive}$  is a special type of execution of PSMT protocol  $\Pi$ , by the secrecy property of  $\Pi$ , the adversary cannot obtain any information about the message  $m$  by passively listening  $P + B$  wires in **PSMTED** protocol. From Claim 2, we know that valid transcripts of two different messages cannot be adversely close to each other. So irrespective of the actions of the adversary, the transcript received by **R** cannot be a valid transcript for any message other than  $m$  for any value of  $r_2$ . Hence if **R** outputs a message  $m$  then it is the same message sent by **S**. Thus protocol **PSMTED** satisfies the properties given in Definition 11.  $\square$

Claim 1, along with Claim 3 completes the proof of Lemma 8. Till now, we have shown that the communication complexity of **PSMTED** protocol is the lower bound on the communication complexity of any multi phase PSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . Moreover, the communication complexity of **PSMTED** protocol to send a message  $m$  is same as the share complexity of distributing  $n$  shares for the message  $m$ , such that any set of  $n - (F + B)$  shares has full information while any set of  $P + B$  shares has no information about the message. All these facts implies that share complexity of such a distribution gives the lower bound on the communication complexity of any multi phase PSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . We now proceed to derive the share complexity of such a distribution scheme.

**Lemma 9.** *The share-complexity (that is the sum of length of all shares) of distributing  $n$  shares for a message  $m$  containing  $\ell$  field elements from  $\mathbb{F}$ , using protocol  $(B, F, P)$ -**PSMTED**, such that any set of  $n - (F + B)$  shares has full information about the message, while any set of  $P + B$  shares has no information about the message is  $\Omega\left(\frac{n\ell}{(n-2B-F-P)}\right)$ .*

PROOF: To prove this lemma, we use similar arguments as used in Theorem 6. Let  $\Pi$  be a  $(B, F, P)$ -**PSMTED** protocol. We now define the following notations:

1.  $\mathcal{M}$  denotes the message space from where the message  $m$  is selected. In our context,  $\mathcal{M} = \mathbb{F}^\ell$ .
2. For  $i = 1, \dots, n$ ,  $\mathbf{X}_i^m$  denotes the set of all possible  $i^{th}$  share that could be generated by protocol  $\Pi$ , corresponding to message  $m \in \mathcal{M}$ .
3. For  $j \geq i$ ,  $\mathbf{M}_{i,j}^m \subseteq \mathbf{X}_i^m \times \mathbf{X}_{i+1}^m \times \dots \times \mathbf{X}_j^m$  denotes the set of all possible  $\{i^{th}, (i+1)^{th}, \dots, j^{th}\}$  shares, that could be generated by protocol  $\Pi$ , corresponding to message  $m \in \mathcal{M}$ .
4.  $\mathbf{M}_{i,j} = \bigcup_{m \in \mathcal{M}} \mathbf{M}_{i,j}^m$  and  $\mathbf{X}_i = \bigcup_{m \in \mathcal{M}} \mathbf{X}_i^m$ . We call  $\mathbf{X}_i$  as the *capacity* of  $i^{th}$  share and  $\mathbf{M}_{i,j}$  as the *capacity* of the set of  $\{i^{th}, (i+1)^{th}, \dots, j^{th}\}$  shares.

To generate  $n$  shares for message  $m$ , one element from the set  $\mathbf{X}_i$  is selected as the  $i^{\text{th}}$  share, for  $i = 1, \dots, n$ . Moreover, each element of the set  $\mathbf{X}_i$  can be represented by  $\log |\mathbf{X}_i|$  bits. Thus, the share complexity corresponding to  $m$  will be  $\sum_{i=1}^n \log |\mathbf{X}_i|$  bits. In the sequel, we try to estimate  $\mathbf{X}_i$ .

From the properties of share distribution, any set of  $B + P$  shares is *independent* of the message. Thus, for any two messages  $m_1, m_2 \in \mathcal{M}$ , it must hold that

$$\mathbf{M}_{B+F+1, 2B+F+P}^{m_1} = \mathbf{M}_{B+F+1, 2B+F+P}^{m_2}.$$

Notice that the relation above must hold for any selection of  $B + P$  shares. We focussed on the set of  $\{(B + F + 1)^{\text{th}}, \dots, (2B + F + P)^{\text{th}}\}$  shares just for simplicity. Also, from the properties of share distribution, any set of  $n - (F + B)$  shares has *full* information about the message  $m$  and uniquely determine  $m$ . Thus it must also hold that

$$\mathbf{M}_{B+F+1, n}^{m_1} \cap \mathbf{M}_{B+F+1, n}^{m_2} = \emptyset.$$

We again stress that the above relation must hold for any selection of  $n - (B + F)$  shares. We focussed on the set of  $\{(B + F + 1)^{\text{th}}, \dots, n^{\text{th}}\}$  shares just for simplicity. As mentioned earlier,  $\mathbf{M}_{B+F+1, 2B+F+P}^m$  will be same for all messages  $m$ . Thus, in order that the above relation holds, it must hold that  $\mathbf{M}_{2B+F+P+1, n}^m$  is *unique* for every message  $m$ . This implies that

$$|\mathbf{M}_{2B+F+P+1, n}| = |\mathcal{M}|.$$

From the definition of  $\mathbf{X}_i$  and  $\mathbf{M}_{i, j}$ , we get

$$\prod_{i=2B+F+P+1}^n |\mathbf{X}_i| \geq |\mathbf{M}_{2B+F+P+1, n}| \geq |\mathcal{M}|.$$

Let  $g = n - (2B + F + P)$ . The above inequality holds for any set of  $g$  shares  $\mathcal{D}$ , where  $|\mathcal{D}| = g$ ; i.e.,  $\prod_{i \in \mathcal{D}} |\mathbf{X}_i| \geq |\mathcal{M}|$ . In particular, it holds for every selection  $\mathcal{D}_k$  of  $\{(kg + 1)^{\text{th}} \bmod n, (kg + 2)^{\text{th}} \bmod n, \dots, (kg + g)^{\text{th}} \bmod n\}$  shares, with  $k \in \{0, \dots, n - 1\}$ .

If we consider all the  $\mathcal{D}_k$  sets collectively, each share is counted exactly  $g$  times in the collection. Thus, the product of the capacities of all  $\mathcal{D}_k$  yields the capacity of the full share set to the  $g$ -th power, and since each  $\mathcal{D}_k$  has capacity at least  $|\mathcal{M}|$ , we get

$$|\mathcal{M}|^n \leq \prod_{k=0}^{n-1} \prod_{j \in \mathcal{D}_k} |\mathbf{X}_j| = (\prod_{i=1}^n |\mathbf{X}_i|)^g,$$

and therefore

$$n \log(|\mathcal{M}|) \leq g \sum_{i=1}^n \log(|\mathbf{X}_i|).$$

As  $\log(|\mathcal{M}|) = \ell \log(|\mathbb{F}|)$ , from the above inequality, we get

$$\sum_{i=1}^n \log(|\mathbf{X}_i|) \geq \left( \frac{n \ell \log(|\mathbb{F}|)}{g} \right) \geq \left( \frac{n \ell \log(|\mathbb{F}|)}{n - (2B + F + P)} \right).$$

As mentioned earlier,  $\sum_{i=1}^n \log(|\mathbf{X}_i|)$  denotes the share complexity in bits of distributing  $n$  shares of a message  $m$  using protocol II. From the above inequality,

we find that the share complexity of protocol  $\Pi$  is  $\Omega\left(\frac{n\ell \log(|\mathbb{F}|)}{n-(2B+F+P)}\right)$  bits. Now each field element from  $\mathbb{F}$  can be represented by  $\log(|\mathbb{F}|)$  bits. Thus the share complexity of protocol  $\Pi$  is  $\Omega\left(\frac{n\ell}{n-(2B+F+P)}\right)$  field elements. This completes the proof of Lemma 9.  $\square$

Since  $P \leq t_p$ ,  $F \leq t_f$  and  $B \leq t_b$ ,  $\Omega\left(\frac{n\ell}{n-2B-F-P}\right) = \Omega\left(\frac{n\ell}{n-(2t_b+t_f+t_p)}\right)$ . Theorem 11 now follows from Lemma 8 and Lemma 9.  $\square$

In the sequel, we design a four phase OPSMT protocol, whose total communication complexity matches the bound proved in Theorem 11. This will show that the bound is tight. Before that, we formally discuss why the techniques used in existing OPSMT tolerating  $\mathcal{A}_{t_b}^{static}$  cannot be extended in a straightforward manner for designing OPSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ .

### 6.3. Existing OPSMT Protocols Tolerating $\mathcal{A}_{t_b}^{static}$ and Their Limitations

We now describe a two phase sub-protocol, called  $\Pi_{t_b}^i$ , which was proposed in [45] and has been used in the *three* phase OPSMT protocol of [36], tolerating  $\mathcal{A}_{t_b}^{static}$ . A slight variation of  $\Pi_{t_b}^i$  is also used in the *exponential time two phase* OPSMT protocol of [1] and *polynomial time two phase* OPSMT protocol of [25], tolerating  $\mathcal{A}_{t_b}^{static}$ . We then point out the reasons for the sub-protocol for not scaling up into the design of OPSMT protocols tolerating mixed adversary. In the OPSMT protocols tolerating  $\mathcal{A}_{t_b}^{static}$ ,  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n = 2t_b + 1$  wires. The OPSMT protocols concurrently execute the sub-protocol  $\Pi_{t_b}^i$ , corresponding to each wire  $w_i$ , for  $1 \leq i \leq n$ . The current description of  $\Pi_{t_b}^i$  (given in Table 6) has been taken from [1]. In  $\Pi_{t_b}^i$ ,  $\mathbf{A}$  and  $\mathbf{B}$  are two nodes (any of them can be  $\mathbf{S}$  or  $\mathbf{R}$ ), who are connected by  $n = 2t_b + 1$  wires,  $w_1, w_2, \dots, w_n$ , of which any  $t_b$  can be under the control of  $\mathcal{A}_{t_b}^{static}$ .  $\mathbf{A}$  attempts to privately transmit a random value  $s \in \mathbb{F}$  to  $\mathbf{B}$  over wire  $w_i$  and obtains the feedback afterward. The protocol has the following properties:

- (a) If  $w_i$  is corrupted, then it is disqualified by  $\mathbf{A}$  at the end of the second phase.
- (b) If  $w_i$  is uncorrupted, then  $\mathbf{A}$  is certain at the end of second phase that  $\mathbf{B}$  has correctly received the value  $s$ .
- (c) If  $w_i$  is not under the control of  $\mathcal{A}_{t_b}^{static}$ , then adversary obtains *no* information about  $s$ .

We now prove the properties of  $\Pi_{t_b}^i$ .

**Claim 4 (Property (a)).** *If  $w_i$  is corrupted and has communicated  $p'_i(x) \neq p_i(x)$  to  $\mathbf{B}$ , then it will be disqualified by  $\mathbf{A}$  at the end of the second phase.*

PROOF: To prove the claim, we show that there will be at least one pair  $(p'_i(\alpha_j), r'_{ij})$ , for some  $j$ , such that  $p'_i(\alpha_j) \neq p_i(\alpha_j)$ , which will be detected by  $\mathbf{B}$  and sent over to  $\mathbf{A}$  reliably. From this pair,  $\mathbf{A}$  can always detect that  $w_i$  is corrupted and has transmitted a wrong polynomial. So being a corrupted wire, let

<b>Protocol <math>\Pi_{t_b}^i</math></b>	
<b>Phase I: A to B</b>	
<ul style="list-style-type: none"> <li>• <b>A</b> selects a random polynomial <math>p_i(x)</math> over <math>\mathbb{F}</math> of degree <math>t_b</math>, such that <math>p_i(0) = s</math>. <b>A</b> sends over <math>w_i</math>, the polynomial <math>p_i(x)</math> (We assume that a polynomial is sent by communicating its coefficients) and over each wire <math>w_j, 1 \leq j \leq n</math>, <b>A</b> sends <math>r_{ij} = p_i(\alpha_j)</math>, where <math>\alpha_j, 1 \leq j \leq n</math> are publicly known values from <math>\mathbb{F}</math>.</li> </ul>	
<b>Phase II: B to A</b>	
<ul style="list-style-type: none"> <li>• Let <b>B</b> receive the polynomial <math>p'_i(x)</math> over <math>w_i</math> and the value <math>r'_{ij}, 1 \leq j \leq n</math> over wire <math>w_j, 1 \leq j \leq n</math>.</li> <li>• For every pair of values such that <math>p'_i(\alpha_j) \neq r'_{ij}</math>, <b>B</b> tries to reliably send these values to <b>A</b>. For this, <b>B</b> broadcasts (In [50, 36], these values are transmitted using efficient reliable protocol, using a technique called <i>Matching Technique</i>, which ensures that <b>A</b> correctly receives them) the pair <math>(p'_i(\alpha_j), r'_{ij})</math>.</li> </ul>	
<b>Local Computation by A</b>	
<ul style="list-style-type: none"> <li>• For every broadcasted pair <math>(p'_i(\alpha_j), r'_{ij})</math>, <b>A</b> verifies <math>p'_i(\alpha_j) \stackrel{?}{=} p_i(\alpha_j)</math> and <math>r'_{ij} \stackrel{?}{=} r_{ij}</math>.</li> <li>• If the first test fails, then it implies that <b>B</b> has received incorrect <math>p_i(x)</math> over <math>w_i</math> and hence <b>A</b> discards wire <math>w_i</math>. If the first test succeeds but the second test fails, then <b>A</b> concludes that <b>B</b> has received incorrect <math>r_{ij} = p_i(\alpha_j)</math> over <math>w_j</math> and hence discards wire <math>w_j</math>.</li> <li>• After completing the above test for each broadcasted pair, if <math>w_i</math> is not discarded, then <b>A</b> concludes that <b>B</b> has received <math>s</math> correctly.</li> </ul>	

Table 6: Sub-protocol used in existing PSMT/OPSMT protocols against  $\mathcal{A}_{t_b}^{static}, n = 2t_b + 1$

$w_i$  has transmitted  $p'_i(x) \neq p_i(x)$ . Out of the  $n$  wires between **A** and **B**, at most  $t_b$  are under the control of  $\mathcal{A}_{t_b}^{static}$ . Hence those corrupted  $t_b$  wires may transmit values corresponding to  $p'_i(x)$ . Since both  $p_i(x)$  and  $p'_i(x)$  are polynomials of degree  $t_b$ , they may have at most  $t_b$  common points (lying on both of them). In the worst case, polynomial  $p'_i(x)$  may match with the values received over  $2t_b$  wires ( $t_b$  corrupted wires +  $t_b$  honest wires). But since  $n = 2t_b + 1$ , there is  $n - 2t_b = 1$  honest wire, say  $w_j$ , who will correctly deliver  $r'_{ij} = r_{ij} = p_i(\alpha_j)$  to **B** and  $p'_i(\alpha_j) \neq p_i(\alpha_j)$ . In this case,  $w_i$  and  $w_j$  will conflict each other and hence **B** will broadcast  $(p'_i(\alpha_j), r'_{ij})$ .

At the end of second phase, **A** will correctly receive  $(p'_i(\alpha_j), r'_{ij})$  and will identify that  $p'_i(\alpha_j) \neq p_i(\alpha_j)$  and hence will conclude that  $w_i$  is corrupted.  $\square$

**Claim 5 (Property (b)).** *If  $w_i$  is uncorrupted, then **A** is certain at the end of second phase that **B** has correctly received the value  $s$ .*

PROOF: If  $w_i$  is uncorrupted then  $p'_i(x) = p_i(x)$ . Moreover, the honest wires will correctly deliver  $r'_{ij} = r_{ij}$ . So for every honest  $w_j$ ,  $p'_i(\alpha_j) = r'_{ij}$  will hold. However, a corrupted  $w_j$  may deliver  $r'_{ij} \neq r_{ij}$  and so  $p'_i(\alpha_j) \neq r'_{ij}$ . In this case, **B** will broadcast  $(p'_i(\alpha_j), r'_{ij})$ . On receiving such pairs, **A** will identify that  $p'_i(\alpha_j) = p_i(\alpha_j)$  but  $r'_{ij} \neq r_{ij}$  and hence will discard  $w_j$ . Since  $w_i$  has delivered  $p_i(x)$  correctly, **A** will be certain that **B** has correctly received the value  $s$ .  $\square$

**Claim 6 (Property (c)).** *If  $w_i$  is not under the control of  $\mathcal{A}_{t_b}^{static}$ , then adversary obtains no information about  $s$ .*

PROOF: If  $w_i$  is not under the control of  $\mathcal{A}_{t_b}^{static}$ , then adversary will not know  $p_i(x)$ . However,  $\mathcal{A}_{t_b}^{static}$  will know  $t_b$  distinct points on  $p_i(x)$  by eavesdropping at most  $t_b$  wires. However, degree of  $p_i(x)$  is  $t_b$  and hence  $\mathcal{A}_{t_b}^{static}$  will be short by one point to uniquely interpolate  $p_i(x)$ . Thus,  $\mathcal{A}_{t_b}^{static}$  will not know any information about  $s = p_i(0)$ .  $\square$

The three phase OPSMT protocol of [36] performs  $n = 2t_b + 1$  parallel executions of sub-protocol  $\Pi_{t_b}^i$ , one for each wire  $w_i$ . By replacing the broadcast in second step of **Phase II** of  $\Pi_{t_b}^i$ , with the technique called *matching technique* [50], of reliably sending the conflicting pair of values  $(p'_i(\alpha_j), r'_{ij})$ , the OPSMT protocol of [36] incurs a total communication cost of  $\mathcal{O}(n^2)$ . The OPSMT protocol of [36] is communication optimal due to the following high-level idea: Since there are  $n - t_b = t_b + 1$  honest wires free from  $\mathcal{A}_{t_b}^{static}$ , the execution of the sub-protocol  $\Pi_{t_b}^i$  corresponding to these honest wires will be successful. Thus the  $s$  values which are transmitted using these executions will be correctly and secretly established between **A** and **B**. These  $s$  values can be treated as a random one time pad of length  $n - t_b = t_b + 1$ , established between **S** and **R**, about which  $\mathcal{A}_{t_b}^{static}$  has *no* information. Using this pad, **S** can mask a message of size  $t_b + 1 = \Theta(n)$  by X-ORing it with the pad. **S** then reliably sends the blinded message by broadcasting it, incurring a communication cost of  $\mathcal{O}(n^2)$ . This results in the three phase OPSMT protocol of [36], which sends a message of size  $\Theta(n)$  by communicating  $\mathcal{O}(n^2)$  field elements over  $n = 2t_b + 1$  wires.

Similarly, the two phase OPSMT protocol of [1] and [25] parallelly executes a slightly modified version of sub-protocol  $\Pi_{t_b}^i$ , one for each wire  $w_i$  and tries to establish an information theoretic secure one time pad of appropriate length between **S** and **R**.

Now for designing OPSMT protocol against  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , we require  $n = 2t_b + t_p + t_f + 1$  wires between **S** and **R** (see Theorem 10). Let  $\Pi_{(t_b, t_f, t_p)}^i$  be the extended version of sub-protocol  $\Pi_{t_b}^i$ , tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . The only change in  $\Pi_{(t_b, t_f, t_p)}^i$  is that the degree of the polynomial  $p_i(x)$  is now  $t_b + t_p$  instead of  $t_b$ . This is because  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  can passively listen the contents of at most  $t_b + t_p$  wires out of  $2t_b + t_p + t_f + 1$  wires. Out of the  $n$  wires, there are only  $n - (t_b + t_f + t_p) = t_b + 1$  wires which are completely free from  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . Hence the execution of sub-protocol  $\Pi_{(t_b, t_f, t_p)}^i$  corresponding to these honest wires will contribute to the establishment of common secret information between **S** and **R**, about which  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  will have *no* information. Note that the remaining  $t_b + t_f + t_p$  executions of sub-protocol  $\Pi_{(t_b, t_f, t_p)}^i$  corresponding the wires under the control of  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , may not lead to the establishment of any *common secret* information between **S** and **R**. This is because along  $t_f$  wires the sub-protocol may fail due to fail-stop corruption. Furthermore, the adversary may know the secret corresponding to the execution of remaining  $t_b + t_p$  executions of  $\Pi_{(t_b, t_f, t_p)}^i$  by passively listening to the polynomials of these wires. Thus, in

the worst case, the resultant protocol establishes a random one time pad of length  $t_b + 1$  between  $\mathbf{S}$  and  $\mathbf{R}$  by communicating  $\mathcal{O}(n^2)$  field elements, where  $n = 2t_b + t_f + t_p + 1$ . Now using this pad,  $\mathbf{S}$  can securely send a message containing  $t_b + 1$  field elements by communicating  $\mathcal{O}(n^2)$  field elements. Now notice that unlike in the case of only Byzantine adversary (where  $n = 2t_b + 1$  and  $t_b = \Theta(n)$ ),  $t_b$  may not be  $\Theta(n)$  in the presence of mixed adversary, where  $n = 2t_b + t_f + t_p + 1$ . In the worst case,  $t_b$  may be constant. This does not lead to an OPSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  (see Theorem 11). This is because if  $n = 2t_b + t_f + t_p + 1$  and the total communication complexity of the protocol is  $\mathcal{O}(n^2)$ , then the protocol is OPSMT in the presence of  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  only if it sends a message of size  $t_b + t_f + t_p = \Theta(n)$ .

Hence the techniques of [36] and [1, 25] for designing OPSMT protocol against  $\mathcal{A}_{t_b}^{static}$  cannot be extended in a straightforward manner for the design of OPSMT protocol against mixed adversary. To achieve optimality,  $\mathbf{S}$  should be able to securely establish a one time pad of size  $t_b + t_f + t_p$ , instead of  $t_b + 1$  by communicating  $\mathcal{O}(n^2)$  field elements. Thus we require new techniques for designing OPSMT protocol against mixed adversary, which we explore in the sequel.

#### 6.4. Four Phase OPSMT Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$

Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n = 2t_b + t_f + t_p + 1$  wires  $w_i, 1 \leq i \leq n$ . We design a four phase OPSMT protocol 4-OPSMT-Static, which securely sends a message containing  $n$  field elements by communicating  $\mathcal{O}(n^2)$  field elements, tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . We first design few sub-protocols and finally combine them to obtain 4-OPSMT-Static.

##### 6.4.1. Protocol Pad-Establishment-Static - A Conditional Single Phase Protocol to Establish a One Time Pad Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$

Suppose  $\mathbf{A}$  and  $\mathbf{B}$  are connected by  $n = 2t_b + t_f + t_p + 1$  wires that are under the influence of  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . Also assume that  $\mathbf{A}$  in advance knows the identity of at least  $\frac{t_b}{2}$  wires which are under the control of  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  in Byzantine fashion. Under this assumption, we design a single phase protocol called Pad-Establishment-Static, which securely establishes a random one time pad of length  $n$  between  $\mathbf{A}$  and  $\mathbf{B}$ , which is information theoretically secure from  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . The protocol is given in Table. 7.

We now prove the properties of protocol Pad-Establishment-Static.

**Lemma 10 (Correctness).** *Suppose  $\mathbf{A}$  in advance knows the identity of at least  $\frac{t_b}{2}$  wires which are under the control of  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  in Byzantine fashion. Then protocol Pad-Establishment-Static correctly establishes the  $n$  tuple  $q = [q_1(0) \dots q_n(0)]$  between  $\mathbf{A}$  and  $\mathbf{B}$  in a single phase, tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ .*

PROOF: From  $L_{fault}$ ,  $\mathbf{B}$  identifies  $|L_{fault}| \geq \frac{t_b}{2}$  Byzantine corrupted wires and neglects them. Among the remaining wires, at most  $t_f$  may fail to deliver any information due to fail-stop corruption. So in the worst case,  $N' = n -$

**Protocol Pad-Establishment-Static:**  $n = 2t_b + t_f + t_p + 1$

**Assumption:** **A** knows the identity of at least  $\frac{t_b}{2}$  wires, which are Byzantine corrupted.

**Computation by A :**

1. **A** saves the identity of the wires which are known to be Byzantine corrupted, in a list  $L_{fault}$ . According to the assumption,  $\frac{t_b}{2} \leq |L_{fault}| \leq t_b$ .
2. **A** selects  $n$  random polynomials  $q_j(x), 1 \leq j \leq n$ , over  $\mathbb{F}$ , each of degree  $t_b - |L_{fault}| + t_p$ .
3. For  $j = 1, \dots, n$ , using  $q_j(x)$ , **A** computes RS codeword  $Q_j = [q_{j1} \ q_{j2} \ \dots \ q_{jn}]$  of size  $n$ , where  $q_{ji} = q_j(\alpha_i)$ , for  $i = 1, \dots, n$ .

**Communication by A :**

1. For  $i = 1, \dots, n$ , if  $w_i \notin L_{fault}$ , then **A** sends to **B** the  $i^{th}$  component of the  $n$  codewords, namely  $q_{1i}, q_{2i}, \dots, q_{ni}$  over  $w_i$ .
2. **A** broadcasts  $L_{fault}$  to **B**.

**Computation by B :**

1. **B** correctly receives  $L_{fault}$  and neglects any information received over  $w_i \in L_{fault}$ .
2. Among the wires in  $\{w_1, \dots, w_n\} - L_{fault}$ , let  $w_{i_1}, \dots, w_{i_{N'}}$  be the wires that delivered some information to **B**. Note that  $n - t_f - |L_{fault}| \leq N' \leq n - |L_{fault}|$ . Moreover,  $N'$  is at least  $n - t_f - |L_{fault}|$ , as among the  $n - |L_{fault}|$  wires, at most  $t_f$  wires may fail to deliver any information due to fail-stop corruption.
3. For  $j = 1, \dots, n$ , let **B** receive  $q'_{ji_1}, \dots, q'_{ji_{N'}}$  over wires  $w_{i_1}, \dots, w_{i_{N'}}$ , respectively. Let  $Q'_j = [q'_{ji_1} \ \dots \ q'_{ji_{N'}}]$ .
4. For  $j = 1, \dots, n$ , **B** recovers  $q_j(x)$  (and hence  $q_j(0)$ ) by executing  $RS - DEC(N', Q'_j, t_b - |L_{fault}|, 0, t_b - |L_{fault}| + t_p + 1)$ .

The  $n$  tuple  $q = [q_1(0) \ \dots \ q_n(0)]$  is established correctly and securely between **A** and **B**.

Table 7: Single Phase Protocol to Establish a One Time Pad of length  $n = 2t_b + t_f + t_p + 1$

$|L_{fault}| - t_f$ . The received vector  $Q'_j$  corresponds to the RS codeword  $Q_j$  that is encoded using polynomial  $q_j(x)$  of degree  $t_b - |L_{fault}| + t_p$ . Since **B** has neglected information corresponding to  $|L_{fault}|$  Byzantine corrupted wires, there are at most  $t_b - |L_{fault}|$  corrupted values in each  $Q'_j$ . Now **B** can recover the original polynomial  $q_j(x)$  corresponding to each  $Q'_j$ . In fact, substituting  $N' = n - |L_{fault}| - t_f, k = t_b - |L_{fault}| + t_p + 1, c = t_b - |L_{fault}|$  and  $d = 0$  in the inequality of Theorem 2, we find that  $RS - DEC(N', Q'_j, t_b - |L_{fault}|, 0, t_b - |L_{fault}| + t_p + 1)$  will be able to correct all the  $t_b - |L_{fault}| \leq \frac{t_b}{2}$  Byzantine errors in  $Q'_j$ .  $\square$

**Lemma 11 (Security).** *In protocol Pad-Establishment-Static,  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  will get no information about the pad  $q = [q_1(0) \ \dots \ q_n(0)]$ .*

PROOF: In the protocol, the adversary gets at most  $t_b - |L_{fault}| + t_p$  distinct points on each  $t_b - |L_{fault}| + t_p$  degree polynomial  $q_j(x)$ . This implies information

theoretic security of each  $q_j(0)$ .  $\square$

**Lemma 12 (Communication Complexity).** *Protocol Pad-Establishment-Static communicates  $\mathcal{O}(n^2)$  field elements.*

PROOF: For each  $q_j(x), 1 \leq j \leq n$ , **A** sends  $n - |L_{fault}| = \mathcal{O}(n)$  values which incurs a total communication complexity of  $\mathcal{O}(n^2)$ . Also communication complexity of broadcasting  $L_{fault}$  is  $\mathcal{O}(n^2)$ .  $\square$

6.4.2. *Protocol Error-Identification-Static - A Three Phase Protocol to Identify at least  $\frac{t_b}{2}$  Byzantine Corrupted Wires Tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$*

As before, let **A** and **B** are connected by  $n = 2t_b + t_f + t_p + 1$  wires that are under the influence of  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . We now design a three phase protocol called Error-Identification-Static that has the following properties:

1. If at most  $\frac{t_b}{2}$  wires get Byzantine corrupted during first phase, then **A** securely establishes a one time pad of length  $n$  with **B** at the end of second phase.
2. If more than  $\frac{t_b}{2}$  wires get corrupted during first phase, then the pad will not be established. However, either **A** comes to know the identity of at least  $\frac{t_b}{2}$  Byzantine corrupted wires at the end of second phase or **B** comes to know the identity of at least  $\frac{t_b}{2}$  Byzantine corrupted wires at the end of third phase, depending upon the adversary behavior.

Thus the protocol creates a win-win situation against the adversary as follows: if the adversary does at most  $\frac{t_b}{2}$  Byzantine faults in the first phase, then an information theoretic secure one time pad is established between **A** and **B**. Otherwise, either **A** or **B** will come to know the identity of more than  $\frac{t_b}{2}$  Byzantine corrupted wires. Informally, the protocol works as follows: **A** selects  $n$  polynomials each of degree  $t_b + t_p$  and sends a RS codeword of length  $n$  for each of these polynomials to **B**. **B** applies *RS-DEC* to the received vectors, assuming the number of errors in the vectors to be at most  $\frac{t_b}{2}$  and tries to recover the  $n$  polynomials. If corresponding to some vector, **B** is unable to recover anything, then **B** concludes that more than  $\frac{t_b}{2}$  errors occurred in that vector. In this case, **B** sends back this vector to **A**, who after comparing it with its corresponding original codeword, finds the identity of at least  $\frac{t_b}{2} + 1$  corrupted wires. Otherwise, **B** recovers  $n$  polynomials of degree  $t_b + t_p$ , but is unable to decide about their correctness. In this case, **B** broadcasts the  $n$  error lists to **A**, that are output by applying *RS-DEC* to the codewords. **A** then verifies whether all the error lists are “good” or not. If yes, then **A** concludes that **B** has correctly recovered all the  $n$  polynomials. Otherwise, **A** identifies at least one polynomial that is *not* recovered correctly by **B** because of more than  $\frac{t_b}{2}$  faults during **Phase I**. **A** then broadcasts to **B**, the original codeword of that polynomial, generated by him during **Phase I**. This broadcast enables **B** to identify more than  $\frac{t_b}{2}$  faults after local verification at the end of **Phase III**. The protocol is now presented in Table 8.

We now proceed to formally prove the properties of protocol Error-Identification-Static.

**Protocol Error-Identification-Static**

**Phase I: (A to B) :**

**Computation by A :**

1. **A** randomly selects  $n$  polynomials  $p_1(x), \dots, p_n(x)$  over  $\mathbb{F}$ , each of degree  $t_b + t_p$ .
2. For  $j = 1, \dots, n$ , **A** computes RS codeword  $P_j = [p_{j1} \ p_{j2} \ \dots \ p_{jn}]$ , where for  $i = 1, \dots, n$ ,  $p_{ji} = p_j(\alpha_i)$ .

**Communication by A :**

1. For  $i = 1, \dots, n$ , **A** sends the  $i^{th}$  component of all  $n$  codewords, namely  $p_{1i}, p_{2i}, \dots, p_{ni}$ , over  $w_i$ .

**Phase II: (B to A) :**

**Computation by B :**

1. Let **B** receive information over wires  $w_{i_1}, \dots, w_{i_{N'}}$ , where  $N' \geq n - t_f$ .
2. For  $j = 1, \dots, n$ , let **B** receive  $p'_{ji_1}, \dots, p'_{ji_{N'}}$  over wires  $w_{i_1}, \dots, w_{i_{N'}}$  respectively. Let  $P'_j = [p'_{ji_1} \ \dots \ p'_{ji_{N'}}]$ .
3. For  $j = 1, \dots, n$ , **B** executes  $RS - DEC(N', P'_j, \frac{t_b}{2}, 0, t_b + t_p + 1)$ .

**Communication by B :**

1. If  $\exists j \in \{1, 2, \dots, n\}$  such that  $RS - DEC(N', P'_j, \frac{t_b}{2}, 0, t_b + t_p + 1)$  does not output any polynomial of degree  $t_b + t_p$ , then **B** broadcasts "ERROR" signal and vector  $P'_j$ , along with its index  $j$ .
2. If for each  $P'_j$ ,  $RS - DEC$  outputs some polynomial of degree  $t_b + t_p$  and an error list, then **B** proceeds as follows:
  - For  $j = 1, \dots, n$ , let  $Error\_List_j$  denote the error list output by  $RS - DEC$  for  $P'_j$ . Also let  $L_j$  be the number of pairs in  $Error\_List_j$ . By the property of  $RS - DEC$  (as described in section 4),  $L_j \leq \frac{t_b}{2}$  will hold, where  $\frac{t_b}{2}$  is the number of errors that  $RS - DEC$  is instructed to correct. **B** broadcasts  $Error\_List_j$  for  $j = 1, \dots, n$ .

**Computation by A at the End of Phase II :**

1. If **A** receives "ERROR" signal and index  $j$  along with  $P'_j$ , then **A** locally compares  $P'_j$  with  $P_j$  ( after restricting  $P_j$  to wires  $w_{i_1}, \dots, w_{i_{N'}}$ ), finds the identity of at least  $\frac{t_b}{2} + 1$  faulty wires which delivered incorrect components of  $P_j$  during first phase and terminates the protocol.
2. If **A** receives  $n$  error-lists and all the  $n$  error lists are "good", then **A** concludes that **B** has recovered each  $p_j(x)$ ,  $1 \leq j \leq n$  correctly and terminates the protocol. Otherwise, **A** finds at least one  $j \in \{1, 2, \dots, n\}$ , such that  $Error\_List_j$  is "bad". If there are multiple such  $j$ 's, **A** randomly selects one. In this case, **A** concludes that **B** has reconstructed  $\bar{p}_j(x) \neq p_j(x)$  and initiates **Phase III** as follows:

**Conditional Phase III: (A to B) :**

1. If **A** has identified a  $j$  such that **B** has reconstructed  $\bar{p}_j(x) \neq p_j(x)$ , then **A** broadcasts to **B** the index  $j$  and  $P_j = [p_{j1} \ p_{j2} \ \dots \ p_{jn}]$ .
2. **B** correctly receives  $P_j$ , compares it with the vector  $P'_j$  (which it had received during **Phase I**), identifies more than  $\frac{t_b}{2}$  faulty wires and terminates the protocol.

Table 8: A Three Phase Protocol to Identify More than  $\frac{t_b}{2}$  Byzantine Faults

- Lemma 13.** 1. If at most  $\frac{t_b}{2}$  Byzantine errors occur during **Phase I**, then an information theoretically secure pad  $p = [p_1(0) \ p_2(0) \ \dots \ p_n(0)]$  of length  $n$  is established between **A** and **B** at the end of **Phase II**.
2. If more than  $\frac{t_b}{2}$  Byzantine errors occur during **Phase I**, then either **A** or **B** comes to know the identity of more than  $\frac{t_b}{2}$  corrupted wires at the end of **Phase II** or **Phase III** respectively.

*Proof:* We prove the theorem for the worst case where during **Phase I**,  $t_f$  wires failed to deliver any information to **B**. Thus **B** receives information over  $N' = n - t_f = 2t_b + t_p + 1$  wires during first phase. For  $j = 1, \dots, n$ , each of the received vectors  $P'_j$  will contain  $N' = 2t_b + t_p + 1$  values, out of which at most  $t_b$  could be corrupted. Also, each  $P'_j$  corresponds to the RS codeword  $P_j$  that is encoded using polynomial  $p_j(x)$  of degree  $t_b + t_p$ . During **Phase II**, **B** tries to correct at most  $c = \frac{t_b}{2}$  and detect additional  $d = 0$  errors in each  $P'_j$  by applying *RS-DEC*. By substituting  $N' = 2t_b + t_p + 1, c = \frac{t_b}{2}, d = 0$  and  $k = t_b + t_p + 1$  in the inequality of Theorem 2, we find that *RS-DEC*( $N', P'_j, \frac{t_b}{2}, 0, t_b + t_p + 1$ ) will be able to correct at most  $\frac{t_b}{2}$  errors and detect *no* additional errors in  $P'_j$ . Moreover, **B** has no information about the exact number of Byzantine errors that occurred during **Phase I** (except that it is at most  $t_b$ ). Now there are following two cases:

**Case I: At most  $\frac{t_b}{2}$  Byzantine errors occurred during Phase I :** In this case at most  $\frac{t_b}{2}$  values in each  $P'_j$  could be corrupted. Hence for each  $P'_j$ , *RS-DEC* will output  $p_j(x)$  and a "good" error list after successfully correcting all the errors in  $P'_j$ . However, **B** will not know whether the recovered polynomials are correct or not. This is because the number of actual errors that can happen (i.e  $t'_b$ ) can be more than the error correction capability (i.e  $c$ ) of *RS-DEC*. Moreover, as *RS-DEC* has no capability of detecting additional errors (as  $d = 0$  here), **B** is not sure whether  $p_j(x)$  is the original polynomial used for encoding  $P_j$  and the error list is "good". The situation here is similar to the one that arises in Example 2. Hence to know the status of the recovered polynomials, **B** broadcasts each error list to **A** who will correctly receive them. When **A** gets the error lists from **B** and finds them to be "good", he concludes that **B** has recovered each  $p_j(x)$  correctly. Hence the vector  $p = [p_1(0) \ p_2(0) \ \dots \ p_n(0)]$  is established correctly between **A** and **B**.

The security of  $p$  follows from the fact that during **Phase I**, the adversary gets at most  $t_b + t_p$  points (by passively listening over  $t_b + t_p$  wires) on each  $p_j(x)$ , which is of degree  $t_b + t_p$ . Thus each  $p_j(0)$  is information theoretic secure. Also notice that each of the  $n$  error lists are "good", thus they leak no extra information about  $p_j(x)$ 's to  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . This is because in this case, the values in each error lists are indeed corrupted, which are already known to the adversary and hence add no extra information to the knowledge of adversary.

**Case II: More than  $\frac{t_b}{2}$  Byzantine errors occurred during Phase I :** Without loss of generality, let  $p_j(x)$  be one of the polynomials, corresponding

to which at least  $\frac{t_b}{2} + 1$  values have been corrupted by adversary during **Phase I**. Thus  $j^{\text{th}}$  received vector  $P'_j$  will have more than  $\frac{t_b}{2}$  corrupted values. So **B** will fail to correctly reconstruct  $p_j(x)$  by executing  $RS - DEC(N', P'_j, \frac{t_b}{2}, 0, t_b + t_p + 1)$ . Now there are two possible cases:

1. Suppose the values in  $P'_j$  are corrupted in such a way that  $RS - DEC$ , when applied to  $P'_j$ , fails to output any  $t_b + t_p$  degree polynomial. In this case, **B** knows that more than  $\frac{t_b}{2}$  values in  $P'_j$  are corrupted. However, he will not know the exact identity of the corrupted wires, who delivered those corrupted values. In order to facilitate **A** to find the identity of those corrupted wires, **B** broadcasts  $P'_j$  to **A**, along with "ERROR" signal and index  $j$ . Once **A** correctly receives these values and performs local comparison of  $P'_j$  with  $P_j$ , **A** will know the identity of all the corrupted wires (at least  $\frac{t_b}{2} + 1$ ) who delivered incorrect values of  $P_j$  to **B** during first phase.
2. Suppose the values in  $P'_j$  are corrupted in such a way that  $RS - DEC$ , when applied to  $P'_j$ , outputs a  $t_b + t_p$  degree polynomial  $\bar{p}_j(x)$ , along with  $Error\_List_j$ . In this case  $\bar{p}_j(x) \neq p_j(x)$ . Moreover,  $Error\_List_j$  is "bad" and contains a correct value of  $P_j$  in it (pointed as a corrupted value)<sup>13</sup>. The reason is that there are  $t_b + t_p + 1$  honest wires which will deliver correct points on  $p_j(x)$ . Among them at most  $t_b + t_p$  could lie on  $\bar{p}_j(x)$  as well, as  $p_j(x)$  and  $\bar{p}_j(x)$  (both of degree  $t_b + t_p$ ) may have at most  $t_b + t_p$  common points lying on them. But the remaining one honest value lies on *only*  $p_j(x)$  and not lies on  $\bar{p}_j(x)$ . Hence that honest value will be considered as an error location and will be included in  $Error\_List_j$ .

However as described in **Case I**, **B** will not know whether the recovered polynomial is correct or not. Hence, **B** broadcasts  $Error\_List_j$  to **A**. Once **A** correctly receives  $Error\_List_j$  and performs local verification, **A** will find  $Error\_List_j$  to be "bad" and will conclude that **B** has recovered incorrect  $p_j(x)$  because of more than  $\frac{t_b}{2}$  incorrect values in  $P'_j$ . But **A** will not know the identity of these corrupted wires. To facilitate **B** to find out the identity of corrupted wires, **A** will execute third phase, where he will broadcast  $P_j$  to **B**. After receiving  $P_j$  correctly, **B** finds the identity of corrupted wires (more than  $\frac{t_b}{2}$ ) after performing local comparison of  $P_j$  and  $P'_j$ .

This completes the proof of the lemma. □

**Lemma 14.** *The communication complexity of protocol Error-Identification-Static is  $\mathcal{O}(n^2 t_b)$ .*

*Proof:* During first phase, **A** sends a RS codeword of length  $n$  for  $n$  polynomials, thus communicating  $\mathcal{O}(n^2)$  field elements. During second phase, in the worst

---

<sup>13</sup>This case is similar to Property 4 as explained in Section 4.

case, **B** broadcasts  $n$  error-lists, each containing at most  $\frac{t_b}{2}$  pairs, thus communicating  $\mathcal{O}(n^2 t_b)$  field elements. Communication complexity of conditional third phase is  $\mathcal{O}(n^2)$  field elements. Hence overall complexity is  $\mathcal{O}(n^2 t_b)$  field elements.  $\square$

#### 6.4.3. Reducing the Communication Complexity of Protocol Error-Identification-Static

In protocol Error-Identification-Static, the most communication oriented step is in **Phase II**, where **B** may have to broadcast  $n$  error lists to reliably send them to **A**. This incurs a communication cost of  $\mathcal{O}(n^2 t_b)$ . We now present a nice trick to reduce the communication complexity of sending  $n$  error-lists from  $\mathcal{O}(n^2 t_b)$  to  $\mathcal{O}(n^2)$  during **Phase II** of protocol Error-Identification-Static, without changing the properties of the protocol.

Let  $ERROR\_List_J$  be the error-list with maximum number of pairs  $L_J$ , where  $J \in \{1, 2, \dots, n\}$ . If there are several error-lists with  $L_J$  pairs, then **B** arbitrarily selects one. **B** then broadcasts *only*  $ERROR\_List_J$  and sends the remaining error-lists concatenated into a list  $Y$  by executing protocol PRMT-Mixed( $Y, |Y|, n, t_b, t_f, L_J$ ). **A** correctly receives  $ERROR\_List_J$  and verifies whether it is "good". If it is good, then **A** concludes that **B** has correctly recovered  $p_J(x)$ . Moreover, **A** will identify  $L_J$  faulty wires from  $ERROR\_List_J$  as all the pairs listed in  $ERROR\_List_J$  are indeed corrupted values. Now from Theorem 3, protocol PRMT-Mixed will correctly deliver the list  $Y$  containing the remaining error-lists. The rest of the protocol will now be same.

On the other hand, if **A** finds that  $ERROR\_List_J$  is "bad", then **A** concludes that **B** has not recovered  $p_J(x)$  correctly. In this case, **A** fails to know  $L_J$  faults from  $ERROR\_List_J$  and hence PRMT-Mixed will fail to deliver  $Y$  correctly. But **A** identifies one polynomial, namely  $p_J(x)$ , which is not recovered correctly by **B** (due to more than  $\frac{t_b}{2}$  errors during **Phase I**). *Note that while the properties of protocol Error-Identification-Static (Lemma 13) remain intact by incorporating these changes, the communication complexity reduces to  $\mathcal{O}(n^2)$ .* We now provide the complete description of Error-Identification-Static that incorporates the above mentioned changes for attaining a communication complexity of  $\mathcal{O}(n^2)$  in Table 9.

**Lemma 15.** *The communication complexity of new Error-Identification-Static (presented in Table 9) after incorporating new steps is  $\mathcal{O}(n^2)$ .*

*Proof:* During first phase, **A** sends a RS codeword of length  $n$  for  $n$  polynomials, thus communicating  $\mathcal{O}(n^2)$  field elements. During second phase, broadcasting a *single* error-list ( $ERROR\_List_J$ ) requires communicating  $\mathcal{O}(n^2)$  field elements. From Lemma 2 and Theorem 3, sending the remaining error-lists by executing PRMT-Mixed( $Y, |Y|, n, t_b, t_f, L_J$ ) will require communication of  $\mathcal{O}\left(\frac{|Y|}{L_J} * n\right) = \mathcal{O}(n^2)$  field elements as  $|Y| \leq (n - 1) * (2L_J)$ . Communication complexity of conditional third phase is  $\mathcal{O}(n^2)$  field elements. Hence overall complexity is  $\mathcal{O}(n^2)$  field elements.  $\square$

**Protocol Error-Identification-Static**

**Phase I: (A to B)** : Same as presented in Table 8.

**Phase II: (B to A)** :

**Computation by B:** Same as presented in Table 8.

**Communication by B:**

1. If  $\exists j \in \{1, 2, \dots, n\}$  such that  $RS - DEC(N', P'_j, \frac{t_b}{2}, 0, t_b + t_p + 1)$  does not output any polynomial of degree  $t_b + t_p$ , then **B** broadcasts "ERROR" signal and vector  $P'_j$ , along with its index  $j$ .
2. If for each  $P'_j$ ,  $RS - DEC$  outputs some polynomial of degree  $t_b + t_p$  and an error list, then **B** proceeds as follows:
  - (a) For  $j = 1, \dots, n$ , let  $Error\_List_j$  denote the error list output by  $RS - DEC$  for  $P'_j$ . Also let  $L_j$  be the number of pairs in  $Error\_List_j$ . By the property of  $RS - DEC$  (as described in section 4),  $L_j \leq \frac{t_b}{2}$  will hold, where  $\frac{t_b}{2}$  is the number of errors that  $RS - DEC$  is instructed to correct.
  - (b) Let  $J \in \{1, 2, \dots, n\}$  be the smallest index, such that  $Error\_List_J$  has maximum number of pairs  $L_J$ .
  - (c) **B** concatenates all the error-lists, except  $Error\_List_J$  to form a list  $Y$ .
  - (d) **B** broadcasts  $Error\_List_J$  along with its index  $J$  to **A**.
  - (e) **B** sends the list  $Y$  by executing the protocol  $PRMT-Mixed(Y, |Y|, n, t_b, t_f, L_J)$ .

**Computation by A at the End of Phase II** :

1. If **A** receives "ERROR" signal and index  $j$  along with  $P'_j$ , then **A** locally compares  $P'_j$  with  $P_j$  ( after restricting  $P_j$  to wires  $w_{i_1}, \dots, w_{i_{N'}}$ ), finds the identity of at least  $\frac{t_b}{2} + 1$  faulty wires which delivered incorrect components of  $P_j$  during first phase and terminates the protocol.
2. If **A** receives the index  $J$  and  $Error\_List_J$ , then **A** locally verifies  $Error\_List_J$  and then performs the following steps:
  - (a) If  $Error\_List_J$  is "bad" then **A** concludes that **B** has reconstructed  $\bar{p}_J(x) \neq p_J(x)$  and executes conditional **Phase III**.
  - (b) If  $Error\_List_J$  is "good" then **A** concludes that **B** has correctly recovered  $p_J(x)$ . **A** also identifies  $L_J$  Byzantine corrupted wires from  $Error\_List_J$  and hence from Theorem 3, correctly receives  $Y$  delivered by  $PRMT-Mixed$ .
  - (c) From  $Y$ , **A** obtains the remaining  $n - 1$  error lists. **A** then checks if the remaining  $n - 1$  error-lists are "good". If yes then **A** concludes that **B** has recovered each  $p_j(x), 1 \leq j \leq n$  correctly and terminates the protocol. Otherwise, **A** finds at least one  $j \in \{1, 2, \dots, n\} \setminus \{J\}$ , such that  $Error\_List_j$  is "bad". If there are multiple such  $j$ 's, **A** randomly selects one. In this case, **A** concludes that **B** has reconstructed  $\bar{p}_j(x) \neq p_j(x)$  and initiates **Phase III**.

**Conditional Phase III: (A to B)** :

1. If **A** has identified an  $\alpha$  (notice that  $\alpha$  can be  $J$  or  $j$ ) such that **B** has reconstructed  $\bar{p}_\alpha(x) \neq p_\alpha(x)$ , then **A** broadcasts to **B** the index  $\alpha$  and  $P_\alpha = [p_{\alpha 1} \ p_{\alpha 2} \ \dots \ p_{\alpha n}]$ .
2. **B** correctly receives  $P_\alpha$ , compares it with the vector  $P'_\alpha$  (which it had received during **Phase I**), identifies more than  $\frac{t_b}{2}$  faulty wires and terminates the protocol.

Table 9: A Three Phase Protocol to Identify More than  $\frac{t_b}{2}$  Byzantine Faults with a Communication Complexity of  $\mathcal{O}(n^2)$

**Lemma 16.** *The properties given in Lemma 13 will hold for new Error-Identification-Static (presented in Table 9) even after incorporating new steps.*

PROOF: If at most  $\frac{t_b}{2}$  Byzantine errors occur during **Phase I**, then each of the  $n$  error lists will be "good". In this case, **A** on receiving  $Error\_List_J$  will identify that it is "good" and hence will know the identity of  $L_J$  Byzantine corrupted wires. So from Theorem 3, protocol PRMT-Mixed will correctly deliver the remaining  $n - 1$  error lists. **A** will find all the remaining  $n - 1$  error lists to be "good" and hence he concludes that **B** has recovered each  $p_j(x)$  correctly. Thus the vector  $p = [p_1(0)p_2(0) \dots p_n(0)]$  will be established correctly between **A** and **B**. The secrecy of  $p = [p_1(0)p_2(0) \dots p_n(0)]$  can be argued in the same way as done in Lemma 13.

Now consider the case when more than  $\frac{t_b}{2}$  wires are Byzantine corrupted during **Phase I**. Then, we have the following two cases:

1. If there exists some  $j \in \{1, 2, \dots, n\}$  such that after applying  $RS - DEC$  on  $P'_j$ , **B** does not obtain any  $t_b + t_p$  degree polynomial, then property **2** of Lemma 13 will follow from the proof of part 1 of **Case II** of Lemma 13.
2. If for each  $P'_j$ ,  $RS - DEC$  outputs a polynomial of degree  $t_b + t_p$ , then we have the following two sub-cases:

**Sub-Case I :** If  $Error\_List_J$  is "good", then it implies that **B** has correctly recovered  $p_J(x)$ . In this case, **A** on receiving  $Error\_List_J$  will also conclude the same. Moreover, since each value in  $Error\_List_J$  is indeed corrupted, the wires which delivered those values to **B** during **Phase I** are Byzantine corrupted. Thus **A** will know the identity of  $L_J$  Byzantine corrupted wires from  $Error\_List_J$ . Now, from Theorem 3, protocol PRMT-Mixed will correctly deliver the remaining  $n - 1$  error lists, concatenated to a list  $Y$ . Since more than  $\frac{t_b}{2}$  Byzantine errors has occurred during **Phase I**, at least one of the error lists in the remaining  $n - 1$  error lists, say  $Error\_List_j$ , will be "bad", which will be identified by **A**. In this case, **A** will execute the conditional **Phase III** and hence at the end of **Phase III**, **B** will know the identity of Byzantine corrupted wires (more than  $\frac{t_b}{2}$ ), which delivered incorrect values of  $P_j$  during **Phase I**. Thus property **2** of Lemma 13 will hold.

**Sub-Case II :** If  $Error\_List_J$  is "bad", then it implies that **B** has not correctly recovered  $p_J(x)$  because of more than  $\frac{t_b}{2}$  corrupted values in  $P'_j$ . In this case, **A** on receiving  $Error\_List_J$  will conclude the same and will execute conditional **Phase III**. At the end of **Phase III**, **B** will know the identity of Byzantine corrupted wires (more than  $\frac{t_b}{2}$ ), which delivered incorrect values of  $P_j$  during **Phase I**. Thus property **2** of Lemma 13 will hold in this case as well.

This completes the proof of the lemma. □

6.4.4. *Protocol 4-OPSMT-Static: A Four Phase OPSMT Tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$*

We now design a *four* phase OPSMT called 4-OPSMT-Static tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , where **S** and **R** are connected by  $n = 2t_b + t_f + t_p + 1$  wires. The protocol sends a message  $m$  containing  $n$  field elements by communicating  $\mathcal{O}(n^2)$  field elements. This also shows the sufficiency of the condition given in Theorem 10. The protocol uses Error-Identification-Static and Pad-Establishment-Static as sub-protocols. By using these two sub-protocols, **S** and **R** tries to establish an information theoretically secure pad containing  $n$  field elements. Once this is done, **S** can blind the message by X-ORing it with the pad and broadcast the blinded message to **R**. On receiving the blinded message, **R** extracts the message by X-ORing the blinded message with the pad. The protocol is presented in Table 10.

<b>Protocol 4-OPSMT-Static</b>	
<b>R</b> and <b>S</b> starts executing protocol Error-Identification-Static, where <b>Phase I</b> is initiated by <b>R</b> .	
1.	IF at the end of <b>Phase II</b> of protocol Error-Identification-Static, <b>R</b> finds that the pad $p = [p_1(0) \ p_2(0) \ \dots \ p_n(0)]$ is established securely between <b>R</b> and <b>S</b> , then <b>R</b> broadcasts “SUCCESS-R” signal to <b>S</b> in the <b>Phase III</b> . <b>S</b> on receiving the signal “SUCCESS-R”, computes $\Gamma = m \oplus p$ , broadcasts $\Gamma$ to <b>R</b> in <b>Phase IV</b> and terminates the protocol. <b>R</b> correctly receives $\Gamma$ , recovers $m = \Gamma \oplus p$ and terminates the protocol.
2.	IF at the end of <b>Phase II</b> of protocol Error-Identification-Static, <b>R</b> identifies at least $\frac{t_b}{2} + 1$ Byzantine corrupted wires, then <b>R</b> securely establishes the one time pad $q = [q_1(0) \ q_2(0) \ \dots \ q_n(0)]$ with <b>S</b> at the end of <b>Phase III</b> by executing the single phase protocol Pad-Establishment-Static. Once this is done, <b>S</b> computes $\Gamma = m \oplus q$ , broadcasts $\Gamma$ to <b>R</b> during <b>Phase IV</b> and terminates the protocol. <b>R</b> correctly receives $\Gamma$ , recovers $m = \Gamma \oplus q$ and terminates the protocol.
3.	IF conditional <b>Phase III</b> of protocol Error-Identification-Static is executed, then <b>S</b> identifies at least $\frac{t_b}{2} + 1$ Byzantine corrupted wires at the end of third phase. <b>S</b> then securely establishes the one time pad $q = [q_1(0) \ q_2(0) \ \dots \ q_n(0)]$ with <b>R</b> by executing the single phase protocol Pad-Establishment-Static during <b>Phase IV</b> . Moreover, <b>S</b> also computes $\Gamma = m \oplus q$ , broadcasts $\Gamma$ to <b>R</b> during <b>Phase IV</b> and terminates the protocol. <b>R</b> gets the pad $q$ at the end of <b>Phase IV</b> , recovers $m = \Gamma \oplus q$ and terminates the protocol.

Table 10: A Four Phase OPSMT Tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ ,  $n = 2t_b + t_f + t_p + 1$ ,  $|m| = n$

We now prove the properties of protocol 4-OPSMT-Static.

**Lemma 17 (Correctness).** *Protocol 4-OPSMT-Static correctly delivers  $m$  in four phases.*

PROOF: In the protocol,  $m$  is masked by **S** using either the pad  $p$  or  $q$ . To prove the lemma, we show that **R** will also get the same pad in four phases. In the protocol, there are following two possibilities:

**Case I: Error-Identification-Static terminates in two phases :** Here there are further two possibilities:

**Sub-Case (a): At the end of second phase, R concludes that the pad  $p$  is correctly established with S:** In this case, **R** broadcasts

“SUCCESS-R” signal to **S** during **Phase III**. So at the end of **Phase III**, **S** will know that pad  $p$  is established between him and **R**.

**Sub-Case (b): At the end of Phase II, R identifies at least  $\frac{t_b}{2} + 1$  Byzantine corrupted wires:** In this case, with the knowledge of  $\frac{t_b}{2} + 1$  Byzantine corrupted wires, **R** executes the single phase protocol Pad-Establishment-Static to establish the pad  $q$ . From Lemma 10, **S** will correctly get the pad  $q$  at the end of **Phase III**.

**Case II: Error-Identification-Static terminates in three phases :** In this case, **S** will identify at least  $\frac{t_b}{2} + 1$  Byzantine corrupted wires at the end of **Phase III** (Lemma 13). Now with the knowledge of  $\frac{t_b}{2} + 1$  Byzantine corrupted wires, **S** executes the single phase protocol Pad-Establishment-Static to establish the pad  $q$ . From Lemma 10, **R** will correctly get the pad  $q$  at the end of **Phase IV**.

This completes the proof of the lemma.  $\square$

**Lemma 18 (Security).** *In protocol 4-OPSMT-Static,  $m$  will be information theoretically secure.*

PROOF: The secrecy of  $m$  depends upon the secrecy of the pad, using which  $m$  is masked. If  $p$  is used as the masking pad, then secrecy of  $m$  follows from the secrecy of pad  $p$  (see Lemma 13). On the other hand, if  $q$  is used as the masking pad, then secrecy of  $m$  follows from the secrecy of pad  $q$  (see Lemma 11).  $\square$

**Lemma 19 (Communication Complexity).** *Communication complexity of protocol 4-OPSMT-Static is  $\mathcal{O}(n^2)$ .*

PROOF: From Lemma 15 and Lemma 12, establishing the pad by executing protocols Error-Identification-Static and Pad-Establishment-Static incurs a communication cost of  $\mathcal{O}(n^2)$ . Moreover, since the message size is  $n$ , the blinded message  $\Gamma$  will also contain  $n$  field elements and hence broadcasting it will require communicating  $\mathcal{O}(n^2)$  field elements. Thus the communication complexity of the protocol is  $\mathcal{O}(n^2)$ .  $\square$

**Theorem 12.** *Protocol 4-OPSMT-Static is an efficient OPSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ .*

PROOF: From Theorem 11, any four phase PSMT over  $n = 2t_b + t_f + t_p + 1$  wires must communicate  $\Omega(n^2)$  field elements to securely send a message containing  $n$  field elements against  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . From Lemma 19, the total communication complexity of the protocol is  $\mathcal{O}(n^2)$ . Hence protocol 4-OPSMT-Static is an OPSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . It is easy to see that both **S** and **R** performs polynomial computation in the protocol.  $\square$

**Theorem 13.** *Let **S** and **R** be connected by  $n = 2t_b + t_f + t_p + 1$  wires. Then there exists an efficient four phase OPSMT protocol which securely sends a message containing  $\ell \geq n$  field elements by communicating  $\mathcal{O}(n\ell)$  field elements against  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ .*

PROOF: In order to send the message, **S** will divide it into several sub-blocks and securely sends each sub-block by concurrently executing the protocol 4-OPSMT-Static. Since each of these protocols will take four phases, the over all protocol will terminate in four phase. Moreover, the communication complexity of the protocol will be  $\mathcal{O}\left(\frac{\ell}{n}.n^2\right) = \mathcal{O}(n\ell)$ . From Theorem 11, any four phase PSMT over  $n = 2t_b + t_f + t_p + 1$  wires must communicate  $\Omega(n\ell)$  field elements to securely send a message containing  $\ell \geq n$  field elements tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ . Thus the resultant protocol will be an efficient OPSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ .  $\square$

**Remark 2.** *Note that 4-OPSMT-Static sends only codeword of polynomials, in contrast to the existing protocol summarized in section 6.3, which sends both polynomial and its codeword. The advantage that we get by sending only codeword is that we obtain one information theoretic secure value per codeword (after some intermediate information exchanges and then applying RS decoding). Soon, we will show that this technique can be used to design OPSMT protocols even against mobile mixed adversary.*

## 7. Network Model and Definitions Used For Mobile Adversary

Till now, we have considered a static adversary, who corrupts the same set of wires in each phase of the protocol. We now move on to design PSMT protocols against mobile mixed adversary. For that, we describe the network model used in the presence of mobile mixed adversary.

As in the case of  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , we assume that **S** and **R** are two nodes in an undirected synchronous network and the network is abstracted as *wires*, where **S** and **R** are connected by  $n$  parallel and bi-directional wires. We assume that there exists an adversary  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ , who has *unbounded computing power* and controls **different** set of  $t_b, t_f$  and  $t_p$  wires (among  $n$  wires), in Byzantine, fail-stop and passive fashion respectively, in different phases of a protocol. Hence if a wire is corrupted by the adversary in Byzantine/fail-stop/passive fashion in  $i^{th}$  phase, then it is healed at the end of that phase. Hence a wire controlled by the adversary in  $i^{th}$  phase will be free from the influence of adversary in  $(i + 1)^{th}$  phase unless the adversary chooses the wire to corrupt in  $(i + 1)^{th}$  phase as well. Though  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  controls different set of wires in different phases of the protocol, it does not allow the adversary to gain any information which has previously passed (in earlier phases of the protocol) through the wires under its control in current phase. This is because the wires (and hence the nodes along these wires) erases all the local information from their memory at the end of each phase.

The mobile mixed adversary gain information from the wires in a cumulative fashion. For example, suppose during first phase of a protocol,  $\mathcal{A}_{(1,1,1)}^{mobile}$  controls  $w_1, w_2$  and  $w_3$  in Byzantine, fail-stop and passive fashion respectively in a network, where **S** and **R** are connected by wires  $w_1, w_2, \dots, w_5$ . Now suppose during second phase, it controls  $w_2, w_4$  and  $w_5$  in Byzantine, fail-stop and passive fashion respectively. Then  $w_1$  and  $w_3$  will behave correctly during second

phase and adversary has no access to the information passing through them in second phase. At the end of second phase, adversary will know the information which passed through  $w_1$  and  $w_3$  during first phase and the information that passed through  $w_2$  and  $w_5$  during second phase.

## 8. Existing Literature and Our Contributions For PSMT Tolerating Mobile Mixed Adversary

A mobile Byzantine adversary, denoted by  $\mathcal{A}_{t_b}^{mobile}$ , corrupts different set of  $t_b$  nodes in different phases of the protocol in Byzantine fashion. PSMT tolerating  $\mathcal{A}_{t_b}^{mobile}$  was studied by Srinathan et.al in [54]. They have shown that any two or more phase PSMT tolerating  $\mathcal{A}_{t_b}^{mobile}$  is possible iff  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n \geq 2t_b + 1$  wires. This shows that the *connectivity requirement for PSMT is same for both static and mobile Byzantine adversary*.

Since  $\mathcal{A}_{t_b}^{mobile}$  is stronger than  $\mathcal{A}_{t_b}^{static}$ , the lower bound on communication complexity of any multi phase PSMT tolerating  $\mathcal{A}_{t_b}^{static}$  (i.e.,  $\Omega\left(\frac{n\ell}{n-2t_b}\right)$ ) will trivially define a lower bound for multi phase PSMT tolerating  $\mathcal{A}_{t_b}^{mobile}$ . Surprisingly, in [39], Patra et. al have shown the tightness of this bound by designing a *three* phase polynomial time OPSMT protocol, which sends a message of size  $\ell$  by communicating  $\mathcal{O}(n\ell)$  field elements against  $\mathcal{A}_{t_b}^{mobile}$ , where  $\mathbf{S}$  and  $\mathbf{R}$  are connected by  $n \geq 2t_b + 1$  wires. This shows that *communication complexity for PSMT is same against both static and mobile Byzantine adversary*. Thus the results of [54] and [39] shows contradictions to the intuition that the connectivity requirement and communication complexity of PSMT protocols will be more against  $\mathcal{A}_{t_b}^{mobile}$  in comparison to  $\mathcal{A}_{t_b}^{static}$ .

### 8.1. Our Contribution in PSMT over Undirected Synchronous Networks Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$

We present the characterization, lower bound on communication complexity and protocols that matches the lower bound for PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . Specifically, we show the following:

1. Any two or more phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  is possible iff there exists  $n \geq 2t_b + t_f + t_p + 1$  wires between  $\mathbf{S}$  and  $\mathbf{R}$ .
2. Any two or more phase PSMT protocol over  $n \geq 2t_b + t_f + t_p + 1$  wires tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  must communicate  $\Omega\left(\frac{n\ell}{n-(2t_b+t_f+t_p)}\right)$  field elements to securely transmit a message containing  $\ell$  field elements. Moreover, we show that this bound is *asymptotically tight* by designing a *nine* phase OPSMT protocol that sends  $\ell$  field elements by communicating  $\mathcal{O}\left(\frac{n\ell}{n-(2t_b+t_f+t_p)}\right) = \mathcal{O}(n\ell)$  field elements, where  $n = 2t_b + t_f + t_p + 1$ . Finally, we design a *three* phase OPSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  by extending and employing an intelligent technique introduced by [25] (for Byzantine adversary).

Our results reiterate the contradiction to the intuition that *the connectivity requirement and communication complexity of PSMT protocols will be more against  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  in comparison to  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ .*

**Remark 3 (Single Phase PSMT Tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ ).** *Notice that any single phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  will also work against  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . This is because the protocol has only one phase. So Theorem 5, Theorem 6 and Theorem 8 will hold against  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  as well.*

### 8.2. Techniques Used for Designing PSMT Protocols Against $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$

The techniques used for designing three phase OPSMT against  $\mathcal{A}_{t_b}^{mobile}$  [39] cannot be extended for designing OPSMT against  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . Moreover, the techniques used for designing protocol 4-OPSMT-Static against  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  will not work against  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  (a formal discussion on this is given in Section 9.2). So to design our *nine* phase OPSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ , we describe few new techniques and sub-protocols, which can tolerate  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . Furthermore, we design a *three* phase OPSMT by extending a technique proposed in [25] for Byzantine adversary, to mixed adversary.

## 9. Multi Phase PSMT Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$

In this section, we provide the necessary and sufficient condition for the existence of any multi phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . We then prove the lower bound on the communication complexity of multi phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . We then point out why the techniques used in designing OPSMT tolerating  $\mathcal{A}_{t_b}^{mobile}$  and the techniques used in designing protocol 4-OPSMT against  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  can not be reused/extended for designing OPSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . We then design a protocol for reliable message transmission tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  and then using it as a building block, we design a *nine* phase OPSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ .

### 9.1. Characterization and Lower Bound On Communication Complexity of PSMT Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$

The characterization and lower bound on communication complexity of any multi phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  is given by the following theorem:

**Theorem 14.** *Any  $r \geq 2$  phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  is possible iff there exists  $n \geq 2t_b + t_f + t_p + 1$  wires between **S** and **R**. Moreover, any such protocol must communicate  $\Omega\left(\frac{n\ell}{n-(2t_b+t_f)}\right)$  field elements to securely send a message  $m$  containing  $\ell$  field elements.*

PROOF: Since  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  is stronger than  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ , the characterization and lower bound on communication complexity of any multi phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  will also hold against  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . Thus the theorem follows from Theorem 10 and Theorem 11. In the sequel, we design a *nine* phase OPSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  with  $n = 2t_b + t_f + t_p + 1$  in Section 9.4.  $\square$

## 9.2. Existing OPSMT Tolerating $\mathcal{A}_{t_b}^{mobile}$ and $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ and Their Limitations

In [39], the authors have designed a three phase OPSMT tolerating  $\mathcal{A}_{t_b}^{mobile}$ , where **S** and **R** are connected by  $n = 2t_b + 1$  wires. The protocol uses sub-protocol  $\Pi_{t_b}^i$ , described in Section 6.3, as a building block to establish an information theoretic secure one time pad between **S** and **R**. However, as explained in Section 6.3 (for  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ ), the extended sub-protocol of  $\Pi_{t_b}^i$  for mixed adversary, namely  $\Pi_{(t_b, t_f, t_p)}^i$  cannot be used for designing OPSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ .

The techniques used for designing protocol 4-OPSMT-Static tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  cannot be reused for designing OPSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . This is due to the failure of protocol Error-Identification-Static to achieve its properties with a communication of  $\mathcal{O}(n^2)$  in the presence of  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . Recall that in Error-Identification-Static, **B** reliably sends  $n$  error lists during second phase. This was done by broadcasting the error list with maximum number of pairs  $L_J$  and then jointly sending the remaining  $n - 1$  error lists by executing protocol PRMT-Mixed, with a block size of  $L_J$ . Also recall that when the maximum sized error list is "good", then from the error list, **A** will know the identities of  $L_J$  wires, which were Byzantine corrupted during first phase. Now since the adversary was static, *the same set of wires will be Byzantine corrupted in the second phase as well and hence **A** could neglect them.* This facilitated **A** to correctly recover the remaining  $n - 1$  error lists from the PRMT-Mixed (see Theorem 3). However, this will not work against  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . This is because *the  $L_J$  wires which were Byzantine corrupted during first phase, may not be under the control of  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  in the second phase.* In the worst case, all these  $L_J$  wires may be completely honest and hence by neglecting them, **A** will lose information sent through  $L_J$  honest wires. Thus protocol PRMT-Mixed will fail to correctly deliver the remaining  $n - 1$  error lists to **A** in the presence  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . So even when the maximum sized error list is "good", **B** will fail to reliably receive the remaining  $n - 1$  error lists. To reliably send the error lists in the presence  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ , **B** may broadcast all of them to **A**. Though broadcasting the  $n$  error lists ensures their proper delivery, it will increase the communication complexity of Error-Identification-Static from  $\mathcal{O}(n^2)$  to  $\mathcal{O}(n^2 t_b)$ . Thus the resultant PSMT protocol incorporating Error-Identification-Static will not be an OPSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . In order to deal with this problem, we design a three phase reliable message transmission protocol against  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  in the next section. Then using the protocol as a building block, we design our nine phase OPSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ .

### 9.3. Three Phase Reliable Message Transmission Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$

Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n = 2t_b + t_f + 1$  wires, under the influence of  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . We now design a *three phase reliable message transmission* protocol called 3-PRMT-Mobile, which *reliably* sends a message  $m$  containing  $n(t_b + 1)$  field elements by communicating  $\mathcal{O}(n^2)$  field elements. The protocol is given in Table 11.

Our three phase protocol works in the presence of  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  as in **Phase II** and **Phase III**,  $\mathbf{R/S}$  uses only broadcast for reliably sending some information. Broadcast sends any information reliably even in the presence of  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . We now prove the properties of protocol 3-PRMT-Mobile.

**Lemma 20 (Correctness).** *Protocol 3-PRMT-Mobile correctly delivers the message  $m$  in three phases tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ .*

PROOF: We prove the theorem for the worst case, when  $\mathbf{R}$  receives information over  $N' = n - t_f = 2t_b + 1$  wires during **Phase I**. Thus each received vector  $C'_j$  will be of length  $2t_b + 1$ . Each  $C'_j$  corresponds to a RS codeword encoded using a polynomial of degree  $\frac{t_b}{2}$ . Moreover, at most  $t_b$  errors could be present in  $C'_j$ . By putting  $N' = 2t_b + 1$ ,  $c = d = \frac{t_b}{2}$  and  $k = \frac{t_b}{2} + 1$  in the inequality of Theorem 2, we find that  $RS - DEC(N', C'_j, \frac{t_b}{2}, \frac{t_b}{2}, \frac{t_b}{2} + 1)$  will be able to correct  $\frac{t_b}{2}$  errors in  $C'_j$  and detect additional  $\frac{t_b}{2}$  errors in  $C'_j$  (if any). Now there are two possible cases:

**At most  $\frac{t_b}{2}$  Errors are Present in Each  $C'_j$**  : In this case,  $RS - DEC$  will correct all these errors and will detect no additional errors. Thus  $RS - DEC$  will correctly output each  $B_j$ . Moreover,  $\mathbf{R}$  will know that  $B_j$  outputted by  $RS - DEC$  is correct. This is because the error correction plus detection capability of  $RS - DEC$  is equal to the maximum number of errors that can happen (i.e  $c + d = t_b$ ). So  $\mathbf{R}$  will correctly recover  $m$  and will terminate the protocol by broadcasting "TERMINATE" signal<sup>14</sup>.  $\mathbf{S}$  on receiving this signal will conclude that  $\mathbf{R}$  has recovered  $m$  and hence will terminate the protocol. So, in this case, the protocol will terminate in *two* phases.

**More than  $\frac{t_b}{2}$  Errors are Present in Some  $C'_j$**  : In this case,  $RS - DEC$  will correct  $\frac{t_b}{2}$  errors in  $C'_j$  and will detect the remaining errors (which can be at most  $\frac{t_b}{2}$ ). Since  $RS - DEC$  can only detect remaining errors and has no capability of correcting them, it will not output anything. This will indicate to  $\mathbf{R}$  that more than  $\frac{t_b}{2}$  errors are present in  $C'_j$ . However,  $\mathbf{R}$  has no means to know the identity of the corrupted wires, who delivered those corrupted components of  $C'_j$ <sup>15</sup>. To know their identities,  $\mathbf{R}$  broadcasts  $C'_j$  to  $\mathbf{S}$ .

<sup>14</sup>This case is similar to Property 1 given in Section 4.

<sup>15</sup>This case is similar to Property 2 in Section 4.

**Protocol 3-PRMT-Mobile**

**Phase I: S to R :**

**Computation and Communication by S:**

1. **S** divides the message  $m$  into blocks  $B_1, B_2, \dots, B_z$ , each containing  $1 + \frac{t_b}{2}$  field elements.
2. For  $j = 1, \dots, z$ , **S** computes the  $n$  length RS codeword  $C_j = [c_{j1} \ c_{j2} \ \dots \ c_{jn}]$  corresponding to block  $B_j$ .
3. For  $i = 1, \dots, n$ , **S** sends  $i^{th}$  component of all the  $z$  codewords, namely  $c_{1i}, c_{2i}, \dots, c_{zi}$  over wire  $w_i$ .

**Phase II: R to S :**

**Computation and Communication by R:**

1. Let **R** receive information over wires  $w_{i_1}, \dots, w_{i_{N'}}$ , where  $n - t_f \leq N' \leq n$ .
2. For  $j = 1, \dots, z$ , let **R** receive  $c'_{ji_1}, \dots, c'_{ji_{N'}}$  over wires  $w_{i_1}, \dots, w_{i_{N'}}$  respectively. Let  $C'_j = [c'_{ji_1} \ \dots \ c'_{ji_{N'}}]$ .
3. For  $j = 1, \dots, z$ , **R** executes  $RS - DEC(N', C'_j, \frac{t_b}{2}, \frac{t_b}{2}, \frac{t_b}{2} + 1)$  and tries to correct  $\frac{t_b}{2}$  errors and simultaneously detect additional  $\frac{t_b}{2}$  errors in  $C'_j$ .
4. If  $\exists J \in \{1, 2, \dots, z\}$ , such that  $RS - DEC(N', C'_j, \frac{t_b}{2}, \frac{t_b}{2}, \frac{t_b}{2} + 1)$  does not output any polynomial of degree  $\frac{t_b}{2}$ , then **R** broadcasts  $C'_J$  and index  $J$  to **S**.
5. Else for each  $C'_j$ ,  $RS - DEC(N', C'_j, \frac{t_b}{2}, \frac{t_b}{2}, \frac{t_b}{2} + 1)$  outputs correct  $B_j$ . **R** recovers  $m$  by concatenating all  $B_j$ 's, broadcasts "TERMINATE" signal to **S** and terminate the protocol.

If **S** receives "TERMINATE" signal, then **S** terminates the protocol. Else **S** executes third phase as follows:

**Phase III: S to R :**

**Computation and Communication by S:**

1. **S** receives  $C'_J$  and index  $J$ . **S** locally compares  $C'_J$  with  $C_J$  and identifies more than  $\frac{t_b}{2}$  wires which were Byzantine corrupted during **Phase I**. **S** saves the identities of these wires in a list  $L_{fault}$ .
2. **S** broadcasts to **R** the list  $L_{fault}$  and terminates the protocol.

**Local Computation by R at the End of Phase III :**

1. **R** correctly receives  $L_{fault}$  and identifies  $|L_{fault}| \geq \frac{t_b}{2} + 1$  wires, which delivered incorrect values during **Phase I**.
2. For  $j = 1, \dots, z$ , **R** removes from  $C'_j$ , the  $c'_{ji}$ 's received over these corrupted wires during **Phase I**.
3. For  $j = 1, \dots, z$ , **R** executes  $RS - DEC(N' - |L_{fault}|, C'_j, t_b - |L_{fault}|, 0, \frac{t_b}{2} + 1)$ .
4. For  $j = 1, \dots, z$ ,  $RS - DEC(N' - |L_{fault}|, C'_j, t_b - |L_{fault}|, 0, \frac{t_b}{2} + 1)$  outputs  $B_j$ . **R** recovers  $m$  by concatenating all  $B_j$ 's and terminate the protocol.

Table 11: A Three Phase Reliable Message Transmission Protocol Tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ ,  $n = 2t_b + t_f + 1, |m| = nt_b$

Upon receiving  $C'_j$  and locally comparing it with  $C_j$ ,  $\mathbf{S}$  will identify at least  $\frac{t_b}{2} + 1$  Byzantine corrupted wires who had send incorrect values during **Phase I**.  $\mathbf{S}$  saves the identities in a list  $L_{fault}$ .  $\mathbf{S}$  then broadcasts  $L_{fault}$  to  $\mathbf{R}$ . On receiving  $L_{fault}$ ,  $\mathbf{R}$  removes all the components of  $C'_j$  received over the wires in  $L_{fault}$ . Thus each  $C'_j$  will now contain  $2t_b + 1 - |L_{fault}|$  components, out of which at most  $t_b - |L_{fault}| < \frac{t_b}{2}$  could be corrupted. Now putting  $N' = N' - |L_{fault}| = 2t_b + 1 - |L_{fault}|$ ,  $c = t_b - |L_{fault}|$ ,  $d = 0$  and  $k = \frac{t_b}{2} + 1$  in the inequality of Theorem 2, we find that  $RS-DEC$  will be able to correct all  $t_b - |L_{fault}|$  errors present in  $C'_j$  and will correctly output  $B_j$ . Moreover, since  $\mathbf{R}$  now knows that at most  $t_b - |L_{fault}| < \frac{t_b}{2}$  errors are present in  $C'_j$ , he concludes that outputted  $B_j$  is correct. Thus by combining all  $B_j$ 's,  $\mathbf{R}$  will recover  $m$  correctly. So, in this case, the protocol will terminate in *three* phases.

This completes the proof of the correctness.  $\square$

**Lemma 21 (Communication Complexity).** *Protocol 3-PRMT-Mobile sends  $m$  containing  $nt_b$  field elements by communicating  $\mathcal{O}(n^2)$  field elements.*

PROOF: During **Phase I**,  $\mathbf{S}$  sends an  $n$  length RS codeword for each  $B_j$  of size  $1 + \frac{t_b}{2}$ . So the total communication cost of **Phase I** is  $\mathcal{O}\left(\frac{|m|}{\frac{t_b}{2}} * n\right) = \mathcal{O}(n^2)$ , as  $|m| = nt_b$ . In the second phase,  $\mathbf{R}$  may either broadcast "TERMINATE" signal or an  $N'$  length received vector  $C'_j$ . So in the worst case, the communication cost of **Phase II** is  $\mathcal{O}(n^2)$ . If **Phase III** is executed, then  $\mathbf{S}$  broadcasts  $L_{fault}$ , where  $\frac{t_b}{2} + 1 \leq |L_{fault}| \leq t_b$ . This incurs a communication cost of  $\mathcal{O}(nt_b)$ . Thus the overall communication cost of 3-PRMT-Mobile is  $\mathcal{O}(n^2)$ .  $\square$

#### 9.4. Nine Phase OPSMT Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$

Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n = 2t_b + t_f + t_p + 1$ . We now present a *nine* phase OPSMT protocol called 9-OPSMT-Mobile, which securely sends a message containing  $\Theta(n)$  field elements by communicating  $\mathcal{O}(n^2)$  field elements against  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . The protocol uses several ideas from protocol 4-OPSMT-Static and uses protocol 3-PRMT-Mobile as a black-box. The protocol establishes an information theoretically secure one time pad of length either  $n - 1 = \Theta(n)$  or  $\frac{n}{2} = \Theta(n)$  between  $\mathbf{S}$  and  $\mathbf{R}$ , depending upon the behavior of the adversary, by communicating  $\mathcal{O}(n^2)$  field elements. Accordingly,  $\mathbf{S}$  sends a message containing either  $n - 1$  or  $\frac{n}{2}$  field elements by communicating  $\mathcal{O}(n^2)$  field elements. The protocol is given in Table 12, Table 13 and Table 14.

Note that according to the description provided in section 9.3, protocol 3-PRMT-mobile is executed with  $2t_b + t_f + 1$  wires between  $\mathbf{S}$  and  $\mathbf{R}$ . So in protocol 9-OPSMT-Mobile, 3-PRMT-mobile is executed with any set of predefined  $2t_b + t_f + 1$  wires between  $\mathbf{S}$  and  $\mathbf{R}$ . Thus  $\mathbf{S}$  and  $\mathbf{R}$  neglects a pre-determined set of  $t_p$  wires and run protocol 3-PRMT-mobile on the remaining  $2t_b + t_f + 1$  wires. This does not affect the correctness and working of protocol 9-OPSMT-Mobile.

**Protocol 9-OPSMT-Mobile**

**Phase I: S to R :**

**Computation by S:**

1. **S** selects  $n$  random polynomials  $p_1(x), \dots, p_n(x)$  over  $\mathbb{F}$ , each of degree  $t_b + t_p$ . Let for  $j = 1, \dots, n$ ,  $p_j(0) = s_j$ , where  $s_j$  is a random element in  $\mathbb{F}$ .
2. For  $j = 1, \dots, n$ , **S** computes an  $n$  length RS codeword  $C_j = [c_{j1} \ c_{j2} \ \dots \ c_{jn}]$  using polynomial  $p_j(x)$ .

**Communication by S:**

1. For  $i = 1, \dots, n$ , **S** sends the  $i^{th}$  component of all the  $n$  codewords, namely  $c_{1i}, \dots, c_{ni}$  over  $w_i$ .

**Phase II: R to S :**

**Computation by R:**

1. Let **R** receive information over wires  $w_{i_1}, \dots, w_{i_{N'}}$ , where  $n - t_f \leq N' \leq n$ .
2. For  $j = 1, \dots, n$ , let **R** receive  $c'_{ji_1}, \dots, c'_{ji_{N'}}$  over wires  $w_{i_1}, \dots, w_{i_{N'}}$  respectively. Let  $C'_j = [c'_{ji_1} \ \dots \ c'_{ji_{N'}}]$ .
3. For  $j = 1, \dots, n$ , **R** executes  $RS - DEC(N', C'_j, \frac{t_b}{2}, 0, t_b + t_p + 1)$ .
4. If  $\exists J \in \{1, 2, \dots, n\}$ , such that  $RS - DEC(N', C'_J, \frac{t_b}{2}, 0, t_b + t_p + 1)$  does not output any polynomial of degree  $t_b + t_p$ , then **R** broadcasts  $C'_J$  and index  $J$  to **S**. If **R** executes this step then the remaining protocol will follow the steps provided in Table 13.
5. Else let  $RS - DEC(N', C'_j, \frac{t_b}{2}, 0, t_b + t_p + 1)$  output polynomial  $\bar{p}_j(x)$  of degree  $t_b + t_p$ , along with error list  $Error\_List_j$  containing at most  $\frac{t_b}{2}$  pairs, for each  $j = 1, \dots, n$ . **R** then combines **only the first**  $\frac{n}{2}$  error lists and reliably sends them to **S** using three phase protocol 3-PRMT-Mobile. This will occupy **Phase II**, **Phase III** and **Phase IV**. If **R** executes this step then the remaining protocol will follow the steps provided in Table 14.

Table 12: A Nine Phase OPSMT Tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ ,  $n = 2t_b + t_f + t_p + 1$

**Theorem 15.** *Protocol 9-OPSMT-Mobile correctly and securely sends a message containing  $\Theta(n)$  field elements in at most nine phases by communicating  $\mathcal{O}(n^2)$  field elements tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ .*

*Proof:* We prove the theorem for the worst case where exactly  $t_f$  wires (probably different set) failed to deliver any information in each phase due to fail-stop corruption. Thus each vector  $C'_j$  received during first phase will be of length  $N' = n - t_f = 2t_b + t_p + 1$ . Each  $C'_j$  corresponds to a RS codeword  $C_j$  encoded using a polynomial of degree  $t_b + t_p$ . Now consider the following two cases:

**Case I: At most  $\frac{t_b}{2}$  wires are Byzantine Corrupted During Phase I:**

In this case, for each  $C'_j$ ,  $RS - DEC(N', C'_j, \frac{t_b}{2}, 0, t_b + t_p + 1)$  will output the correct  $p_j(x)$  and a corresponding "good" error list at the end of **Phase II**. But as  $RS - DEC$  does not have extra error detecting capability apart from capability

<p><b>Execution I - Protocol 9-OPSMT-Mobile Continued ...</b>  <b>(This Part Will be Executed if R Executes Step 4 During Phase II)</b></p> <p><b>Phase III: S to R :</b></p> <p><b>Computation by S:</b></p> <ol style="list-style-type: none"> <li>1. <b>S</b> correctly receives index <math>J</math> and vector <math>C'_J</math>.</li> <li>2. After locally comparing <math>C'_J</math> with the corresponding original codeword <math>C_J</math>, <b>S</b> identifies at least <math>\frac{t_b}{2} + 1</math> wires which delivered incorrect components of <math>C'_J</math> to <b>R</b> during <b>Phase I</b>. <b>S</b> saves the identity of these wires in a list <math>L_{fault}</math>.</li> <li>3. For <math>j \in \{1, 2, \dots, n\} \setminus \{J\}</math>, <b>S</b> lists all <math>c_{ji}</math>'s, which were sent during <b>Phase I</b> over <math>w_i \in L_{fault}</math>, in a list <b>ReSendValues</b>. Thus <math> \text{ReSendValues}  = (n - 1) L_{fault}  \geq (n - 1)(\frac{t_b}{2} + 1)</math>.</li> <li>4. <b>S</b> constructs a pad <math>p</math> consisting of <math>n - 1</math> <math>s_j</math>'s with <math>j \in \{1, 2, \dots, n\} \setminus \{J\}</math> where <math>s_j = p_j(0)</math>.</li> <li>5. With a message <math>m</math> containing <math>n - 1</math> field elements, <b>S</b> computes <math>\Gamma = m \oplus p</math>.</li> </ol> <p><b>Communication by S:</b></p> <ol style="list-style-type: none"> <li>1. <b>S</b> reliably sends <math>L_{fault}</math> and <math>\Gamma</math> to <b>R</b> by broadcasting it.</li> <li>2. <b>S</b> reliably re-sends the values in list <b>ReSendValues</b> to <b>R</b> by executing the <i>three phase</i> protocol 3-PRMT-Mobile. This will run in <b>Phase III</b>, <b>Phase IV</b> and <b>Phase V</b>. <b>S</b> then terminates 9-OPSMT-Mobile.</li> </ol> <p><b>Message Recovery by R (At the end of Phase V):</b></p> <ol style="list-style-type: none"> <li>1. <b>R</b> correctly receives <math>\Gamma</math>, <math>L_{fault}</math> and the values in <b>ReSendValues</b>.</li> <li>2. From <math>L_{fault}</math>, <b>R</b> identifies <math> L_{fault}  &gt; \frac{t_b}{2}</math> wires which had delivered incorrect values during <b>Phase I</b>.</li> <li>3. For <math>j \in \{1, 2, \dots, n\} \setminus \{J\}</math>, corresponding to each <math>C'_j</math>, <b>R</b> replaces the <math>c'_{ji}</math>'s, which were received during <b>Phase I</b> over <math>w_i \in L_{fault}</math>, with the corresponding actual <math>c_{ji}</math>'s from the list <b>ReSendValues</b>.</li> <li>4. After replacement, <b>R</b> knows that for each <math>j \in \{1, 2, \dots, n\} - \{J\}</math>, at most <math>t_b -  L_{fault} </math> values could be corrupted in <math>C'_j</math>. <b>R</b> executes <math>RS - DEC(N', C'_j, t_b -  L_{fault} , 0, t_b + t_p + 1)</math> to recover <math>p_j(x)</math>.</li> <li>5. Once the <math>p_j(x)</math>'s for <math>j \in \{1, 2, \dots, n\} \setminus \{J\}</math> are obtained, <b>R</b> constructs the pad <math>p</math> in the same way as done by <b>S</b>.</li> <li>6. <b>R</b> computes <math>m = \Gamma \oplus p</math> and terminates protocol 9-OPSMT-Mobile.</li> </ol>
---

Table 13: Remaining Execution of Protocol 9-OPSMT-Mobile if **R** Executes Step 4 During **Phase II**. In this case,  $|m| = n - 1$

of correcting  $\frac{t_b}{2}$  errors, **R** will not know whether reconstructed  $p_j(x)$  are correct or not. So **R** follows step 5 in **Phase II** and sends first  $\frac{n}{2}$  error lists to **S** by executing protocol 3-PRMT-Mobile. By the correctness of 3-PRMT-Mobile, **S** will correctly receive all the  $\frac{n}{2}$  error lists and will find all of them to be "good". So **S** will conclude that first  $\frac{n}{2}$  polynomials, namely  $p_1(x), \dots, p_{\frac{n}{2}}(x)$ , are recovered correctly by **R**. Hence **S** uses  $p = [p_1(0), \dots, p_{\frac{n}{2}}(0)]$  as a pad to blind a message  $m$  of size  $\frac{n}{2}$  and sends the blinded message  $\Gamma$  to **R** by broadcasting it. Once **R** receives  $\Gamma$ , he can recover the message using  $\Gamma$  and the pad  $p = [p_1(0), \dots, p_{\frac{n}{2}}(0)]$ . Thus in this case protocol 9-OPSMT-Mobile sends  $\frac{n}{2}$  field elements in five phases.

**Execution II - Protocol 9-OPSMT-Mobile Continued ...**  
**(This Part Will be Executed if R Executes Step 5 During Phase II)**

**Local Computation by S (At the end of Phase IV):**

1. **S** correctly receives the first  $\frac{n}{2}$  error lists, sent by **R** using protocol 3-PRMT-Mobile.
2. **S** then checks the status of these error lists.
  - (a) If all the  $\frac{n}{2}$  error lists are "good", then **S** concludes that **R** has correctly recovered  $p_j(x)$  for  $j = 1, \dots, \frac{n}{2}$  and an information theoretically secure pad  $p = [p_1(0) \ p_2(0) \ p_{\frac{n}{2}}(0)]$  is established with **R**. So, **S** considers a message  $m$  containing  $\frac{n}{2}$  field elements and computes  $\Gamma = m \oplus p$ .
  - (b) Else  $\exists J \in \{1, 2, \dots, \frac{n}{2}\}$ , such that  $Error\_List_J$  is "bad". In this case, **S** concludes that more than  $\frac{t_b}{2}$  values has been changed in  $J^{th}$  codeword during **Phase I**.

**Phase V: S to R:**

1. If **S** has computed  $\Gamma$ , then **S** broadcasts  $\Gamma$  and a terminating signal to **R**.
2. Else **S** broadcasts index  $J$  along with "ERROR" signal, asking **R** to broadcast the  $J^{th}$  vector  $C'_J$  as received by **R** during **Phase I**.

**Local Computation by R (At the End of Phase V):**

1. If **R** receives terminating signal and  $\Gamma$ , then **R** concludes that it has correctly recovered  $p_1(x), \dots, p_{\frac{n}{2}}(x)$  during **Phase I**. **R** then forms the pad  $p = [p_1(0) \ p_2(0) \ p_{\frac{n}{2}}(0)]$ , computes  $m = p \oplus \Gamma$  and terminates protocol 9-OPSMT-Mobile.
2. Else **R** receives index  $J$  and "ERROR" signal. In this case, **R** concludes that more than  $\frac{t_b}{2}$  corrupted values are present in  $C'_J$ . So **R** executes **Phase VI** as follows:

**Phase VI: R to S:**

1. **R** broadcasts the vector  $C'_J$  to **S**.

**Local Computation by S (At the End of Phase VI):**

1. Upon receiving  $C'_J$  and comparing it with  $C_J$ , **S** identifies more than  $\frac{t_b}{2}$  wires which were Byzantine corrupted during **Phase I** and saves them in a list  $L_{fault}$ .
2. Corresponding to each  $j \in \{\frac{n}{2} + 1, \dots, n\}$  and each  $w_i \in L_{fault}$ , **S** adds the value  $c_{ji}$  to a list **ReSendValues**.
3. With the pad  $p = [p_{\frac{n}{2}+1}(0) \ \dots \ p_n(0)]$  and a message  $m$  containing  $\frac{n}{2}$  field elements, **S** computes  $\Gamma = m \oplus p$ .

**Phase VII: S to R:**

1. **S** reliably sends  $\Gamma$  and  $L_{fault}$  to **R** by broadcasting them.
2. **S** reliably sends **ReSendValues** by executing the *three* phase protocol 3-PRMT-Mobile and terminates protocol 9-OPSMT-Mobile. This will run in **Phase VII**, **Phase VIII** and **Phase IX**.

**Message Recovery by R at the End of Phase IX:**

1. **R** recovers  $m$  in the same way as in **Execution I**, given in Table 13. The only difference is that **R** performs the computation with respect to vectors  $C'_{\frac{n}{2}+1}, \dots, C'_n$ .

Table 14: Remaining Execution of Protocol 9-OPSMT-Mobile if **R** Executes Step 5 During **Phase II**. In this Case,  $|m| = \frac{n}{2}$ .

**Case II: More than  $\frac{t_b}{2}$  wires are Byzantine Corrupted During Phase I:**

This case may lead to further two cases:

- (1)  $RS - DEC$  outputs some polynomial of degree  $t_b + t_p$  for each  $C'_j$
- (2) There exists a  $J \in \{1, 2, \dots, n\}$  for which  $RS - DEC$  fails to output any polynomial.

While in (1), occurrence of more than  $\frac{t_b}{2}$  faults cannot be immediately detected by  $\mathbf{R}$  (as  $RS - DEC$  is applied with  $d = 0$ ), in (2) it can be immediately detected by  $\mathbf{R}$ . Now in (1), if more than  $\frac{t_b}{2}$  Byzantine errors occur in the codewords of *only* last  $\frac{n}{2}$  polynomials i.e for  $p_J(x)$  such that  $\frac{n}{2} + 1 \leq J \leq n$  (this implies that at most  $\frac{t_b}{2}$  Byzantine errors took place in the first  $\frac{n}{2}$  codewords), then the proof is same as in **Case I**. On the other hand, if more than  $\frac{t_b}{2}$  faults occurs for  $J^{th}$  codeword, where  $J \in \{1, 2, \dots, \frac{n}{2}\}$ , then the proof is given below.

1. **During Phase II,  $\mathbf{R}$  reconstructs  $\bar{p}_J(x) \neq p_J(x)$ ,  $J \in \{1, 2, \dots, \frac{n}{2}\}$ :** In this case,  $Error - List_J$  is a "bad" error list. Since  $\mathbf{R}$  reliably sends back first  $\frac{n}{2}$  error lists using 3-PRMT-Mobile,  $\mathbf{S}$  correctly receives  $Error - List_J$  and finds it to be "bad".  $\mathbf{S}$  concludes that  $\mathbf{R}$  has reconstructed some  $\bar{p}_J(x) \neq p_J(x)$ . So  $\mathbf{S}$  asks  $\mathbf{R}$  to broadcast the  $J^{th}$  vector  $C'_J$ , as received during **Phase I**. On receiving  $C'_J$ ,  $\mathbf{S}$  compares it with its corresponding original codeword  $C_J$  and identifies  $|L_{fault}| \geq \frac{t_b}{2} + 1$  wires which delivered incorrect values to  $\mathbf{R}$  during **Phase I**. Now by executing 3-PRMT-Mobile,  $\mathbf{S}$  re-sends the components of the last  $\frac{n}{2}$  codewords, which were sent through these corrupted wires during **Phase I**.  $\mathbf{S}$  also broadcasts the identity of these corrupted wires. *Note that re-sending these values, does not leak any additional information about the last  $\frac{n}{2}$   $p_j(x)$ 's to  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  as the adversary already knew these values during **Phase I**.* But now with the new values received,  $\mathbf{R}$  have  $N' = 2t_b + t_p + 1$  components for each of the last  $\frac{n}{2}$  vectors and at most  $t_b - |L_{fault}| \leq \frac{t_b}{2} - 1$  of these  $N'$  components could be corrupted. By substituting  $N' = 2t_b + t_p + 1$ ,  $c = t_b - |L_{fault}|$ ,  $d = 0$  and  $k = t_b + t_p + 1$  in the inequality of Theorem 2, we find that  $RS - DEC(N', C'_j, t_b - |L_{fault}|, 0, t_b + t_p + 1)$ , can correct all the remaining  $t_b - |L_{fault}| < \frac{t_b}{2}$  errors present in  $C'_j$  and can output the corresponding polynomial  $p_j(x)$  where  $j \in \{\frac{n}{2} + 1, \dots, n\}$ .  $\mathbf{R}$  then considers the constant term of these last  $\frac{n}{2}$   $p_j(x)$ 's as the secret pad established with  $\mathbf{S}$ . The secrecy of the pad follows from the fact that at any stage of the execution,  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  will not get more than  $t_b + t_p$  points on the last  $\frac{n}{2}$   $p_j(x)$ 's, each of which are of degree  $t_b + t_p$ . Once  $\mathbf{R}$  obtains the pad, he can compute the message from the blinded message  $\Gamma$ . Thus in this case protocol 9-OPSMT-Mobile sends  $\frac{n}{2}$  field elements in nine phases.
2. **During Phase II,  $\mathbf{R}$  is Unable to Recover Some  $\bar{p}_J(x)$  Where  $J \in \{1, \dots, \frac{n}{2}\}$ :** In this case  $\mathbf{R}$  broadcasts only the  $J^{th}$  received codeword  $C'_J$ , from which  $\mathbf{S}$  (after local verification) identifies at least  $\frac{t_b}{2} + 1$  wires, which delivered incorrect values to  $\mathbf{R}$  during **Phase I**. Now the rest of the proof is same as in the above case. The only difference is that here a pad of length  $n - 1$  will be established between  $\mathbf{S}$  and  $\mathbf{R}$ .

In protocol 9-OPSMT-Mobile, **S** communicates  $\mathcal{O}(n^2)$  field elements for sending  $n$  codewords during the first phase. In second phase **R** either sends a codeword or  $\frac{n}{2}$  error lists each of size at most  $\frac{tb}{2}$ . Sending the codeword by broadcasting requires  $\mathcal{O}(n^2)$  communication complexity. On the other hand, sending  $\frac{n}{2}$  error lists each of size at most  $\frac{tb}{2}$  (so total  $\frac{n}{2} * \frac{tb}{2} = \mathcal{O}(nt_b)$ ) using 3-PRMT-Mobile also requires a communication of  $\mathcal{O}(n^2)$  field elements from Lemma 21. Similarly, re-sending  $\mathcal{O}(nt_b)$  values corresponding to codewords by executing 3-PRMT-Mobile requires communicating  $\mathcal{O}(n^2)$  field elements. It is easy to see that no more than  $\mathcal{O}(n^2)$  field elements are communicated in any other phase as well. Hence the overall communication complexity of the protocol is  $\mathcal{O}(n^2)$ .  $\square$

**Theorem 16.** *Protocol 9-OPSMT-Mobile is an OPSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ .*

PROOF: From Theorem 14, any nine phase PSMT over  $n = 2t_b + t_f + t_p + 1$  wires must communicate  $\Omega(n^2)$  field elements to securely send a message containing  $\Theta(n)$  field elements tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . Since the total communication complexity of protocol 9-OPSMT-Mobile is  $\mathcal{O}(n^2)$ , protocol 9-OPSMT-Mobile is an OPSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ .

**Theorem 17.** *Let **S** and **R** be connected by  $n = 2t_b + t_f + t_p + 1$  wires. Then there exists an efficient nine phase OPSMT protocol which securely sends a message containing  $\ell = \Omega(n)$  field elements by communicating  $\mathcal{O}(n\ell)$  field elements against  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ .*

PROOF: In order to send the message, **S** will divide it into several sub-blocks and securely sends each sub-block by concurrently executing the protocol 9-OPSMT-Mobile. Since each of these protocols will take at most nine phases, the over all protocol will terminate in nine phases. Moreover, the communication complexity of the protocol will be  $\mathcal{O}\left(\frac{\ell}{n} \cdot n^2\right) = \mathcal{O}(n\ell)$ . From Theorem 14, any nine phase PSMT over  $n = 2t_b + t_f + t_p + 1$  wires must communicate  $\Omega(n\ell)$  field elements to securely send a message containing  $\ell = \Omega(n)$  field elements tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . Thus the resultant protocol will be an efficient OPSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ .  $\square$

## 10. OPSMT Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ with Reduced Phase Complexity

Till now, we have designed a *four* phase OPSMT and *nine* phase OPSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  and  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  respectively using some interesting properties of RS codes. We now design a *three* phase OPSMT called 3-OPSMT. The striking feature of this protocol is that it will work against both  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$  as well as against  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . To design the protocol we use some more interesting properties of RS codes, which are described in the sequel.

We now describe the notion of Pseudo-Basis and Pseudo-Dimension, a novel and interesting concept, introduced by Kurosawa et.al. in [25]. These notions were introduced with respect to  $\mathcal{A}_{t_b}^{static}$  to design a two phase polynomial time OPSMT protocol tolerating  $\mathcal{A}_{t_b}^{static}$ . We now extend these notions for mixed adversary and then use them to design protocol 3-OPSMT in the next section.

**Assumption 1 (Assumption Used in this Discussion and in Protocol 3-OPSMT).**

Without loss of generality, for the ease of exposition, we use the following assumption in the following discussion and also in protocol 3-OPSMT: if  $\mathbf{S}$  ( $\mathbf{R}$ ) is expecting some information in some specific format from  $\mathbf{R}$  ( $\mathbf{S}$ ) along a wire in a phase and if no/syntactically incorrect information comes, then  $\mathbf{S}$  ( $\mathbf{R}$ ) substitutes some default value(s) from  $\mathbb{F}$  in the desired format and proceeds with his computation. Thus we separately do not consider the case when no/syntactically incorrect information is received along a wire.

Now let  $\mathbf{A}$  and  $\mathbf{B}$  be two specific nodes ( $\mathbf{A}$  ( $\mathbf{B}$ ) can be  $\mathbf{S}$  or  $\mathbf{R}$ ) which are connected by  $n = 2t_b + t_f + t_p + 1$  wires denoted by  $w_1, \dots, w_n$ , of which at most  $t_b$ ,  $t_f$  and  $t_p$  can be corrupted in Byzantine, fail-stop and passive fashion respectively. Let  $\mathcal{C}$  be the set of all possible  $n$  length RS codewords over  $\mathbb{F}$ , which are RS encoded using polynomials of degree  $t_b + t_p$  over  $\mathbb{F}$ . So the hamming distance [27] of code  $\mathcal{C}$  is  $n - (t_b + t_p) = t_b + t_f + 1$ . We may call the individual codewords in  $\mathcal{C}$  as  $n$ -dimensional vectors. But it should be noted that any  $n$  length codeword is an  $n$  length vector but the reverse is not true.

If  $\mathbf{A}$  sends several codewords, say  $\gamma$  codewords  $C_1, \dots, C_\gamma \in \mathcal{C}$  over the  $n$  wires by transmitting  $i^{\text{th}}$  component of all the codewords over  $i^{\text{th}}$  wire  $w_i$ , then the locations at which Byzantine and fail-stop errors occur in these codewords are not random. This is because for all the codewords, the Byzantine and fail-stop errors always occur at the same  $t_b + t_f$  (or a smaller subset) locations. The concept of pseudo-basis is based on this simple and interesting observation.

Let  $\mathbf{B}$  receive vectors  $Y_1 \dots, Y_\gamma$  over these wires, such that for  $i = 1, \dots, \gamma$ ,

$$Y_i = C_i + E_i, \quad (2)$$

where

$$E_i = (e_{i1}, \dots, e_{in})$$

is an error vector introduced by the adversary. Since at most  $t_b$  Byzantine and at most  $t_f$  fail-stop corruptions could occur, each  $E_i$  may have at most  $t_b + t_f$  non-zero components. Let

$$\text{support}(E_i) = \{j \mid e_{ij} \neq 0\}. \quad (3)$$

Then there exists a set  $\{w_{j_1}, \dots, w_{j_{(t_b+t_f)}}\}$  of wires that are Byzantine and fail-stop corrupted, such that each error vector  $E_i$  satisfies

$$\text{support}(E_i) \subseteq \{j_1, \dots, j_{(t_b+t_f)}\} \quad (4)$$

This means that the space  $E$  spanned by the error vectors  $E_1, \dots, E_\gamma$  has dimension at most  $t_b + t_f$ . The notion of pseudo-basis exploits this idea.

### 10.1. Pseudo-Basis and Pseudo-Dimension

Let  $\mathcal{V}$  denote the  $n$ -dimensional vector space over  $\mathbb{F}$ . For  $i = 1, \dots, \gamma$ , suppose that the receiver received vector  $Y_i$ , such that

$$Y_i = C_i + E_i, \quad (5)$$

where  $C_i \in \mathcal{C}$  is a codeword that the sender sent and  $E_i$  is the error vector caused by the adversary. We say that  $\{E_1, \dots, E_\gamma\}$  is the *real error vector set* of  $\mathcal{Y} = \{Y_1, \dots, Y_\gamma\}$ . We also say that  $\mathcal{E}$  is the *real error vector space* if it is spanned by the real error vector set  $\{E_1, \dots, E_\gamma\}$ .

For two vectors  $V_1, V_2 \in \mathcal{V}$ , we write

$$V_1 = V_2 \text{ mod } \mathcal{C} \quad (6)$$

if  $V_1 - V_2 \in \mathcal{C}$ . Notice that for  $i = 1, \dots, \gamma$ , for every triplet  $(Y_i, C_i, E_i)$ ,

$$Y_i = E_i \text{ mod } \mathcal{C} \quad (7)$$

holds, as  $Y_i - E_i = C_i \in \mathcal{C}$ . We say that  $\{\mathcal{E}_1, \dots, \mathcal{E}_\gamma\}$  is an *admissible error vector set* of  $\mathcal{Y}$  if each  $\mathcal{E}_i$  satisfies  $Y_i = C_i + \mathcal{E}_i$ , for some codeword  $C_i \in \mathcal{C}$  and

$$|\text{support}(\mathcal{E}_1) \cup \dots \cup \text{support}(\mathcal{E}_\gamma)| \leq t_b + t_f \quad (8)$$

We say that  $\mathcal{E}$  is an *admissible error vector space* of  $\mathcal{Y}$  if it is spanned by an admissible error vector set  $\{\mathcal{E}_1, \dots, \mathcal{E}_\gamma\}$ . Notice that for a given  $\mathcal{Y}$ , there exists a unique real error vector set and real error vector space, while there may exist several admissible error vector sets and corresponding admissible error vector spaces. Also notice that the real error vector set (real error vector space) is also an admissible error vector set (admissible error vector space) but the reverse may not be true. Even though an admissible error vector set  $\{\mathcal{E}_1, \dots, \mathcal{E}_\gamma\}$  for a given  $\mathcal{Y}$  may not be unique, the following results hold for any admissible error vector set.

We begin with the definition of *linearly pseudo-express*.

**Definition 15 (Linearly Pseudo-Express).** *We say that a vector  $Y \in \mathcal{Y}$  is linearly pseudo-expressed by  $\{B_1, \dots, B_k\}$  if there exists some  $\alpha = (a_1, \dots, a_k)$ , such that*

$$Y = a_1 B_1 + \dots + a_k B_k \text{ mod } \mathcal{C}$$

We now state the following lemma:

**Lemma 22.** *Let  $\{\mathcal{E}_1, \dots, \mathcal{E}_\gamma\}$  be an admissible error vector set of  $\mathcal{Y}$ . Then  $\mathcal{E}_i$  is linearly expressed by  $\{\mathcal{E}_{j_1}, \dots, \mathcal{E}_{j_k}\}$  iff  $Y_i$  is linearly pseudo-expressed by  $\{Y_{j_1}, \dots, Y_{j_k}\}$ .*

PROOF: Let  $\mathcal{E}_i$  be linearly expressed by  $\{\mathcal{E}_{j_1}, \dots, \mathcal{E}_{j_k}\}$ . This implies that

$$\mathcal{E}_i = a_1 \mathcal{E}_{j_1} + \dots + a_k \mathcal{E}_{j_k}$$

for some  $a_1, \dots, a_k$ . Since  $\{\mathcal{E}_1, \dots, \mathcal{E}_\gamma\}$  is an admissible error vector set of  $\mathcal{Y}$ , it implies that for each  $i$ ,  $Y_i = C_i + \mathcal{E}_i$ , where  $C_i$  is some codeword. Then in mod  $\mathcal{C}$ , we have:

$$\begin{aligned}
& Y_i - (a_1 Y_{j_1} + \dots + a_k Y_{j_k}) \\
&= (\mathcal{C}_i + \mathcal{E}_i) - a_1(\mathcal{C}_{j_1} + \mathcal{E}_{j_1}) - \dots - a_k(\mathcal{C}_{j_k} + \mathcal{E}_{j_k}) \\
&= (\mathcal{C}_i - a_1 \mathcal{C}_{j_1} - \dots - a_k \mathcal{C}_{j_k}) + (\mathcal{E}_i - a_1 \mathcal{E}_{j_1} - \dots - a_k \mathcal{E}_{j_k}) \\
&= (\mathcal{C}_i - a_1 \mathcal{C}_{j_1} - \dots - a_k \mathcal{C}_{j_k}) + 0 \\
&= (\mathcal{C}_i - a_1 \mathcal{C}_{j_1} - \dots - a_k \mathcal{C}_{j_k})
\end{aligned}$$

Now notice that  $\mathcal{C}_i, \mathcal{C}_{j_1}, \dots, \mathcal{C}_{j_k}$  are valid RS codewords belonging to  $\mathcal{C}$  and hence are encoded using polynomials of degree  $t_b + t_p$ . So from the linearity property of polynomials,  $\mathcal{C}_i - a_1 \mathcal{C}_{j_1} - \dots - a_k \mathcal{C}_{j_k}$  will also be a valid RS codeword belonging to  $\mathcal{C}$  and hence is encoded using polynomial of degree  $t_b + t_p$ . So  $\mathcal{C}_i - a_1 \mathcal{C}_{j_1} - \dots - a_k \mathcal{C}_{j_k} \in \mathcal{C}$ . Therefore,  $Y_i - (a_1 Y_{j_1} + \dots + a_k Y_{j_k}) \in \mathcal{C}$ . Thus if  $\mathcal{E}_i$  is linearly expressed by  $\{\mathcal{E}_{j_1}, \dots, \mathcal{E}_{j_k}\}$  then  $Y_i$  is linearly pseudo-expressed by  $\{Y_{j_1}, \dots, Y_{j_k}\}$ .

Next suppose that  $Y_i$  is linearly pseudo-expressed by  $\{Y_{j_1}, \dots, Y_{j_k}\}$ . Then in mod  $\mathcal{C}$ , we have

$$Y_i = (a_1 Y_{j_1} + \dots + a_k Y_{j_k}) \text{ mod } \mathcal{C} \quad (9)$$

for some non-zero  $a_1, \dots, a_k$ . The above equation can be written as

$$\begin{aligned}
& (\mathcal{C}_i + \mathcal{E}_i) = \{a_1(\mathcal{C}_{j_1} + \mathcal{E}_{j_1}) + \dots + a_k(\mathcal{C}_{j_k} + \mathcal{E}_{j_k})\} \text{ mod } \mathcal{C} \\
& \Rightarrow (\mathcal{C}_i - a_1 \mathcal{C}_{j_1} - \dots - a_k \mathcal{C}_{j_k}) + (\mathcal{E}_i - a_1 \mathcal{E}_{j_1} - \dots - a_k \mathcal{E}_{j_k}) \in \mathcal{C}
\end{aligned}$$

Now  $\mathcal{C}_i, \mathcal{C}_{j_1}, \dots, \mathcal{C}_{j_k}$  are valid RS codewords, encoded using polynomials of degree  $t_b + t_p$ . So from the linearity property of polynomials, we have

$$(\mathcal{C}_i - a_1 \mathcal{C}_{j_1} - \dots - a_k \mathcal{C}_{j_k}) \in \mathcal{C} \quad (10)$$

The linearity property of polynomials also implies:

$$(\mathcal{E}_i - a_1 \mathcal{E}_{j_1} - \dots - a_k \mathcal{E}_{j_k}) \in \mathcal{C}$$

as the sum of a valid RS codeword (encoded using polynomial of degree  $t_b + t_p$ ) with another vector can be a valid RS codeword (encoded using polynomial of degree  $t_b + t_p$ ) only if the other vector is also a valid RS codeword (encoded using polynomial of degree  $t_b + t_p$ ). As the number of non-zero components in  $\mathcal{E}_i, \mathcal{E}_{j_1}, \dots, \mathcal{E}_{j_k}$  is at most  $t_b + t_f$ , the vector  $(\mathcal{E}_i - a_1 \mathcal{E}_{j_1} - \dots - a_k \mathcal{E}_{j_k})$  will have at least  $n - (t_b + t_f) \geq t_b + t_p + 1$  zero components. However, in  $\mathcal{C}$ , there is only one codeword, namely 0-codeword (i.e. an  $n$  length tuple containing all 0's)  $(0, \dots, 0)$ , which has at least  $t_b + t_p + 1$  zero components. This is because each element of  $\mathcal{C}$  represents  $n$  distinct points on a  $t_b + t_p$  degree polynomial and  $t_b + t_p + 1$  zero's uniquely define the zero polynomial  $f(x) = \sum_{i=0}^{t_b+t_p} 0x^i$ . These two facts together implies that the vector  $(\mathcal{E}_i - a_1 \mathcal{E}_{j_1} - \dots - a_k \mathcal{E}_{j_k})$  is an all zero vector (0-codeword) which further implies that

$$\mathcal{E}_i = a_1 \mathcal{E}_{j_1} + \dots + a_k \mathcal{E}_{j_k}$$

This means that if  $Y_i$  is linearly pseudo-expressed by  $\{Y_{j_1}, \dots, Y_{j_k}\}$ , then  $\mathcal{E}_i$  is linearly expressed by  $\{\mathcal{E}_{j_1}, \dots, \mathcal{E}_{j_k}\}$ .  $\square$

We next define *pseudo-span*.

**Definition 16 (Pseudo-Span [25]).** We say that  $\{Y_{j_1}, \dots, Y_{j_k}\} \subset \mathcal{Y}$  pseudo-spans  $\mathcal{Y}$  if each  $Y_i \in \mathcal{Y}$  can be linearly pseudo-expressed by  $\{Y_{j_1}, \dots, Y_{j_k}\}$ . That is, each  $Y_i = (a_1 Y_{j_1} + \dots + a_k Y_{j_k}) \bmod \mathcal{C}$ , for some  $a_1, \dots, a_k$ .

**Definition 17 (Pseudo-Basis [25]).** We say that  $\{Y_{j_1}, \dots, Y_{j_k}\} \subset \mathcal{Y}$  is a pseudo-basis of  $\mathcal{Y}$  if it is the minimum subset of  $\mathcal{Y}$  which pseudo-spans  $\mathcal{Y}$ .

**Definition 18 (Pseudo-Dimension [25]).** If  $\{Y_{j_1}, \dots, Y_{j_k}\} \subset \mathcal{Y}$  is a pseudo-basis of  $\mathcal{Y}$  and  $k = |\{Y_{j_1}, \dots, Y_{j_k}\}|$ , then we say that  $\mathcal{Y}$  has pseudo-dimension  $k$ .

We now prove the following theorem:

**Theorem 18.** Let  $\{\mathcal{E}_1, \dots, \mathcal{E}_\gamma\}$  be an admissible error vector set of  $\mathcal{Y}$ . Then  $\mathcal{B}_e = \{\mathcal{E}_{j_1}, \dots, \mathcal{E}_{j_k}\} \subset \{\mathcal{E}_1, \dots, \mathcal{E}_\gamma\}$  is a basis of the admissible error vector space  $\mathcal{E}$  iff  $\mathcal{B}_y = \{Y_{j_1}, \dots, Y_{j_k}\} \subset \mathcal{Y}$  is a pseudo-basis of  $\mathcal{Y}$  (Note that  $\mathcal{B}_e$  and  $\mathcal{B}_y$  have the same indices). In particular, the pseudo-dimension of  $\mathcal{Y}$  is equal to the dimension of  $\mathcal{E}$ .

PROOF: Suppose that  $\mathcal{B}_e$  is a basis of  $\mathcal{E}$ . This implies that  $\mathcal{B}_e$  is the minimum set which spans  $\mathcal{E}$ . Since  $\mathcal{B}_e$  spans  $\mathcal{E}$ , from Lemma 22,  $\mathcal{B}_y$  pseudo-spans  $\mathcal{Y}$ . Next we show that  $\mathcal{B}_y$  is the minimum subset of  $\mathcal{Y}$  which pseudo-spans  $\mathcal{Y}$ . On the contrary, assume that  $\mathcal{B}_y$  is not minimum. That is, suppose that there exists a smaller subset of  $\mathcal{Y}$  which pseudo-spans  $\mathcal{Y}$ . Then from Lemma 22, the corresponding subset of  $\{\mathcal{E}_1, \dots, \mathcal{E}_\gamma\}$  also spans  $\mathcal{E}$ . However, this contradicts the fact that  $\mathcal{B}_e$  is a basis of  $\mathcal{E}$ . This implies that  $\mathcal{B}_y$  is the minimum subset of  $\mathcal{Y}$  which pseudo-spans  $\mathcal{Y}$ , which further implies that  $\mathcal{B}_y$  is the pseudo-basis of  $\mathcal{Y}$ .

Similarly, if  $\mathcal{B}_y$  is a pseudo-basis of  $\mathcal{Y}$  then  $\mathcal{B}_e$  is a basis of  $\mathcal{E}$ . Hence the pseudo-dimension of  $\mathcal{Y}$  is equal to the dimension of  $\mathcal{E}$ .  $\square$

Since the real error vector set is also an admissible error vector set, we obtain the following corollary of Theorem 18.

**Corollary 1.** Let  $\{E_1, \dots, E_\gamma\}$  be the real error vector set of  $\mathcal{Y}$ . If  $\mathcal{B}_y = \{Y_{j_1}, \dots, Y_{j_k}\} \subset \mathcal{Y}$  is a pseudo-basis of  $\mathcal{Y}$ , then  $\mathcal{B}_e = \{E_{j_1}, \dots, E_{j_k}\} \subset \{E_1, \dots, E_\gamma\}$  is a basis of the real error vector space.

Let  $\{E_1, \dots, E_\gamma\}$  be the real error vector set of  $\mathcal{Y}$  and let  $\{C_1, \dots, C_\gamma\}$  be the corresponding original codewords which  $\mathbf{A}$  sent. Then define

$$CORRUPTED = \cup_{i=1}^{\gamma} support(E_i) \quad (11)$$

Then *CORRUPTED* is the set of wires that the adversary has corrupted in Byzantine and fail-stop fashion. We now state the following important theorem:

**Theorem 19.** Let  $\mathcal{B}_y = \{Y_{j_1}, \dots, Y_{j_k}\}$  be the pseudo-basis of  $\mathcal{Y}$  and let  $C_{j_1}, \dots, C_{j_k}$  be the corresponding original codewords. Then

$$CORRUPTED = \cup_{i=1}^{i=k} \text{support}(Y_{j_i} - C_{j_i})$$

PROOF: From the definition of *CORRUPTED*, we get

$$CORRUPTED = \cup_{i=1}^{\gamma} \text{support}(E_i)$$

From Corollary 1, since  $\{Y_{j_1}, \dots, Y_{j_k}\}$  is the pseudo-basis of  $\mathcal{Y}$ , it implies that  $\{E_{j_1}, \dots, E_{j_k}\}$  is the basis of real error vector space. This implies that

$$\cup_{i=1}^{\gamma} \text{support}(E_i) = \cup_{i=1}^{i=k} \text{support}(E_{j_i})$$

The above relationship further implies that

$$\begin{aligned} CORRUPTED &= \cup_{i=1}^{\gamma} \text{support}(E_i) \\ &= \cup_{i=1}^{i=k} \text{support}(E_{j_i}) \\ &= \cup_{i=1}^{i=k} \text{support}(Y_{j_i} - X_{j_i}) \end{aligned}$$

□

**Theorem 20.** The pseudo-dimension of  $\mathcal{Y}$  is at most  $t_b + t_f$ .

PROOF: The dimension of the real error vector space is at most  $t_b + t_f$  because the adversary can Byzantine and fail-stop corrupt at most  $t_b$  and  $t_f$  wires respectively. Hence from Theorem 18, the pseudo-dimension of  $\mathcal{Y}$  is at most  $t_b + t_f$ . □

From the above discussion, we find that if  $\mathbf{B}$  can correctly find the pseudo-basis of  $\mathcal{Y}$  and reliably sends it back to  $\mathbf{A}$ , then  $\mathbf{A}$  can identify the wires which are Byzantine and fail-stop corrupted after doing the local computation. In the next section, we present a polynomial time algorithm, which allows  $\mathbf{B}$  to find the pseudo-basis of  $\mathcal{Y}$ .

### 10.2. How to Find Pseudo-Basis

In [25], the authors have presented a polynomial time algorithm to find pseudo-basis against  $\mathcal{A}_{t_b}^{static}$ . We now extend the algorithm against mixed adversary. We begin with the definition of *linearly pseudo-express*.

We first present a polynomial time algorithm which checks whether  $Y$  can be linearly pseudo-expressed by  $\{B_1, \dots, B_k\}$ . For a non-zero  $\beta = (a_1, \dots, a_k) \in \mathbb{F}^k$ , we define  $X(\beta)$  as

$$X(\beta) = Y - (a_1 B_1 + \dots + a_k B_k). \quad (12)$$

It is clear that  $Y$  can be linearly pseudo-expressed by  $\{B_1, \dots, B_k\}$  iff there exists some non-zero  $\beta$  such that  $X(\beta) \in \mathcal{C}$ . The algorithm for checking whether

$Y$  can be linearly pseudo-expressed by  $\{B_1, \dots, B_k\}$  is presented in Table 15. The algorithm will output YES iff  $X(\beta) \in \mathcal{C}$  for some non-zero  $\beta$ . In the algorithm, each  $x_j(\beta)$  will be a linear expression of  $(a_1, \dots, a_k)$ , as  $Y, \mathcal{B}$  are known and  $\beta$  is unknown. Similarly, each coefficient of  $f_\beta(x)$  will be a linear expression of  $(a_1, \dots, a_k)$ . Hence each  $f_\beta(\alpha_j) = x_j(\beta)$  will give a linear equation on  $(a_1, \dots, a_k)$ . So we will get  $n - (t_b + t_p + 1) \geq t_b + t_f$  linear equations in  $(a_1, \dots, a_k)$ . Moreover, in our context,  $k$  will be at most  $t_b + t_f$ , as will be shown in the sequel. Thus if  $Y$  can be indeed linearly pseudo expressed by  $\mathcal{B}$ , then after solving these linear equations, some non-zero solution for  $\beta = (a_1, \dots, a_k)$  will be obtained and the algorithm will output YES.

<b>Algorithm Pseudo-Linear-Express</b> ( $Y, \mathcal{B} = \{B_1, \dots, B_k\}$ )	
1. Let $\beta = (a_1, \dots, a_k)$ , where $a_1, \dots, a_k$ are unknown variables.	
2. Let $X(\beta) = (x_1(\beta), \dots, x_n(\beta))$ , where $X(\beta)$ is given by Equation 12.	
3. By using Lagrange interpolation, construct a polynomial $f_\beta(x)$ of degree $t_b + t_p$ such that	
	$f_\beta(\alpha_i) = x_i(\beta)$
	for $i = 1, \dots, t_b + t_p + 1$ .
4. Output YES iff the following set of linear equations on $\beta$ has a non-zero solution:	
	$f_\beta(\alpha_{t_b+t_p+2}) = x_{t_b+t_p+2}(\beta),$
	$\vdots$
	$f_\beta(\alpha_n) = x_n(\beta).$
Otherwise output NO.	

Table 15: Algorithm to Check Whether  $Y$  Can be Linearly Pseudo-Expressed by  $\mathcal{B}$

We now finally present the polynomial time algorithm which takes the set of vectors  $\mathcal{Y}$  received by  $\mathbf{B}$  as input and finds in polynomial time, the pseudo-basis  $\mathcal{B} = \{Y_{j_1}, \dots, Y_{j_k}\} \subset \mathcal{Y}$ , pseudo-dimension  $k = |\mathcal{B}| \leq t_b + t_f$  and an index set  $\mathcal{I} = \{j_1, \dots, j_k\} \subset \{1, \dots, \gamma\}$  containing the indices of the vectors selected in  $\mathcal{B}$ . The algorithm uses algorithm Pseudo-Linear-Express as a black-box and is provided in Table 16.

<b>Algorithm Find-Pseudo-Basis</b> ( $\mathcal{Y}$ )	
1. Let $i = 1$ and $\mathcal{B} = \emptyset$ .	
2. While $i \leq \gamma$ and $ \mathcal{B}  < t_b + t_f$ , do:	
(a) By using Algorithm Pseudo-Linear-Express, check whether $Y_i$ can be linearly pseudo-expressed by $\mathcal{B}$ . If NO, then add $Y_i$ to $\mathcal{B}$ .	
(b) Set $i \leftarrow i + 1$ .	
3. Output $\mathcal{B}$ as a pseudo-basis, $k =  \mathcal{B} $ as the pseudo-dimension and index set $\mathcal{I}$ , containing the indices of the $k$ vectors selected in $\mathcal{B}$ .	

Table 16: Algorithm to Find the Pseudo-Basis of  $\mathcal{Y}$

**Theorem 21.** *Algorithm Find-Pseudo-Basis correctly finds the pseudo-basis and pseudo-dimension of  $\mathcal{Y}$ .*

**Note 1 (Convention For Using Algorithm Find-Pseudo-Basis).** *In the rest of the paper, we will use the notation  $(k, \mathcal{B}, \mathcal{I}) = \text{Find-Pseudo-basis}(\mathcal{Y})$  while invoking algorithm Find-Pseudo-Basis.*

### 10.3. Three Phase OPSMT Tolerating $\mathcal{A}_{(t_b, t_f, t_p)}^{\text{static}}$ / $\mathcal{A}_{(t_b, t_f, t_p)}^{\text{mobile}}$

Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n = 2t_b + t_f + t_p + 1$  wires, under the influence of  $\mathcal{A}_{(t_b, t_f, t_p)}^{\text{mobile}}$ . We now present a three phase OPSMT protocol called 3-OPSMT, which securely sends a message containing  $n^2$  field elements by communicating  $\mathcal{O}(n^3)$  field elements tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{\text{mobile}}$ . Since  $\mathcal{A}_{(t_b, t_f, t_p)}^{\text{static}}$  is a special type of  $\mathcal{A}_{(t_b, t_f, t_p)}^{\text{mobile}}$ , the protocol will also work against  $\mathcal{A}_{(t_b, t_f, t_p)}^{\text{static}}$ . In the protocol,  $\mathbf{S}$  establishes a random, information theoretically secure one time pad of size  $n^2$  with  $\mathbf{R}$ . Once the pad is established,  $\mathbf{S}$  uses the pad to blind the message and sends the blinded message reliably to  $\mathbf{R}$ . Let  $\mathcal{C}$  denote the set of all possible RS codewords of length  $n = 2t_b + t_f + t_p + 1$  encoded using polynomials of degree  $(t_b + t_p)$  over  $\mathbb{F}$ . We now present protocol 3-OPSMT. Recall that we are following Assumption 1 in protocol 3-OPSMT.

**Protocol 3-OPSMT**

**Phase I: S to R :**

**Computation by S:**

1. Let  $P = n^2 + t_b + t_f$ . **S** selects  $P$  random polynomials  $F_1(x), \dots, F_P(x)$  over  $\mathbb{F}$ , each of degree  $(t_b + t_p)$ .
2. For  $i = 1, \dots, P$ , **S** computes  $n$  length RS codeword corresponding to  $F_i(x)$ , denoted by  $C_i = [c_{i1} \dots c_{in}]$ , where  $c_{ij} = F_i(\alpha_j)$ , for  $j = 1, \dots, n$ .

**Communication by S:**

1. For  $j = 1, \dots, n$ , **S** sends  $j^{th}$  component of all the  $P$  codewords, namely  $c_{1j}, \dots, c_{Pj}$  over wire  $w_j$ .

**Phase II: R to S :**

**Computation by R:**

1. For  $i = 1, \dots, P$ , let **R** receive the  $n$  length vectors  $Y_i$ , where  $Y_i = C_i + E_i$  and  $E_i$  is the error vector introduced by Byzantine and fail-stop corrupted wires. **R** does not know  $C_i$  and  $E_i$ .
2. Let  $\mathcal{Y} = \{Y_1, \dots, Y_P\}$ . **R** executes  $(k, \mathcal{B}, \mathcal{I}) = \text{Find-Pseudo-Basis}(\mathcal{Y})$  to find pseudo-basis  $\mathcal{B} = \{Y_{j_1}, \dots, Y_{j_k}\} \subset \mathcal{Y}$ , pseudo-dimension  $k = |\mathcal{B}|$  and index set  $\mathcal{I} = \{j_1, \dots, j_k\} \subset \{1, \dots, P\}$ , where  $|\mathcal{I}| \leq (t_b + t_f)$ .

**Communication by R:**

1. **R** reliably sends the triplet  $(\mathcal{B}, k, \mathcal{I})$  to **S** by broadcasting it.

**Phase III: S to R :**

**Computation by S:**

1. **S** correctly receives the triplet  $(\mathcal{B}, k, \mathcal{I})$ .
2. **S** finds  $E_{j_1} = Y_{j_1} - C_{j_1}, \dots, E_{j_k} = Y_{j_k} - C_{j_k}$  and computes  $CORRUPTED = \cup_{i=1}^k \text{support}(E_{j_i})$ .
3. **S** concatenates all the  $F_i(0)$ 's such that  $i \in \{1, \dots, P\} \setminus \mathcal{I}$  and forms an information theoretic secure pad  $Z$  of length at least  $n^2$  (since  $|\mathcal{I}| \leq t_b + t_f$  and  $P = n^2 + t_b + t_f$ ).
4. Let  $Z_{n^2}$  denote the first  $n^2$  elements of  $Z$ . **S** computes  $\Gamma = Z_{n^2} \oplus m$ .

**Communication by S:**

1. **S** reliably sends (to **R**)  $\Gamma$  and list  $CORRUPTED$  containing the identity of Byzantine and fail-stop corrupted wires by broadcasting them.

**Local Computation by R at the End of Phase III :**

1. **R** correctly receives  $\Gamma$  and the list  $CORRUPTED$ .
2. **R** ignores all information received over the wires in  $CORRUPTED$  during **Phase I**.
3. **R** then reconstructs all the polynomial  $F_i(x)$  such that  $i \in \{1, \dots, P\} \setminus \mathcal{I}$ , by considering the correct values on  $F_i(x)$  received over remaining wires during **Phase I**.
4. Finally, **R** recovers pad  $Z$  (and hence  $Z_{n^2}$ ) by concatenating  $F_i(0)$ 's for all  $i \in \{1, \dots, P\} \setminus \mathcal{I}$  and hence the message  $m = \Gamma \oplus Z_{n^2}$ .

We now prove the properties of protocol 3-OPSMT.

**Lemma 23 (Correctness).** *In Protocol 3-OPSMT,  $\mathbf{R}$  will correctly recover  $m$ .*

PROOF: First notice that  $\mathbf{R}$  will correctly receive the blinded message  $\Gamma$  and list *CORRUPTED*, as they are broadcasted by  $\mathbf{S}$ . Now to prove that  $\mathbf{R}$  correctly recovers the message  $m$  sent by  $\mathbf{S}$ , we show that  $\mathbf{S}$  and  $\mathbf{R}$  shares the same pad  $Z$ .  $\mathbf{S}$  and  $\mathbf{R}$  will share  $Z$  if:

1.  $\mathcal{I}$  is same at both ends and
2.  $\mathbf{R}$  is able to correctly recover polynomials  $F_i(x)$  for  $i \in \{1, \dots, P\} \setminus \mathcal{I}$ .

Since  $\mathbf{R}$  sends the triplet  $(\mathcal{B}, k, \mathcal{I})$  to  $\mathbf{S}$  by broadcasting,  $\mathcal{I}$  will be same at both ends. Now we show that irrespective of the behavior of  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ ,  $\mathbf{R}$  will always recover all the polynomials  $F_i(x)$  for  $i \in \{1, \dots, P\} \setminus \mathcal{I}$ . Since  $\mathbf{S}$  correctly receives  $(\mathcal{B}, k, \mathcal{I})$ , *CORRUPTED* will contain all the wires which were corrupted in Byzantine and fail-stop fashion during **Phase I**.  $\mathbf{R}$  also correctly receives *CORRUPTED* from  $\mathbf{S}$ . Now ignoring the values received over the wires in *CORRUPTED* during **Phase I**,  $\mathbf{R}$  recovers all the polynomials with the remaining values. This is possible because each polynomial is of degree  $t_b + t_p$  and at least  $n - |\text{CORRUPTED}| \geq n - (t_b + t_f) = t_b + t_p + 1$  clean values on each polynomial, obtained over the wires in  $\{w_1, \dots, w_n\} \setminus \text{CORRUPTED}$  during **Phase I**, are available to  $\mathbf{R}$ , at the end of **Phase III**.  $\square$

**Lemma 24 (Secrecy).** *In Protocol 3-OPSMT,  $m$  will be information theoretically secure.*

PROOF: The message  $m$  will be information theoretically secure from  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  if the pad  $Z$  is information theoretically secure. According to the protocol,  $Z$  contains  $F_i(0)$  iff  $i \in \{1, \dots, P\} \setminus \mathcal{I}$ . Since  $(\mathcal{B}, k, \mathcal{I})$  was sent by  $\mathbf{R}$  by broadcasting, it may be eavesdropped by  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  during its transmission. But for remaining polynomials  $F_i(x)$ 's where  $i \in \{1, \dots, P\} \setminus \mathcal{I}$ ,  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  will know at most  $t_b + t_p$  points by eavesdropping  $t_b + t_p$  wires during **Phase I**. Since the degree of each of these polynomials is  $t_b + t_p$ ,  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  will lack one point on these polynomials to uniquely interpolate them and hence each  $F_i(0)$  with  $i \in \{1, \dots, P\} \setminus \mathcal{I}$  will be information theoretically secure.  $\square$

**Lemma 25 (Communication Complexity).** *Protocol 3-OPSMT sends a message  $m$  containing  $\ell = n^2$  field elements by communicating  $\mathcal{O}(n^3) = \mathcal{O}\left(\frac{n\ell}{n - (2t_b + t_f + t_p)}\right) = \mathcal{O}(n\ell)$  field elements, tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . Moreover, the protocol is an OPSMT protocol.*

PROOF: During **Phase I**,  $\mathbf{S}$  communicates  $P = n^2 + t_b + t_f$  codewords to  $\mathbf{R}$  which requires communication of  $Pn = (n^2 + t_b + t_f).n = \mathcal{O}(n^3)$  field elements. In **Phase II**,  $\mathbf{R}$  sends triplet  $(\mathcal{B}, q, \mathcal{I})$  through all the wires. This incurs a communication cost of  $\mathcal{O}((t_b + t_f).n.n) = \mathcal{O}(n^3)$ . The communication complexity of **Phase III** for sending *CORRUPTED* and  $\Gamma$  is  $\mathcal{O}(n^2.n) = \mathcal{O}(n^3)$ . Hence overall communication complexity of the protocol is  $\mathcal{O}(n^3)$ .

From Theorem 10, any three phase PSMT tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$  over  $n = 2t_b + t_f + t_p + 1$  wires, must communicate  $\Omega(n^3)$  field elements to securely send a message containing  $n^2$  field elements. Since the total communication complexity of protocol 3-OPSMT is  $\mathcal{O}(n^3)$ , the protocol is an OPSMT protocol tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ .  $\square$

**Theorem 22.** *Let  $\mathbf{S}$  and  $\mathbf{R}$  be connected by  $n = 2t_b + t_f + t_p + 1$  wires, under the influence of  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . Then there exists a three phase polynomial time OPSMT protocol, which securely sends a message containing  $\ell \geq n^2$  field elements by communicating  $\mathcal{O}(n\ell)$  field elements tolerating  $\mathcal{A}_{(t_b, t_f, t_p)}^{mobile}$ . Moreover, the same protocol will work against  $\mathcal{A}_{(t_b, t_f, t_p)}^{static}$ .*

PROOF: In order to send a message  $m$  containing  $\ell \geq n^2$  field elements,  $\mathbf{S}$  can divide  $m$  into several sub-blocks, each of size  $n^2$  and concurrently send them to  $\mathbf{R}$  by executing protocol 3-OPSMT for each sub-block. This will incur communication cost of  $\mathcal{O}(\frac{\ell}{n^2} \cdot n^3) = \mathcal{O}(n\ell)$ . The correctness and security of the resultant protocol follows from Lemma 23 and Lemma 24.  $\square$

## 11. Conclusion and Open Problems

In this paper, we initiated the study of PSMT tolerating static and mobile mixed adversary. We have given the complete characterization of single phase and multiphase PSMT protocols in undirected networks tolerating threshold static/ mobile mixed adversary. We have also proved the lower bound on the communication complexity of any single phase and multi phase PSMT protocol. Moreover, we have shown that our bounds are *asymptotically tight* by designing *communication optimal* protocols. Our investigation shows that it is appropriate to model different type of corruptions in the network by a mixed adversary, rather than considering every fault as Byzantine corruption.

Few questions remain unanswered in the paper which are as follows:

1. We have designed a three phase OPSMT protocol against static/mobile mixed adversary. It would be interesting to see if there exists a two phase OPSMT protocol against static/mobile mixed adversary.
2. Our proposed OPSMT protocols are communication optimal only for messages of some minimum size. It would be interesting to see if there exists OPSMT protocol for a given message of any size.

## References

- [1] S. Agarwal, R. Cramer, and R. de Haan. Asymptotically Optimal Two-Round Perfectly Secure Message Transmission. In C. Dwork, editor, *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, volume 4117 of *Lecture Notes in Computer Science*, pages 394–408. Springer-Verlag, 2006.

- [2] T. Araki. Almost Secure 1-Round Message Transmission Scheme with Polynomial-Time Message Decryption. In Reihaneh Safavi-Naini, editor, *Information Theoretic Security, Third International Conference, ICITS 2008, Calgary, Canada, August 10-13, 2008, Proceedings*, volume 5155 of *Lecture Notes in Computer Science*, pages 2–13. Springer, 2008.
- [3] D. Beaver. Efficient Multiparty Protocols Using Circuit Randomization. In J. Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 420–432, 1992.
- [4] Z. Beerliová-Trubíniová and M. Hirt. Efficient Multi-party Computation with Dispute Control. In S. Halevi and T. Rabin, editors, *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, volume 3876 of *Lecture Notes in Computer Science*, pages 305–328. Springer, 2006.
- [5] Z. Beerliová-Trubíniová and M. Hirt. Perfectly-Secure MPC with Linear Communication Complexity. In R. Canetti, editor, *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008, Proceedings*, volume 4948 of *Lecture Notes in Computer Science*, pages 213–230. Springer, 2008.
- [6] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, 2-4 May 1988, Chicago, Illinois, USA*, pages 1–10. ACM, 1988.
- [7] R. Canetti and T. Rabin. Fast Asynchronous Byzantine Agreement with Optimal Resilience. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, 1993*, pages 42–51. ACM, 1993.
- [8] D. Chaum, C. Crépeau, and I. Damgård. Multiparty Unconditionally Secure Protocols (extended abstract). In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, 2-4 May 1988, Chicago, Illinois, USA*, pages 11–19. ACM, 1988.
- [9] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults (Extended Abstract). In *26th Annual Symposium on Foundations of Computer Science, 21-23 October 1985, Portland, Oregon, USA*, pages 383–395. IEEE, 1985.
- [10] A. Choudhary, A. Patra, B. V. Ashwinkumar, K. Srinathan, and C. Pandu Rangan. Perfectly Reliable and Secure Communication Tolerating Static and Mobile Mixed Adversary. In R. Safavi-Naini, editor, *Information Theoretic Security, Third International Conference, ICITS 2008, Calgary,*

Canada, August 10-13, 2008, *Proceedings*, volume 5155 of *Lecture Notes in Computer Science*, pages 137–155. Springer, 2008.

- [11] A. Choudhary, A. Patra, Ashwinkumar B. V, K. Srinathan, and C. Pandu Rangan. On Minimal Connectivity Requirement for Secure Message Transmission in Asynchronous Networks. In V. Garg, R. Wattenhofer, and K. Kothapalli, editors, *Distributed Computing and Networking, 10th International Conference, ICDCN 2009, Hyderabad, India, January 03-06, 2009*, volume 5408 of *Lecture Notes in Computer Science*, pages 148–162, 2009.
- [12] Y. Desmedt and Y. Wang. Perfectly Secure Message Transmission Revisited. In E. Biham, editor, *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, volume 2656 of *Lecture Notes in Computer Science*, pages 502–517. Springer, 2003.
- [13] D. Dolev. The Byzantine Generals Strike Again. *Journal of Algorithms*, 3:14–30, 1982.
- [14] D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly Secure Message Transmission. *JACM*, 40(1):17–47, 1993.
- [15] M. J. Fischer, N. A. Lynch, and M. Paterson. Impossibility of Distributed Consensus with One Faulty Process. *JACM*, 32(2):374–382, 1985.
- [16] M. Fitzi, M. K. Franklin, J. A. Garay, and S. Harsha 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.
- [17] M. Fitzi, M. Hirt, and U. M. Maurer. Trading Correctness for Privacy in Unconditional Multi-Party Computation (Extended Abstract). In Hugo Krawczyk, editor, *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, Lecture Notes in Computer Science. Springer, 1998.
- [18] M. Franklin and R. Wright. Secure Communication in Minimal Connectivity Models. *Journal of Cryptology*, 13(1):9–30, 2000.
- [19] M. K. Franklin and M. Yung. Secure Hypergraphs: Privacy from Partial Broadcast. *SIAM J. Discrete Math.*, 18(3):437–450, 2004.
- [20] R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. The Round Complexity of Verifiable Secret Sharing and Secure Multicast. In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 580–589. ACM, 2001.

- [21] O. Goldreich, S. Micali, and A. Wigderson. How to Play Any Mental Game. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, USA*, pages 218–229. ACM, 1987.
- [22] W. C. Huffman and V. Pless. *Fundamentals of Error Correcting Codes*. Cambridge University Press, 2006.
- [23] M. V. N. A. Kumar, P. R. Goundan, K. Srinathan, and C. Pandu Rangan. On Perfectly Secure Communication Over Arbitrary Networks. In *PODC 2002, Proceedings of the Twenty-First Annual ACM Symposium on Principles of Distributed Computing, July 21-24, 2002 Monterey, California, USA*, pages 193–202. ACM, 2002.
- [24] K. Kurosawa and K. Suzuki. Almost Secure (1-Round, n-Channel) Message Transmission Scheme. Cryptology ePrint Archive, Report 2007/076, 2007.
- [25] 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.
- [26] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.
- [27] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error Correcting Codes*. North-Holland Publishing Company, 1978.
- [28] R. J. McEliece and D. V. Sarwate. On Sharing Secrets and Reed-Solomon Codes. *Communications of the ACM*, 24(9):583–584, 1981.
- [29] K. Menger. Zur Allgemeinen kurventheorie. *Fundamenta Mathematicae*, 10:96–115, 1927.
- [30] T. K. Moon. *Error Correction Coding: Mathematical Methods and Algorithms*. Wiley India, 2006.
- [31] C. Papadimitriou and U. Vazirani. Discrete Mathematics for Computer Science. Lecture Notes.
- [32] A. Patra, A. Choudhary, and C. Pandu Rangan. Constant Phase Efficient Protocols for Secure Message Transmission in Directed Networks. In Indranil Gupta and Roger Wattenhofer, editors, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Principles of Distributed Computing, PODC 2007, Portland, Oregon, USA, August 12-15, 2007*, pages 322–323. ACM, 2007.
- [33] A. Patra, A. Choudhary, and C. Pandu Rangan. Unconditionally Reliable and Secure Message Transmission in Directed Networks Revisited. In

- R. Ostrovsky, R. De Prisco, and I. Visconti, editors, *Security and Cryptography for Networks, 6th International Conference, SCN 2008, Amalfi, Italy, September 10-12, 2008. Proceedings*, volume 5229 of *Lecture Notes in Computer Science*, pages 309–326. Springer, 2008.
- [34] A. Patra, A. Choudhary, and C. Pandu Rangan. On communication complexity of secure message transmission in directed networks. In K. Kant, S. V. Premmaraju, K. M. Sivalingam, and J. Wu, editors, *Distributed Computing and Networking, 11th International Conference, ICDCN 2010, Kolkata, India, January 3 - 6, 2010, Proceedings*, volume 5935 of *Lecture Notes in Computer Science*, pages 42–53. Springer Verlag, 2010.
- [35] A. Patra, A. Choudhary, C. Pandu Rangan, K. Srinathan, and P. Raghavendra. Perfectly Reliable and Secure Message Transmission Tolerating Mobile Adversary. *International Journal of Applied Cryptography*, 1(3):200 – 224, 2009.
- [36] 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.
- [37] A. Patra, A. Choudhary, K. Srinathan, and C. Pandu Rangan. Perfectly Reliable and Secure Communication in Directed Networks Tolerating Mixed Adversary. In *Distributed Computing, 21st International Symposium, DISC 2007, Lemesos, Cyprus, September 24-26, 2007, Proceedings*, volume 4731 of *Lecture Notes in Computer Science*, pages 496–498. Springer, 2007.
- [38] A. Patra, A. Choudhary, K. Srinathan, and C. Pandu Rangan. Unconditionally reliable and secure message transmission in undirected synchronous networks: Possibility, feasibility and optimality. Accepted for publication in IJACT, 2009.
- [39] A. Patra, A. Choudhary, M. Vaidyanathan, and C. Pandu Rangan. Efficient Perfectly Reliable and Secure Message Transmission Tolerating Mobile Adversary. In Y. Mu, W. Susilo, and J. Seberry, editors, *Information Security and Privacy, 13th Australasian Conference, ACISP 2008, Wollongong, Australia, July 7-9, 2008, Proceedings*, volume 5107 of *Lecture Notes in Computer Science*, pages 170–186. Springer, 2008.
- [40] A. Patra, B. Shankar, A. Choudhary, K. Srinathan, and C. Pandu Rangan. Perfectly Secure Message Transmission in Directed Networks Tolerating Threshold and Non Threshold Adversary. In F. Bao, S. Ling, T. Okamoto, H. Wang, and C. Xing, editors, *Cryptology and Network Security, 6th International Conference, CANS 2007, Singapore, December 8-10, 2007, Proceedings*, volume 4856 of *Lecture Notes in Computer Science*, pages 80–101. Springer, 2007.

- [41] M. Pease, R. E. Shostak, and L. Lamport. Reaching Agreement in the Presence of Faults. *JACM*, 27(2):228–234, 1980.
- [42] T. Rabin and M. Ben-Or. Verifiable Secret Sharing and Multiparty Protocols with Honest Majority (Extended Abstract). In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA*, pages 73–85. ACM, 1989.
- [43] R. M. Roth. *Introduction to Coding Theory*. Cambridge University Press, 2004.
- [44] H. Sayeed and H. Abu-Amara. Perfectly Secure Message Transmission in Asynchronous Networks. In *Proceedings of 7th IEEE Symposium on Parallel and Distributed Processing*. IEEE, 1995.
- [45] H. Sayeed and H. Abu-Amara. Efficient Perfectly Secure Message Transmission in Synchronous Networks. *Information and Computation*, 126(1):53–61, 1996.
- [46] P.W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.
- [47] K. Srinathan. Secure distributed communication. PhD Thesis, IIT Madras, 2006.
- [48] K. Srinathan, A. Choudhary, A. Patra, and C. Pandu Rangan. Efficient Single Phase Unconditionally Secure Message Transmission with Optimum Communication Complexity. In R. A. Bazzi and B. Patt-Shamir, editors, *Proceedings of the Twenty-Seventh Annual ACM Symposium on Principles of Distributed Computing, PODC 2008, Toronto, Canada, August 18-21, 2008*, page 457. ACM, 2008.
- [49] K. Srinathan, M. V. N. Ashwin Kumar, and C. Pandu Rangan. Asynchronous Secure Communication Tolerating Mixed Adversaries. In Yuliang Zheng, editor, *Advances in Cryptology - ASIACRYPT 2002, 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002, Proceedings*, volume 2501 of *Lecture Notes in Computer Science*, pages 224–242. Springer, 2002.
- [50] K. Srinathan, A. Narayanan, and C. Pandu Rangan. Optimal Perfectly Secure Message Transmission. In M. K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 545–561. Springer, 2004.
- [51] K. Srinathan, A. Patra, A. Choudhary, and C. Pandu Rangan. Probabilistic Perfectly Reliable and Secure Message Transmission - Possibility, Feasibility

- and Optimality. In K. Srinathan, C. Pandu Rangan, and M. Yung, editors, *Progress in Cryptology - INDOCRYPT 2007, 8th International Conference on Cryptology in India, Chennai, India, December 9-13, 2007, Proceedings*, volume 4859 of *Lecture Notes in Computer Science*, pages 101–122. Springer, 2007.
- [52] K. Srinathan, A. Patra, A. Choudhary, and C. Pandu Rangan. Unconditionally Reliable Message Transmission in Directed Hypergraphs. In M. K. Franklin, L. C. K. Hui, and D. S. Wong, editors, *Cryptology and Network Security, 7th International Conference, CANS 2008, Hong-Kong, China, December 2-4, 2008. Proceedings*, volume 5339 of *Lecture Notes in Computer Science*, pages 285–303. Springer, 2008.
- [53] K. Srinathan, N. R. Prasad, and C. Pandu Rangan. On the Optimal Communication Complexity of Multiphase Protocols for Perfect Communication. In *Proceedings of 2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA*, pages 311–320. IEEE Computer Society, 2007.
- [54] K. Srinathan, P. Raghavendra, and C. Pandu Rangan. On Proactive Perfectly Secure Message Transmission. In J. Pieprzyk, H. Ghodosi, and E. Dawson, editors, *Information Security and Privacy, 12th Australasian Conference, ACISP 2007, Townsville, Australia, July 2-4, 2007, Proceedings*, volume 4586 of *Lecture Notes in Computer Science*, pages 461–473. Springer, 2007.
- [55] Y. Wang and Y. Desmedt. Secure Communication in Multicast Channels: The Answer to Franklin and Wright’s Question. *J. Cryptology*, 14(2):121–135, 2001.
- [56] A. C. Yao. Protocols for Secure Computations. In *Proceedings of 23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, 3-5 November 1982*, pages 160–164. IEEE Computer Society, 1982.