Perfectly Reliable Message Transmission on Undirected Graphs

A Project Report

submitted in partial fulfillment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

Ashwinkumar B V

under the guidance of

PROF. C. PANDU RANGAN

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY MADRAS

May 2008

Certificate

This is to certify that this project report titled **Perfectly Reliable Message Transmission on Undirected Graphs** submitted by **Ashwinkumar B V** in partial fulfillment for the award of the degree of **Bachelor of Technology** in Computer Science and Engineering, is a bona-fide record of work carried out by him under my guidance and supervision in the Department of Computer Science and Engineering, Indian Institute of Technology, Madras.

Place : Chennai

Prof. C. Pandu Rangan

Acknowledgments

I must thank Prof. C. Pandu Rangan for being such a wonderful guide and showing me the world of Theoretical Computer Science. I still remember how his first course in Discrete Mathematics drew me. I also want to thank Prof Ravi Sundaram and Prof Raj for allowing me to come to North Eastern University where I had some of my most memorable experiences. This is the first place where I got a true flavor of research.

I want to thank the head of the department Prof. Timothy A Gonsalves for letting me use the department facilities. I would like to express my gratitude to Prof. Aravind, Shankar and Kama under whom I took some of my most cherished courses. CS Department has been one of the best places with all the professors begin very friendly and flexile. It has also been a place where I learnt the most, not only in academics but also in practical issues.

It wouldn't have been possible to maintain the sanity hadn't it been for some of the most insane moments I had with my very close friends muthu, vinod , sk, rakshit, kc, atulya, gayab, julie and kamu. I will always remember the crazy chat I and atulya used to have when going for coffee at gurunad. Jamuna has been a second home to me and coming to think of leaving it makes me nostalgic.

I want to thank balu, kurma and naresh for being very supportive and friendly in our combined goal of research. I would never forget their help when I was sick in Boston. I also want to thank george, sameer, bundy, setia and other CS junta for being very friendly and fun going. George has been one of the nicest guys and the one person in our batch I admire the most. It will be unfair to not mention sameer as he was one guy who worked the most in our batch but was at the receiving end most of the time.

I should also thank Ashish and Arpita for introducing me to some very nice problems one of which turned into this thesis. It was fun and I had a great time doing research with them. The long evening discussions and fun talks will always remain in my memory. I should thank them once again for offering to write my first paper(and other papers :)).

I want to thank my seniors Ravishankar, Aravindan, Varad, Karthekeyan, Harsha, Kaushik and Muthu for their guidance in my application for higher studies and their advice. I want to thank all the members of the TCS lab Harini, Sharmila, Vivek, Masilamani, Chandrashekar, Ambika and others for making the lab conducive for research and fun place to stay.

Last but not the least, none of this would have been possible if it weren't for the sacrifices of my mother, father and grand parents. Thank you all.

To my Grand Father

Abstract

We consider for the first time the problem of trade off between the Network Connectivity, Phase Complexity and Communication Complexity of Reliable Communication Tolerating Mixed Adversary [1]. The problem of reliable communication of a message m by a sender **S** to a receiver **R** in an unreliable network involves transporting m from **S** to **R** in a reliable manner. The unreliability of the network is modeled via an adversary controlling the nodes of the network according to different fault models. This problem traditionally known as *perfectly reliable message transmission*(PRMT) in the literature is used as a black box in most of the *multi party computation*(MPC) and Byzantine protocols. This problem was traditionally studied in the presence of Byzantine faults in the network. We study the problem in a more generalized fault model. Specifically we study the problem in the standard model of abstracting the network as a graph and categorizing the nodes based on their fault models. We prove a lower bound on the Phase Complexity given the Network Connectivity and Communication Complexity and provide a corresponding protocol with optimal Phase Complexity(up to constant factor). This solves an open problem posed in [2]. To the best of my knowledge, this is the first time such a treatment has been done. This thesis is based on a joint work with Arpita Patra, Ashish Choudhary, Kannan Srinathan and C. Pandu Rangan.

Table of Contents

List of Symbols

- $\mathcal{N}$ - Network
- P - Set of nodes in the network
- E - Set of links in the network
- **S** - Sender
- **R** - Receiver
- m - message
- n - connectivity of the graph
- $\mathcal{A}_{(t_b,t_f)}$ - Adversary
- t_b - number of Byzantine nodes
- t_f - number of fail-stop nodes
- l - length of message to be sent
- b - number of field elements communicated
- c - constant
- $\prod$,$\prod'$ - protocols
- $\mathbb{F}$ - Finite field
- X - $n - t_f$
- d - constant
- a - $2 \times d$
- N - similar to n
- T - similar to t_f
- MPC - Multi Party Computation
- PRMT - Perfectly Reliable Communication

CHAPTER 1

Introduction

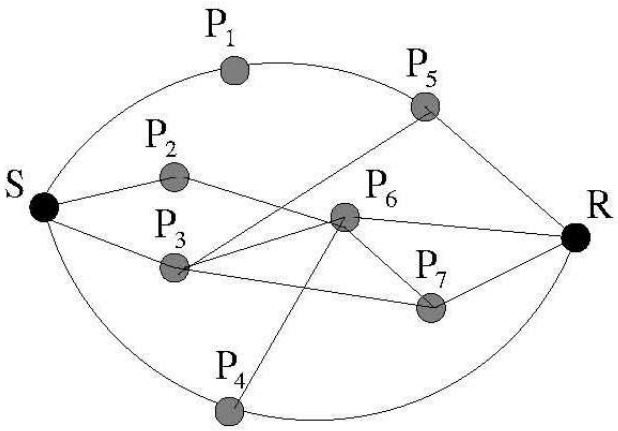
Intuitively one feels that adding redundancy to the system increases its reliability. This is formally used in most of the distributed algorithms to achieve security and reliability. We can even see this in real life examples like (a) Storage of data in multiple computers to deal with system crashes, (b) Critical mails are usually sent through multiple means such as e-mail/post/courier. But redundancy is accompanied by extra work which results in an inherent inefficiency. In this thesis, we explore the limits of this inefficiency inherent in a certain model of communication where reliability is achieved by the phenomenon of distribution and interaction.

We see that in a typical scenario the Source and Receiver which need to communicate are part of a network. The faults in the network are characterized as some limit on the faulty nodes. The requirement for the protocol is to simulate a reliable channel between Source and Receiver. This problem can be handled by coding theory which corrects the errors by adding redundancy. We see that when we combine coding theory with the power of interaction it decreases the necessity for redundancy. But this comes with cost of larger number of phases for interaction.

In keeping with the tradition of describing Distributed Algorithms we state the problem informally in terms of a battlefield scenario. There is army which is returning after a long battle. Through several routes for their return journey among which all but one are under the control of their enemy. The enemy is strong and it will kill any soldiers passing through the route it controls. As the army doesn't know the safe route one would think the best strategy is to go in equal number in each route. Only $\frac{1}{num\_routes}$ of the army will reach back. But we in this thesis give an algorithm where most of the army(albeit a constant fraction of them) will reach their kingdom and also prove that no other algorithm can do better than ours.

1.1 Models and Settings

1.1.1 Network



Consider a synchronous undirected network denoted by $\mathcal{N} = (\mathcal{P}, \mathcal{E})$ where $\mathcal{P}$ is the set of nodes and $\mathcal{E} \subset \mathcal{P} \times \mathcal{P}$ is the set of links. In the *perfectly reliable message transmission* (PRMT) problem over $\mathcal{N}$, a sender $\mathbf{S} \in \mathcal{P}$, wishes to send a message m, which is a sequence of ℓ field elements from a finite field $\mathbb{F}$, to a receiver $\mathbf{R} \in \mathcal{P}$, in such a manner that $\mathbf{R}$ recovers the message without any fault (*perfectly*), in spite of the presence of an adversary $\mathcal{A}_{(t_b,t_f)}$(Defined in section 1.1.2). The reason for modeling network unreliability as an adversary is done so that worst case lower and upper bounds can be proved. All the n nodes in the network are modeled as probabilistic interactive Turing Machines. In this work, we study in the synchronous settings. The underlying network $\mathcal{N}$ is said to be synchronous when all the players share a common global clock. All messages sent are sent on a clock 'tick', and are received on the next 'tick'.

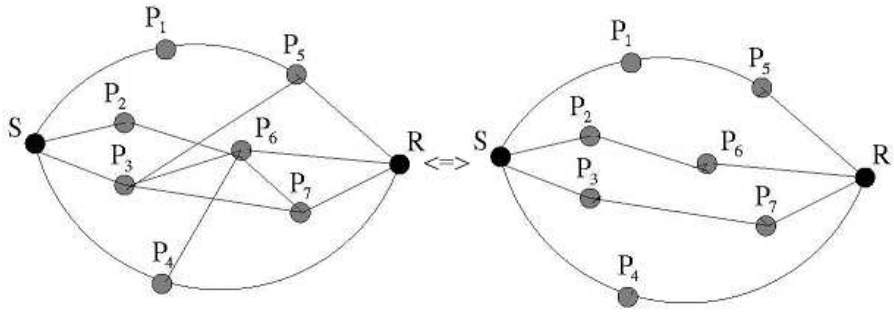
1.1.2 Adversary

The adversary in this work is denoted by $\mathcal{A}_{(t_b,t_f)}$. $\mathcal{A}_{(t_b,t_f)}$ possesses *unbounded* computing power and can corrupt disjoint sets of at most t_b and t_f nodes in Byzantine and Fail-stop fashion respectively, in a centralized manner. Moreover, the adversary is adaptive; i.e, the set of nodes which are going to be under the control of $\mathcal{A}_{(t_b,t_f)}$ is decided *dynamically* depending

upon the data seen by the adversary so far during the protocol execution. Once a node is under the control of the adversary, it is going to be so for the rest of the protocol. If a node P is fail-stop corrupted by $\mathcal{A}_{(t_b,t_f)}$, then the adversary can force P to *crash* at will at any time during the execution of the protocol but can not access its internal data and can not force its behavior to deviate from the protocol. So as long as P is alive, it honestly follows the protocol. Also once P is crashed, it never becomes alive again. If a node P is Byzantine corrupted by $\mathcal{A}_{(t_b,t_f)}$, then the adversary has full access to the internal data of P and can force P to deviate from the protocol arbitrarily. Hence $\mathcal{A}_{(t_b,t_f)}$ can also listen on these nodes. We assume that **S** and **R** do not share any information, what so ever, before the beginning of the protocol. Also, the protocol specification is public and known to everybody, including $\mathcal{A}_{(t_b,t_f)}$. However, $\mathcal{A}_{(t_b,t_f)}$ will not know the random coin tosses used by **S** and **R** in the protocol. Also as $\mathcal{A}_{(t_b,t_f)}$ has *unbounded computing* power, we cannot solve the PRMT problem using public key cryptography, digital signatures etc, as they are all based on the assumption that adversary has *polynomial time* computing power. The adversary we study is known as threshold mixed adversary in the literature. But we note that the lower bound proved in Chapter 4 holds even against polynomial adversaries and it should be fairly easy to model it for random errors.

1.1.3 Abstraction

Like the most of the literature we follow the model used by Dolev et.al [3]. We see that if some intermediate node between **S** and **R** is under the control of the adversary. Hence all the paths between **S** to **R** passing through that node can be modeled by a single wire between **S** and **R**. By the above argument the network can be abstracted as vertex disjoint paths(also known as wires) from **S** to **R**. This follows from by Menger's theorem [4] which states that a graph is $c-(\mathbf{S},\mathbf{R})$-connected iff **S** and **R** are connected by at least c vertex disjoint paths.



The adversary which controls t_b nodes in Byzantine fashion and t_f nodes in fail-stop fashion now controls t_b wires in Byzantine fashion and t_f wires in fail-stop fashion. This is possible as the adversary can position himself at some of the nodes in the min-cut of the graph.

1.2 Motivation and Significance

The PRMT problem is well-motivated for it being one of the fundamental primitives used by all fault-tolerant distributed algorithms like Byzantine agreement, multi-party computation etc (see [3, 5, 6, 7, 8, 9, 10] and their references). All these popular fault-tolerant distributed algorithms assume that the underlying network is a complete graph, thereby implicitly assuming the existence of a PRMT protocol that can simulate a complete graph over the actual network which is seldom a complete graph itself. Chapter 7 of [11] solves the problem of Byzantine agreement for general graphs. In literature all these problems have been solved in cryptographic model also. The PRMT problem was first proposed and solutions is a different model was given by [3].

Here we will argue about the model of Network and Adversary. The reason for considering an adversary who can corrupt in both Byzantine and fail-stop is of theoretical interest as it is a more generalized adversary. Modeling every node as Byzantine may be an overkill when the adversary may not be powerful enough to corrupt every node in Byzantine fashion. This is one step closer towards a very generalized adversary. Also when we look at the solutions of Distributed Algorithms in cryptographic model the Byzantine nodes will behave like Fail-stop nodes. But keys over some of the nodes may be compromised which behave like Byzantine in the absence of authentication. Even this problem is nicely captured in our model. Hence if we try to design networks with very high resiliency and lower connectivity in different models

our bound comes into the picture.

1.3 Our Contributions

The efficiency of any PRMT protocol may be expressed by three parameters, namely,

1. Connectivity (n) of the network
2. Communication complexity (b), which is the total number of field elements communicated by **S** and **R**
3. Phase Complexity (r).

Let us first consider the scenario where we work with minimal connectivity; viz $n = 2t_b + t_f + 1$. From the basic results of coding theory (as given in chapter 3), any *single phase* PRMT protocol has to communicate $\Omega(n\ell)$ field elements to reliably send m, containing ℓ field elements. Hence it is clear that a PRMT protocol with communication complexity of less than $n\ell$, *must* run for several phases. Hence a natural and fundamental question here is the following:

> *Given $n = 2t_b + t_f + 1$ and $b < n\ell$, what is the minimum value of r? Do we have such an $O(r)$ phase efficient PRMT protocol?*

It is clear that for any PRMT protocol, $\ell \leq b \leq n\ell$. We may refer a protocol with communication complexity of $b = c\ell$, where c is a constant independent of n, as *communication optimal* protocol. Extending the above question to this very interesting and important case, we may ask the following:

> *When $n = 2t_b + t_f + 1$ and $b = O(\ell)$, what is the minimum value of r? Do we have such an $O(r)$ phase efficient PRMT protocol?*

Note that if such a protocol exists, it will be simultaneously optimal in connectivity, communication complexity and phase complexity. So far, we have considered only minimally connected network but if we have higher connectivity, then again the required number of phases may be reduced. Specifically, when $n \geq 2t_b + t_f + 1$ and $b < n\ell$, we ask for minimum r and

a corresponding phase optimal protocol. Unifying all the above questions, we formulate the following most generic question, which is the *holy grail for PRMT problem*:

> *Given an n-connected network ($n \geq 2t_b + t_f + 1$) and a value b, where $\ell \leq b < n\ell$, what is the minimum number of phases r needed to reliably send m, where $|m| = \ell$, within $O(b)$ communication complexity?*

In this report, we completely resolve the above question by deriving exact expression for lower bound for the phase complexity of PRMT protocols and also design a PRMT protocol whose total phase complexity matches this bound, thus proving that our bound is *asymptotically* tight. From our general result, we obtain several interesting results on the inherent trade-off available between the three parameters, namely n, b and r. We also derive several surprising corollaries for specific instances. For example our general result imply that when $t_b = 0, n = t_f + 1$ and $b = O(\ell)$, then any PRMT protocol requires $\Omega(\log(t_f))$ phases to send m. The PRMT protocols existing in the literature usually attempts to optimize one of the parameters and rarely two of the parameters mentioned above [8, 12]. However, our protocol is simultaneously optimal in all the three parameters.

Surprise Factor: As mentioned above, if $t_b = 0, n = t_f + 1$ and $b = O(\ell)$, then any PRMT protocol requires $\Omega(\log(t_f))$ phases to send m containing ℓ field elements. This is surprising because from Theorem 2.1.3, against Byzantine adversary, there exists a three (constant) phase PRMT protocol, which achieves reliability with *constant factor* overhead. These two results seem to be counter intuitive, since Byzantine adversary is more powerful than fail-stop adversary (Byzantine adversary can maliciously change the information over the wires, where as fail-stop adversary can only block the communication over a wire). Here we informally justify that it is not so (a formal argument is given by the proof of Theorem 4.1.1). In a minimally connected network tolerating only Byzantine adversary, we have $n = 2t_b + 1$ wires. So the number of corrupted wires is less than half of the total number of wires. This allow us to do error detection/correction in three phases, resulting in a three phase PRMT protocol with a communication complexity of $O(\ell)$ (see the PRMT protocol of [8]). However, in a minimally connected network tolerating only fail-stop adversary, we have only $n = t_f + 1$ wires between **S** and **R**, of which there is only *one* un-corrupted wire. It is this reduced con-

nectivity (in comparison to the case of Byzantine adversary) that does not allow us to design a constant phase PRMT protocol against fail-stop adversary in a minimally connected network, with communication complexity of $O(\ell)$.

Remark 1 *Note that sending m reliably in a single phase, even by broadcasting it along all the n wires, has communication complexity of $b = n\ell$. Hence, we are interested only in the case where $\ell \leq b < n\ell$. Also if $t_f = 0$ and $n \geq 2t_b + 1$, then m can be sent reliably in three phases by communicating $O(\ell)$ field elements [8] and hence is optimal in all the three parameters. However, the lower bound and protocol given in this report, considers a mixed adversary, where $t_b, t_f \geq 0$.*

Remark 2 *Throughout the report, we use X to denote $n - t_f$; i.e., $X = (n - t_f)$. Hence $n = X + t_f$.*

Remark 3 *In our protocols, for simplicity, we assume that Byzantine adversary does not block the wires under its control. Our protocols can be easily adapted to tolerate such a behavior by the Byzantine adversary, without affecting the phase and communication complexity.*

1.4 Organization of thesis

In Chapter 2 we mention some of the existing results and preliminary results which will be used in the thesis. Following this we prove certain bounds on single phase protocols in Chapter 3. After this in Chapter 4 we prove lower bounds on the phase complexity which is one of the main contributions of this thesis. Finally we end the work by giving a protocol which runs in optimal number of phases in Chapter 5.

CHAPTER 2

Coding Theory Preliminaries and Existing Results

In this section we will review the existing results and the basic coding theory preliminaries necessary for proving the lower bound and the protocol construction.

2.1 Existing Results

PRMT Tolerating Byzantine Adversary: PRMT problem was first introduced and solved by Dolev et.al [3] under the presence of a t_b-active Byzantine adversary $\mathcal{A}_{t_b}$, who can corrupt at most t_b nodes in the network in Byzantine fashion. Dolev et.al abstracted the network in the form of vertex disjoint paths (also known as *wires*) between **S** and **R** The reason for such an abstraction is as follows: suppose some intermediate node between **S** and **R** is under the control of the adversary. Then all the paths between **S** and **R**, irrespective of their length, passing through that node are also compromised. Hence, all the paths between **S** to **R** passing through that node can be modeled by a single wire between **S** and **R**. The characterization of PRMT given by Dolev et.al is as follows:

Theorem 2.1.1 ([3]) *PRMT over an undirected network $\mathcal{N}$ tolerating $\mathcal{A}_{t_b}$ is possible iff $\mathcal{N}$ is $(2t_b + 1)$-(**S, R**)-connected.*

In [7], Srinathan et.al have given the lower bound on the communication complexity of any *single* phase PRMT protocol tolerating $\mathcal{A}_{t_b}$. The lower bound is as follows:

Theorem 2.1.2 ([7]) *Any single phase PRMT protocol over $n \geq 2t_b + 1$ wires communicates $\Omega\left(\frac{n\ell}{n-2t_b}\right)$ field elements to reliably send a message containing ℓ field elements against $\mathcal{A}_{t_b}$.*

Interestingly, in [8], Arpita et.al have shown that if more than one phase is allowed, then the communication complexity of PRMT protocols can be reduced significantly. The result of Arpita et.al is as follows:

Theorem 2.1.3 ([8]) *Let $\mathcal{N}$ be an undirected network, under the influence of $\mathcal{A}_{t_b}$ such that* **S** *and* **R** *are connected by $n = 2t_b + 1$ wires. Then three phases are sufficient for the existence of any PRMT protocol which sends a message containing ℓ field elements by incurring a communication complexity of $O(\ell)$ field elements.*

The above theorem says that in a minimal connected network against Byzantine adversary, three phases are sufficient for any PRMT protocol to achieve reliability with *constant factor* overhead.

PRMT Tolerating Mixed Adversary: Studying mixed adversary in the context of PRMT is well motivated. 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 a strongly protected node, while he may affect in a Byzantine fashion a weakly protected node. Thus, we may capture the abilities of an adversary in a more realistic manner using two parameters t_b and t_f, where t_b and t_f are the number of nodes under the influence of adversary in Byzantine and fail-stop 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. Doing so will be an "overkill". Recently, Arpita et.al [13] have given the characterization for PRMT over undirected networks tolerating a mixed adversary $\mathcal{A}_{(t_b,t_f)}$.

Theorem 2.1.4 ([13]) *PRMT over an undirected network $\mathcal{N}$ tolerating $\mathcal{A}_{(t_b,t_f)}$ is possible iff $\mathcal{N}$ is $(2t_b + t_f + 1)$-(***S, R***)-connected.*

PROOF: The necessity of the above condition follows from the following argument. Let Π be a PRMT protocol over $\mathcal{N}$, where there exists $n = 2t_b + t_f$ wires between **S** and **R**. Now consider the following adversarial strategy: the adversary blocks the communication over t_f wires. Now consider the network $\mathcal{N}'$ that is induced by $\mathcal{N}$ on deleting these t_f paths from $\mathcal{N}$ (this can be interpreted as an adversary blocking the communication over t_f paths). It follows that $\mathcal{N}'$ is not a $(2t_b + 1)$-(**S,R**)-connected network. Evidently, if Π is a PRMT protocol on $\mathcal{N}$, then Π' is a PRMT protocol on $\mathcal{N}'$, where Π' is the protocol Π restricted to the players in $\mathcal{N}'$. However, from Theorem 2.1.1, we know that Π' is non-existent. Thus Π is impossible too. The sufficiency of the above condition is shown by the following protocol: Let **S** and **R**

are connected by $n \geq 2t_b + t_f + 1$ wires. **S** sends the message m through all the n wires. **R** recovers the message by taking majority among the received values. □

Now we demonstrate that Theorem 2.1.4 shows more fault tolerance in comparison to Theorem 2.1.1. Let $\mathcal{N}$ be a network where **S** and **R** are connected by $n = 4$ wires. Then from Theorem 2.1.1, the maximum number of Byzantine faults that can be tolerable is one. However, from Theorem 2.1.4, it is possible to tolerate one fail-stop fault, in addition to one Byzantine fault.

2.2 Coding Theory Preliminaries

Let $Ch_{(t_b,t_f)}$ denotes a noisy channel, where at most t_f and t_b locations can be arbitrarily erased and changed respectively during the transmission of a codeword. A *block* error-erasure correcting code encoding a *message* of k field elements to a *codeword* of n field elements is an injective mapping $\mathcal{C} : \mathbb{F}^k \rightarrow \mathbb{F}^n (n > k)$, where $\mathbb{F}$ is the underlying field. The encoding function is used in conjunction with a decoding function $\mathcal{D} : \mathbb{F}^n \rightarrow \mathbb{F}^k$ with the property that if its input differs from a valid codeword in at most t_b locations, apart from at most t_f erasures, then $\mathcal{D}$ outputs the message corresponding to that codeword. We say that the code corrects t_b Byzantine errors and t_f erasures. Clearly, such a decoding function will always exist if any two valid codewords differ in at least $2t_b + t_f + 1$ locations.

The maximum attainable efficiency of any *block* error-erasure correcting code is subject to the *Singleton Bound*, given by the following lemma:

Lemma 2.2.1 (Singleton bound [14]) *Let $\mathcal{C}$ be an error-erasure correcting code which reliably transmits k field elements by communicating a total of n field elements and has a* distance *of d. Then $n \geq k + d - 1$.*

For a (t_b, t_f) error-erasure correcting code, the distance d (which is the minimum Hamming distance between any two codewords) is at least $2t_b + t_f + 1$. Thus we have the following corollary:

Corollary 2.2.2 *Let $\mathcal{C}$ be a (t_b, t_f) block error-erasure correcting code. Then $k \leq n - (2t_b + t_f)$.*

We now give the definition of a special kind of *block* error-erasure correcting code called Reed-Solomon code, which we use in our protocols.

Definition 2.2.3 ([14]) *Let $\mathbb{F}$ be a finite field and $\alpha_1, \alpha_2, \ldots \alpha_n$ be distinct elements of $\mathbb{F}$. Given $k < n \leq |\mathbb{F}|$, and an arbitrary block $\mathbf{B} = [m_1\ m_2\ \ldots\ m_k]$, the encoding function for the Reed-Solomon code is defined as $[p_{\mathbf{B}}(\alpha_1)\ p_{\mathbf{B}}(\alpha_2)\ \ldots\ p_{\mathbf{B}}(\alpha_n)]$ where $p_{\mathbf{B}}(x)$ is the polynomial $\sum_{i=0}^{k-1} m_{i+1}x^i$.*

We denote RS code by $RS(n, k)$, which encodes a message block of size k into a codeword of size n.

Theorem 2.2.4 ([14]) *The Reed-Solomon code meets the singleton Bound.*

From Corollary 2.2.2 and Theorem 2.2.4, we get the following corollary.

Corollary 2.2.5 *Let $\mathcal{C}$ be a (t_b, t_f) block error-erasure correcting RS code. Then $k \leq n - (2t_b + t_f)$.*

Theorem 2.2.6 gives the number of errors which can be corrected and detected by RS codes.

Theorem 2.2.6 ([14]) *Let C denote the $RS(n, k)$ codeword for a message block of size k. Suppose the codeword is sent over the channel $Ch_{(t_b,t_f)}$. Let n' denotes the size of the received codeword C', where $n' \geq n - t_f$. Then RS decoding can correct up to c Byzantine errors in C' and simultaneously detect additional d Byzantine errors in C' iff $n' - k \geq 2c + d$.*

CHAPTER 3

Bounds for Single Phase PRMT

3.1 Single phase

We now prove the lower bound on the communication complexity of any single phase PRMT protocol tolerating $\mathcal{A}_{(t_b,t_f)}$. The bound is obtained by showing the "equivalence" between single phase PRMT protocol and error-erasure correcting code. More specifically, we show that the maximum attainable efficiency of any single phase PRMT against $\mathcal{A}_{(t_b,t_f)}$ is bounded by the maximum attainable efficiency of a (t_b, t_f) error-erasure correcting code, given by the *Singleton Bound*.

We now describe a single phase PRMT protocol **PRU-SP**$(m, \ell, n, t_b, t_f, k)$ against $\mathcal{A}_{(t_b,t_f)}$, obtained by using the corresponding $RS(n, k)$ code. In the protocol **S** and **R** are connected by $n \geq 2t_b + t_f + 1$ wires, $w_i, 1 \leq i \leq n$, under the influence of $\mathcal{A}_{(t_b,t_f)}$. **S** wants to send m where $|m| = \ell$.

Protocol PRU-SP$(m, \ell, n, t_b, t_f, k)$: Single Phase PRMT

- **S** breaks m into blocks $\mathbf{B_1}, \mathbf{B_2}, \ldots, \mathbf{B_{\ell/k}}$, each consisting of k field elements, where $k = (X-2t_b), X = n - t_f$. If ℓ is not an exact multiple of k, a default padding is used to make ℓ mod $k = 0$.
- For each $\mathbf{B_j}$, **S** computes the RS(n, k) codeword of $\mathbf{B_j}, 1 \leq j \leq \ell/k$ denoted by $(c_{j1}\ c_{j2}\ \ldots\ c_{jn})$ in parallel. **S** sends c_{ji} along wire $w_i, 1 \leq i \leq n$.
- **R** receives the (possibly corrupted/erased) c_{ji}'s for all the blocks of m and applies the RS decoding algorithm to each of them and constructs all blocks $\mathbf{B_j}$ in parallel. **R** concatenates $\mathbf{B_j}$'s to recover the message m.

Lemma 3.1.1 *Protocol* **PRU-SP** *correctly sends* m *by communicating* $O\left(\frac{n\ell}{(X-2t_b)}\right)$ *field elements.*

PROOF: Follows from Corollary 2.2.5 and working of the protocol. □

The reverse process is equally valid - given a single phase PRMT protocol against $\mathcal{A}_{(t_b,t_f)}$, we can convert it into an (t_b, t_f) block error-erasure correcting code, whose efficiency is

bounded by Singleton Bound (Corollary 2.2.2). Thus, the maximum attainable efficiency for any single phase PRMT protocol is also subject to the Singleton Bound. Thus we have the following theorem.

Theorem 3.1.2 *Let* **S** *and* **R** *be connected by* $n \geq 2t_b + t_f + 1$ *wires, under the influence of* $\mathcal{A}_{(t_b,t_f)}$*. Then any single phase PRMT protocol communicates* $\Omega\left(\frac{n\ell}{X-2t_b}\right)$ *field elements to reliably send* m*, where* $|m| = \ell$ *and* $X = n - t_f$*. Moreover, the bound is tight.*

PROOF: Follows from the above discussion. □

Remark 4 *The conversion from single phase PRMT protocol to an error-erasure correcting code is straightforward if the messages sent along each wire in the protocol are of same length. Suppose however, there exists a protocol* Π *that does not have this symmetry property and beats the Singleton bound. Then consider the protocol* Π' *which consists of* n *sequential executions of protocol* Π *with the identities of the wires being "rotated" by a distance of* i *in the* i^{th} *execution. Clearly this protocol achieves the symmetry property by "spreading the load"; further its message expansion factor is equal to that of* Π*. It therefore beats the Singleton bounds as well, which is a contradiction. Thus without any loss of generality, we assume that the messages sent along each wire is of same length.*

Protocol **PRU-SP** has another important property given in the following theorem.

Theorem 3.1.3 *If* **R** *in advance knows the exact identity of* $\alpha \leq t_b$ *wires which are Byzantine corrupted, then the protocol* **PRU-SP***(*m, ℓ, n, t_b, t_f, k*) will reliably transmit* m *in a single phase, using block size* $k = (X - 2t_b) + \alpha$*, by communicating* $O\left(\frac{n\ell}{(X-2t_b)+\alpha}\right)$ *field elements.*

PROOF: Since **R** knows α wires which are Byzantine corrupted, it simply ignores these wires and therefore the connectivity/set of active wires reduces to $n-\alpha$. Among the remaining wires, at most $t_b - \alpha$ could be Byzantine corrupted. Substituting these values in Corollary 2.2.5, we get $k \leq n - \alpha - 2(t_b - \alpha) - t_f = (X - 2t_b) + \alpha$ where $X = n - t_f$. Hence **PRU-SP** will work correctly with $k = (X - 2t_b) + \alpha$. □

Before we end our discussion on single phase PRMT, we present a two phase PRMT protocol **SP-REL** (based on RS code which is a specific instance of error-erasure correcting code), that possesses both error correction and error detection capabilities of the underlying error-erasure correcting code. Such a protocol is used later in designing phase optimal PSMT

protocol. **SP-REL** is based on the following principle: **S** and **R** guess that adversary will fail-stop at most $t_f - k_f$ wires and corrupt at most $\frac{t_b}{2}$ wires. If it indeed happens, then **S** can reliably send $\frac{t_b}{2} + k_f$ extra field elements (in addition to what is permitted by Singleton Bound) in a single phase to **R**. However if adversary either fail-stop $t_f - k_f + 1$ (or more) wires or corrupts more than $\frac{t_b}{2}$ wires, then **R** will not be able to recover anything. However, **R** either comes to know the identity of at least $t_f - k_f + 1$ fail-stop wires or *detects more than* $\frac{t_b}{2}$ *Byzantine faults*. In the later case, **R** can broadcast back the received information to **S**, who after local verification can identify more than $\frac{t_b}{2}$ Byzantine corrupted wires.

Protocol SP-REL (m, n, t_b, t_f, k_f)

1. **S** performs the same computation and communication as done in protocol **PRU-SP**, except that now **S** divides m into blocks $B_1, B_2, \ldots, B_p$, each consisting of k field elements, where $k = (X - 2t_b) + \frac{t_b}{2} + k_f$.
2. Let **R** receives information over a wires, of which at most t_b could be corrupted. Thus **R** receives a values, corresponding to each of the p codewords.
3. IF $a < n - t_f + k_f$ then **R** broadcasts to **S**, "ERROR1" signal along with the count of the number of wires and their identity over which **R** has not received any information (which is $n - a$) and terminates the protocol. /* In this case, more than $t_f - k_f$ fail-stop errors has occurred.*/
4. IF $a \geq n - t_f + k_f$ then **R** applies RS decoding algorithm to each of the p received codewords and correct $\frac{t_b}{2}$ errors and simultaneously detect additional $\frac{t_b}{2}$ faults (if it had occurred) in each of the p codewords in parallel.
5. If after correcting $\frac{t_b}{2}$ errors, the decoding algorithm does not detect additional faults in any of the p received codewords, then **R** correctly recovers $\mathbf{B_j}$, $1 \leq j \leq p$. **R** then concatenates them to recover m, broadcasts "SUCCESS" signal to **S** and terminates the protocol.
6. IF $\exists e \in \{1, 2, \ldots, p\}$, such that after correcting $\frac{t_b}{2}$ errors, the decoding algorithm detects additional faults (at most $\frac{t_b}{2}$) in the e^{th} received codeword, then **R** broadcasts the e^{th} received codeword, along with index e and "ERROR2" signal to **S**. **R** also broadcasts the identity of wires which failed to deliver any information and terminates. /* In this case, more than $\frac{t_b}{2}$ errors has occurred. If there are more than one e, then **R** randomly selects one.*/

Local Computation by S

• IF **S** receives "SUCCESS" signal, then it does nothing. IF **S** receives "ERROR1" signal, then **S** comes to know the identity of $n - a$ wires which failed to deliver any information to **R**. ELSE, **S** receives the e^{th} codeword, as received by **R** and after comparing it with original e^{th} codeword, **S** identifies more than $\frac{t_b}{2}$ Byzantine corrupted wires.

Lemma 3.1.4 *(***Correctness:***) In* **SP-REL***, if at most $t_f - k_f$ wires get fail-stop corrupted and at most $\frac{t_b}{2}$ wires get Byzantine corrupted, then* **R** *will be able to recover m. Otherwise,* **S** *will*

either come to know the identity of at least $t_f - k_f + 1$ *fail-stop or at least* $\frac{t_b}{2} + 1$ *Byzantine corrupted wires.*

PROOF: We consider the following three cases:

1. **More than** $t_f - k_f$ **wires get fail-stop corrupted during first phase:**
   In this case, irrespective of the number of Byzantine errors, **R** will come to know the exact identity of more than $t_f - k_f$ wires which are fail-stop corrupted and broadcasts their identity to **S**.

2. **At most** $t_f - k_f$ **and** $\frac{t_b}{2}$ **wires get fail-stop and Byzantine corrupted respectively during first phase:**
   We consider the worst case, where exactly $t_f - k_f$ wires get fail-stop corrupted. Thus, **R** will receive an $n' = n - (t_f - k_f)$ length RS codeword for each B_i, which is RS encoded using a polynomial of degree $k - 1 = (X - 2t_b) + \frac{t_b}{2} + k_f - 1$. Substituting the value of n' and k in Theorem 2.2.6, we find that RS decoding can correct $c = \frac{t_b}{2}$ and detect additional $d = \frac{t_b}{2}$ Byzantine errors in each codeword. Since the number of Byzantine errors in each codeword is at most $\frac{t_b}{2}$, the decoding algorithm will correct them (and does not detect any additional error) and recover each B_i (and hence m) correctly.

3. **At most** $t_f - k_f$ **wires get fail-stop corrupted but more than** $\frac{t_b}{2}$ **wires get Byzantine corrupted during first phase:**
   Suppose more than $\frac{t_b}{2}$ errors occur during the transmission of e^{th} codeword, where $e \in \{1, 2, \ldots, p\}$. In this case, from the previous argument, the decoding algorithm will correct $\frac{t_b}{2}$ errors and will detect additional errors (at most $\frac{t_b}{2}$) in the e^{th} received codeword. So **R** will come to know that more than $\frac{t_b}{2}$ errors occurred during the transmission of e^{th} codeword. So **R** broadcasts the e^{th} received codeword to **S**, who after locally comparing it with the original e^{th} codeword finds the identity of more than $\frac{t_b}{2}$ Byzantine corrupted wires.

□

Remark 5 *In* **SP-REL**, **R** *can reliably send the identity of fail-stop corrupted wires by broadcasting a bit vector of size* n*, where* i^{th} *bit of the vector is* $1(0)$*, if wire* w_i *(not) delivered any*

information to **R**. *This requires communicating* $O\left(n * \frac{n}{\log(|\mathbb{F}|)}\right)$ *field elements. In the rest of the report, whenever we use* **SP-REL**, *we assume that* **R** *sends back the identity of fail-stop wires using this* **bit-vector** *technique.*

Lemma 3.1.5 *The communication complexity of* **SP-REL** *is* $O\left(\frac{|m|n}{(X-2t_b)+\frac{t_b}{2}+k_f}\right)+O\left(\frac{n^2}{\log(|\mathbb{F}|)}\right)+O(n^2)$.

PROOF: Follows from the working of the protocol. □

SP-REL bring to the fore an important property (given in Corollary 3.1.6) which holds for any single phase PRMT protocol and is used to derive the lower bound on phase complexity in the next chapter.

Corollary 3.1.6 *Let* **S** *and* **R** *be connected by* $n \geq 2t_b + t_f + 1$ *wires and* **S** *wants to reliably send* m *to* **R**. *If* **S** *in advance knows that adversary will not do any Byzantine corruption (i.e.,* $t_b = 0$*) and will fail-stop at most* $t_f - k_f$ *wires,* $1 \leq k_f \leq t_f - 1$, *then* **S** *has to communicate at least* $\frac{N\ell}{(X+k_f)}$ *field elements to reliably send* m. *Moreover the minimum number of wires that the adversary needs to fail-stop in order that* **R** *does not recover* m *is* $t_f - k_f + 1$. *Thus if* **S** *does not know in advance the number of fail-stop corruptions the adversary may perform,* $\frac{N\ell}{X+k_f}$ *is a trivial lower bound on the number of field elements to be sent by* **S**, *so that either* **R** *recovers* m *or comes to know the identity of at least* $t_f - k_f + 1$ *fail-stop corrupted wires The above expression can also be viewed as* $\ell \times \frac{X+t_f}{X+k_f}$.

CHAPTER 4

Lower Bound

4.1 Lower Bound on Phase Complexity of PRMT Against $\mathcal{A}_{(t_b,t_f)}$

We now derive a nontrivial lower bound on phase complexity for any PRMT protocol which transmits ℓ field elements by communicating $O(b)$ field elements against $\mathcal{A}_{(t_b,t_f)}$, where $\ell \leq b < n\ell$. Recall that according to the definition of PRMT, **R** should correctly output the message with probability one (no error probability). We assume that the protocol specification is public and adversary is also aware of the steps of the protocol. Accordingly, adversary devices his strategy. However, adversary has no access to the internal random coin tosses of **S** and **R**. Without loss of generality, we assume that during each phase, the information sent over each wire is of same length because using a similar argument given in Remark 4, we can show that any PRMT protocol which sends un-equal sized information over each wire, does no better than a protocol which sends equal sized information over each wire.

Theorem 4.1.1 *Let* **S** *and* **R** *be connected by* $n \geq 2t_b + t_f + 1$ *wires such that* $t_f > 0$ *and* $t_f > (n - t_f)$. *Then any PRMT protocol from* **S** *to* **R** *under the influence of* $\mathcal{A}_{(t_b,t_f)}$, *must run for* $\Omega\left(\frac{\log(\frac{t_f}{n-t_f})}{\log(\frac{cb}{\ell})}\right) = \Omega(\mathcal{D})$ *phases for transmitting* m, *where* $|m| = \ell$, *with a communication complexity of* $O(b)$ *field elements, where* $\ell \leq b < n\ell$ *and* $c > 1$ *is a positive constant.*

Remark 6 *The lower bound of* $\Omega(\mathcal{D})$ *phases does not hold good if* $t_f = 0$. *If* $t_f = 0$, *then* $n \geq 2t_b + 1$. *So from Theorem 2.1.3, three phase is necessary and sufficient to optimally send* ℓ *field elements by communicating* $O(\ell)$ *field elements. The lower bound also does not holds good if* $t_f \leq (n - t_f)$. *If* $t_f \leq (n - t_f)$, *then we can send* ℓ *field elements by communicating* $O(\ell)$ *field elements in constant phases. A three phase PRMT protocol for this case is given in section 5.2 (see Corollary 5.2.2).*

PROOF: We present an adversarial behavior, against which no PRMT protocol can send m with a communication complexity $O(b)$ in less than $\Omega(\mathcal{D})$ phases. The Byzantine adversary remain passive throughout the protocol. Any lower bound derived with this assumption is surely a lower bound when Byzantine adversary is active. So, here only fail-stop adversary is active. Once a wire (corrupted in fail-stop fashion) fails to deliver information through it, it is marked as faulty wire and can be removed from the set of *active* (currently used) wires. Therefore, in the beginning of any protocol, number of active wires is n and number of *undisclosed* fail-stop wires is t_f. Whenever a wire (fail-stop corrupted) stops the communication, it reveals its corrupted status and thus number of *undisclosed* fail-stop wires reduces. Let the number of *undisclosed* fail-stop wires after i^{th} disclosure be denoted by L_i. Initially $L_0 = t_f$. Informally, the adversarial strategy is as follows: During each phase, adversary checks how much portion of m, **S** is trying to send to **R**. This he can find out from the protocol specification. If the size of the portion that **S** tries to send during a particular phase is more than a "specific" limit, then the adversary does the minimum number of fail-stop corruption, so that **R** can recover only a specific "sub-portion" of the portion sent by **S**. Otherwise, the adversary does no fail-stop corruption. Specifically, after the i^{th} disclosure of fail-stop corrupted wires, adversary does the following:

- *If* **S** *tries to send a portion of size* $q \leq \frac{\ell \log(\frac{cb}{\ell}) \log(t_f)}{\log(L_i) \log(\frac{t_f}{n-t_f})} = \frac{\ell \log(t_f)}{\log(L_i)\mathcal{D}}$ *then adversary does nothing.*

- *Otherwise adversary tries to fail-stop the minimum number of wires so that* **R** *can recover only* $\frac{\ell \log(t_f)}{\log(L_i)\mathcal{D}}$ *portion of the total message that has been sent by* **S** *(the total message is strictly greater than* $\frac{\ell \log(t_f)}{\log(L_i)\mathcal{D}}$*). If this is not possible then adversary will fail-stop in such a way that the number of undisclosed fail-stop wires reduces to* t_f^{ϵ}*, where* ϵ *is a fixed positive fraction and then remains inactive in the rest of the protocol execution.*

In the sequel, we consider three possible cases and prove for each of the cases the number of phases is indeed $\Omega(\mathcal{D})$ and also show that such an adversarial act is mountable against any protocol.

Remark 7 *Throughout our computation, we use logarithm to the base* e*. This is because it*

will make the use of calculus easier. Also this wont change the lower bound by more than a constant factor.

1. **Claim 4.1.2** *If adversary does not become active throughout the protocol, then the protocol terminates in $\Omega(\mathcal{D})$ phases.*

   PROOF: The reason why adversary remained inactive throughout the protocol is that **S** never tried to transmit more than $\frac{\ell \log(t_f)}{\log(t_f)\mathcal{D}}$ field elements in any phase. Hence, denoting the maximum message size as q that has been sent in any phase, we get,

   $$q \leq \frac{\ell \log(t_f)}{\log(L_0)\mathcal{D}} \leq \frac{\ell \log(t_f)}{\log(t_f)\mathcal{D}} = \frac{\ell}{\mathcal{D}} \text{ because } L_0 = t_f$$

   So the minimum number of phases P, required to send ℓ field elements is given by, $P \geq \frac{\ell}{q} = \mathcal{D} = \Omega(\mathcal{D})$. Hence the claim. □

2. **Claim 4.1.3** *If the adversary fail-stop in such a way that the number of undisclosed fail-stop wires at the end of the protocol remains strictly greater than t_f^ϵ, then the protocol runs $\Omega(\mathcal{D})$ phases.*

   PROOF: Clearly, in this case the maximum message size sent by **S** in any phase is given by,

   $$q \leq \frac{\ell \log(t_f)}{\log(t_f^\epsilon)\mathcal{D}} = \frac{\ell \log(t_f)}{\epsilon \log(t_f)\mathcal{D}} = \frac{\ell}{\epsilon\mathcal{D}}$$

   So even if at most q field elements are communicated in each phase, the protocol takes $P \geq \frac{\ell}{q} = \epsilon\mathcal{D} = \Omega(\mathcal{D})$ phases to send ℓ field elements. Hence the claim. □

3. **Claim 4.1.4** *If the adversary fail-stop in such a manner that the number of undisclosed fail-stop wires becomes less than or equal to t_f^ϵ, then also the protocol takes $\Omega(\mathcal{D})$ phases.*

   PROOF: So in this case the number of undisclosed fail-stop wires reduces down to less than or equal to t_f^ϵ. Suppose the adversary uses k phases denoted as $Ph_1, Ph_2, \ldots, Ph_k$ to reduce the number of undisclosed wires from t_f to t_f^ϵ. We will show that k, the number of phases in which fail-stop corruption occurs is $\Omega(\mathcal{D})$, excluding other phases where adversary does nothing. Let the number of undisclosed wires after Ph_i reduces from L_{i-1} to L_i. So recording the count of undisclosed wires after every disclosure (of adversary) starting with the initial count of $L_0 = t_f$, we get a decreasing sequence

$t_f, L_1, L_2, \ldots, L_{k-1}, t_f^{\epsilon}$. Let $\alpha_1, \alpha_2, \ldots, \alpha_k$ be the number of field elements communicated by **S** in the corresponding phases. In all these k phases, **S** must have tried to send more than $\frac{\ell \log(t_f)}{\log(L_i)\mathcal{D}}$ field elements of the message and the adversary exposed the minimum number of wires (hence after Ph_i reducing number of undisclosed wires from L_{i-1} to L_i) such that only $\frac{\ell \log(t_f)}{\log(L_i)\mathcal{D}}$ field elements of the total message is recoverable by **R**. Then by Corollary 3.1.6, the number of field elements $\alpha_i, 1 \leq i \leq k$ transmitted in Ph_i is given by $\alpha_i \geq \frac{\ell \log(t_f)}{\log(L_{i-1})\mathcal{D}} \times \frac{X+L_{i-1}}{X+L_i}$. This is so because during phase Ph_i, the number of active wires is $X+L_{i-1}$, the number of unknown errors is L_{i-1} and **S** tried to sent at least $\frac{\ell \log(t_f)}{\log(L_{i-1})\mathcal{D}}$ field elements. The above mentioned attack by the adversary is mountable since adversary is aware of α_i (adversary knows the protocol specification) and can solve $\frac{\ell \log(t_f)}{\log(L_i)\mathcal{D}} \times \frac{X+L_{i-1}}{X+x} = \alpha_i$ for x and accordingly blocks only $L_{i-1} - x$ wires so that **R** recovers only $\frac{\ell \log(t_f)}{\log(L_i)\mathcal{D}}$ field elements of the message and L_{i-1} reduces to L_i. Since the communication complexity of the protocol is $O(b)$, the sum of all α_i's should

be bounded by $d\,b$, for some constant $d \geq 1$. Hence

$$
\begin{aligned}
d\ b \geq \sum_{i=1}^{k} \alpha_i &\geq \frac{\ell \log(t_f)}{\log(t_f)\mathcal{D}} \times \frac{X+t_f}{X+L_1} + \frac{\ell \log(t_f)}{\log(L_1)\mathcal{D}} \times \frac{X+L_1}{X+L_2} + \ldots \\
&\quad + \frac{\ell \log(t_f)}{\log(L_{k-1})\mathcal{D}} \times \frac{X+L_{k-1}}{X+t_f^{\epsilon}} \\
&\geq \frac{\ell \log(t_f)}{\log(t_f)\mathcal{D}} \times \frac{X+t_f}{X+L_1} + \frac{\ell \log(t_f)}{\log(t_f)\mathcal{D}} \times \frac{X+L_1}{X+L_2} + \ldots \\
&\quad + \frac{\ell \log(t_f)}{\log(t_f)\mathcal{D}} \times \frac{X+L_{k-1}}{X+t_f^{\epsilon}} \ \text{/* } \frac{1}{\log(L_i)} \geq \frac{1}{\log(t_f)} \text{ */} \\
&\geq \frac{\ell \log(t_f)}{\log(t_f)\mathcal{D}} \left[\frac{X+t_f}{X+L_1} + \frac{X+L_1}{X+L_2} + \ldots + \frac{X+L_{k-1}}{X+t_f^{\epsilon}} \right] \\
\frac{d\,b\mathcal{D}}{\ell} &\geq \frac{X+t_f}{X+L_1} + \frac{X+L_1}{X+L_2} + \ldots + \frac{X+L_{k-1}}{X+t_f^{\epsilon}} \\
\frac{d\,b\mathcal{D}}{\ell k} &\geq \frac{(\frac{X+t_f}{X+L_1} + \frac{X+L_1}{X+L_2} + \ldots + \frac{X+L_{k-1}}{X+t_f^{\epsilon}})}{k} \ \text{/* Dividing both side by } k * / \\
&\geq (\frac{X+t_f}{X+L_1} \times \frac{X+L_1}{X+L_2} \times \ldots \times \frac{X+L_{k-1}}{X+t_f^{\epsilon}})^{\frac{1}{k}} \ \text{/*as AM} \geq \text{ GM */} \\
&= \left[\frac{X+t_f}{X+t_f^{\epsilon}} \right]^{\frac{1}{k}} \\
\text{Thus,} \quad \left[\frac{d\ b\mathcal{D}}{\ell k} \right]^k &\geq \left[\frac{X+t_f}{X+t_f^{\epsilon}} \right] = X \frac{1+(\frac{t_f}{X})}{1+\frac{t_f^{\epsilon}}{X}} \\
&\geq \frac{1+(\frac{t_f}{X})}{1+(\frac{t_f}{X})^{\epsilon}} \geq \frac{(\frac{t_f}{X})}{2(\frac{t_f}{X})^{\epsilon}} \geq \frac{1}{2^k} \left(\frac{t_f}{X} \right)^{(1-\epsilon)} \ \text{/* since } t_f > X \text{ */} \\
\left[\frac{2d\ b\mathcal{D}}{\ell k} \right]^k &\geq \left(\frac{t_f}{X} \right)^{1-\epsilon} \Rightarrow \left[\frac{a\ b \log(\frac{t_f}{n-t_f})}{\ell k \log(\frac{cb}{\ell})} \right]^k - \left(\frac{t_f}{n-t_f} \right)^{1-\epsilon} \geq 0 \ \text{/* } a = 2d\,,\, a \geq 2 * /
\end{aligned}
\tag{4.1}
$$

Let $Y = \left[\frac{a\ b \log(\frac{t_f}{n-t_f})}{\ell k \log(\frac{cb}{\ell})} \right]^k - \left(\frac{t_f}{n-t_f} \right)^{1-\epsilon}$. So, for our desired PRMT protocol the value of k should be such that Y is non-negative.

Lemma 4.1.5 *Any protocol which takes k rounds the corresponding value of Y should be non negative.*

PROOF: By the above argument. □

Lemma 4.1.6 $Y = \left[\frac{a\ b \log(\frac{t_f}{n-t_f})}{\ell k \log(\frac{cb}{\ell})} \right]^k - \left(\frac{t_f}{n-t_f} \right)^{1-\epsilon}$ *is an increasing function for all* $k \leq$

$\frac{ab\log(\frac{t_f}{n-t_f})}{\ell\, e\log(\frac{cb}{\ell})} = \frac{ab\mathcal{D}}{e\,\ell}$

PROOF:

$$\begin{aligned} Y &= \left[\frac{a\ b\log(\frac{t_f}{n-t_f})}{\ell k\log(\frac{cb}{\ell})}\right]^k - \left(\frac{t_f}{n-t_f}\right)^{1-\epsilon} \qquad (4.2)\\ Y &= \left(\frac{Z}{K}\right)^k - Z' \quad \text{where } Z = \frac{a\ b\log(\frac{t_f}{n-t_f})}{\ell\log(\frac{cb}{\ell})} \quad \text{and } Z' = \left(\frac{t_f}{n-t_f}\right)^{1-\epsilon}\\ \log(Y+Z') &= k\log(Z) - k\log(k)\\ \frac{1}{Y+Z'}\frac{dY}{dK} &= \log(Z) - k\log(k) - 1\\ \frac{dY}{dK} &= \left(\frac{Z}{k}\right)^k\left[\log(\frac{Z}{K\ e})\right] \end{aligned}$$

Now putting $\frac{dY}{dK} \geq 0$, we get

$$\begin{aligned} \left(\frac{Z}{k}\right)^k\left[\log(\frac{Z}{K\ e})\right] &\geq 0 \qquad (4.3)\\ \Rightarrow \frac{Z}{K\ e} &\geq 1\\ \Rightarrow k &\leq \frac{Z}{e} = \frac{a\ b\log(\frac{t_f}{n-t_f})}{e\ \ell\log(\frac{cb}{\ell})} = \frac{ab\mathcal{D}}{e\ \ell} \end{aligned}$$

□

Lemma 4.1.7 $\frac{\mathcal{D}}{ac} < \frac{ab\mathcal{D}}{e\ \ell}$

PROOF: Recall that $c \geq 1$ and $a \geq 2$. Also $e = 2.73...$ Thus,

$$a \geq 2 \Rightarrow a^2 \geq 4 \Rightarrow 1 \geq \frac{4}{a^2} \Rightarrow c > \frac{4}{a^2} > \frac{e}{a^2} \Rightarrow \frac{a}{e} > \frac{1}{ac} \qquad (4.4)$$

Hence $\frac{\mathcal{D}}{ac} < \frac{a\mathcal{D}}{e} < \frac{ab\mathcal{D}}{\ell\, e}$, since $b \geq \ell$ □

Lemma 4.1.8 *The value of* $Y = \left[\frac{ab\mathcal{D}}{\ell k}\right]^k - \left(\frac{t_f}{n-t_f}\right)^{1-\epsilon}$ *is non-positive at* $k = \frac{\mathcal{D}}{a\ c}$ *for all* $a \geq 2$ *and some specific positive fraction* ϵ.

PROOF: We prove this by contradiction. So let Y be non-negative at $k = \frac{\mathcal{D}}{a\ c}$. This implies at $k = \frac{\mathcal{D}}{a\ c}$, $\left[\frac{ab\mathcal{D}}{\ell k}\right]^k \geq \left(\frac{t_f}{n-t_f}\right)^{1-\epsilon}$ holds. So, putting $k = \frac{\mathcal{D}}{a\ c}$ in this relation and

simplifying we get,

$$\begin{aligned}
\left[\frac{a^2bc}{\ell}\right]^{\frac{\mathcal{D}}{a\,c}} &\geq \left(\frac{t_f}{n-t_f}\right)^{1-\epsilon} \\
\frac{\mathcal{D}}{ac}\left[\log(\frac{cb}{\ell})+2\log(a)\right] &\geq (1-\epsilon)\log(\frac{t_f}{n-t_f}) \text{ /* Taking log on both sides */} \\
\log(\frac{cb}{\ell})\left[\frac{1}{ac}-(1-\epsilon)\right]+\frac{2\log(a)}{ac} &\geq 0 \text{ /* putting value of } \mathcal{D} \text{ and simplifying */} \qquad (4.5)
\end{aligned}$$

As the value of c and ϵ are under the control of adversary he chooses then to be $c = a^{10} + 1\{c \geq 2^{10}+1\}$ and $\epsilon = 0.4$.

$$\begin{aligned}
\frac{2\log(a)}{ac} &\geq \log(\frac{cb}{\ell})\left[(1-0.4)-\frac{1}{ac}\right] \text{ /* } \epsilon = 0.4 \text{ */} \qquad (4.6) \\
\frac{2\log(a)}{ac} &\geq \log(c)\left[0.6-\frac{1}{ac}\right] \text{ /* } b \geq l \text{ */} \\
\frac{2\log(a)}{ac} &\geq 10\times\log(a)\left[0.6-\frac{1}{ac}\right] \text{ /* } c \geq a^{10} \text{ */} \\
\frac{2}{ac} &\geq 6-\frac{10}{ac} \text{ /* canceling } \log(a) \text{ on both sides*/} \\
\frac{12}{ac} &\geq 6 \\
2 &\geq ac
\end{aligned}$$

The final equation is obviously wrong as $a \geq 2$ and $c = a^{10} + 1$.Hence this is a contradiction that Y is non-negative at $k = \frac{\mathcal{D}}{a\,c}$. Hence the lemma. □

By Lemma 4.1.6, 4.1.7, 4.1.8 we can see that Y is negative for all values of $k \leq \frac{\mathcal{D}}{ac}$. Hence by Lemma 4.1.5, any valid protocol Y should be non-negative for which $k > \frac{\mathcal{D}}{a\,c} = \Omega(\mathcal{D})$.

Thus, in all three cases, we proved that the number of phases required is $\Omega(\mathcal{D})$. This completes the proof of Theorem 4.1.1. □

We now dispose two important corollaries of our lower bound result.

Corollary 4.1.9 *Any PRMT protocol over $n = t_f + 1$ wires, influenced by $\mathcal{A}_{t_f}$ (i.e., $t_b = 0$), must run for $\Omega(\log(t_f))$ phases to reliably send ℓ field elements by communicating $O(\ell)$ field elements.*

Thus against only fail-stop adversary over a minimal connected network, it takes $\log(t_f)$ phases to achieve reliability with *constant factor* overhead. Comparing this with Theorem 2.1.3,

we find that even though fail-stop adversary is much weaker than Byzantine adversary, it takes more phases against fail-stop adversary to achieve reliability with *constant factor* overhead. The reason behind this surprising result was explained earlier in Chapter 3. Theorem 2.1.3 and Corollary 4.1.9 defines the phase complexity lower bounds for two extreme cases. The intermediate scenario is captured by the following corollary which brings out the fundamental inherent trade-off between phase complexity and communication complexity in the presence of $\mathcal{A}_{(t_b,t_f)}$.

Corollary 4.1.10 *Let* **S** *and* **R** *be connected by* $2t_b + t_f + 1$ *wires, influenced by* $\mathcal{A}_{(t_b,t_f)}$*, such that* $t_b, t_f > 0$ *and* $t_f > (n - t_f)$*. Then any PRMT protocol must run for* $\Omega(\log(\frac{t_f}{t_b}))$ *phases to reliably send* ℓ *field elements by communicating* $O(\ell)$ *field elements.*

CHAPTER 5

Upper Bound

5.1 Upper Bound on Phase Complexity of PRMT Against $\mathcal{A}_{(t_b,t_f)}$

Let **S** and **R** be connected by $n \geq 2t_b + t_f + 1$ wires, under the influence of $\mathcal{A}_{(t_b,t_f)}$. We now provide a PRMT protocol **PRMT-II**, which terminates in $O(\mathcal{D})$ phases and sends a *sufficiently large* message m containing ℓ field elements (ℓ will be fixed shortly) by communicating $O(b)$ field elements where $\ell \leq b < n\ell$ and $\mathcal{D} = \frac{\log(\frac{t_f}{n-t_f})}{\log(\frac{cb}{\ell})}$. This shows that the bound proved in Chapter 4 is tight. We first design a three phase PRMT protocol called **PRMT-I**, which is used in **PRMT-II**.

5.2 A Three Phase PRMT Protocol

In protocol **PRMT-I**, **S** and **R** are connected by $N \geq 2t_b + T + 1$ wires, of which at most t_b could be Byzantine corrupted and $T \leq t_f$ could be fail-stop corrupted. Also $n - t_f = N - T = X$. The protocol reliably sends M containing L field elements by communicating $O(\frac{NL}{X})$ field elements, where $L \geq N^2$.

Theorem 5.2.1 *Protocol* **PRMT-I** *reliably sends M, containing L field elements, in at most three phases by communicating $O(\frac{NL}{X})$ field elements, where $X = (n-t_f) = N-T \geq 2t_b+1$ and $L \geq N^2$.*

PROOF: We prove the theorem for the worst case, when during **Phase I**, the fail-stop adversary blocks all T wires under its control. So during **Phase I**, **R** receives $n' = N-T$ values for each $B_j, 1 \leq j \leq z$, which are RS encoded using polynomials of degree $k-1 = (X-2t_b)+\frac{t_b}{2}-1$. Putting the values of n' and k in Theorem 2.2.6, we find that RS decoding can correct $c = \frac{t_b}{2}$ errors and simultaneously detect additional $d = \frac{t_b}{2}$ errors in each of the z received codewords. If at most $\frac{t_b}{2}$ errors occur during first phase, then the decoding algorithm will correct these

errors and will not detect additional errors in any received codeword. So **R** will recover each B_j (and hence m).

On the other hand, if more than $\frac{t_b}{2}$ errors occur during **Phase I**, then $\exists e, e \in \{1, 2, \ldots, z\}$, such that e^{th} received codeword contains more than $\frac{t_b}{2}$ corrupted values. So, RS decoding will correct $\frac{t_b}{2}$ errors and simultaneously detect the remaining faults (which is at most $\frac{t_b}{2}$). Hence **R** sends back e^{th} received codeword to **S**, who after comparing it with the original codeword of B_e, finds the identity of more than $\frac{t_b}{2}$ faults and saves them in L_{fault}. By broadcasting L_{fault}, **S** informs the identity of these faulty wires to **R**. Finally **S** re-sends M by using **PRU-SP**$(M, |M|, N, t_b, T, |L_{fault}|)$, which from Theorem 3.1.3 will be executed successfully.

During **Phase I**, **S** communicates $O\left(\frac{|M|}{(X-2t_b)+\frac{t_b}{2}} * N\right) = O(\frac{NL}{X})$ field elements because $|M| = L$. During **Phase II**, **R** communicates $O(N^2)$ field elements by broadcasting the e^{th} received codeword. During **Phase III**, **S** re-sends M by executing **PRU-SP**$(M, |M|, N, t_b, T, |L_{fault}|)$, which communicates $O\left(\frac{|M|}{(X-2t_b)+|L_{fault}|} * N\right) = O(\frac{NL}{X})$ field elements because $\frac{t_b}{2} < |L_{fault}| \leq t_b$. Since, $L \geq N^2$, the total communication complexity is $O(\frac{NL}{X})$. □

Corollary 5.2.2 *1. If $X \geq T$, then $X = \Theta(N)$ because $N = X + T$ [1]. Hence, in this case protocol* **PRMT-I** *sends L field elements by communicating $O(L)$ field elements, where $L \geq N^2$.*

2. If $T > X$, then $T = \Theta(N)$ because $N = X + T$. Hence, in this case protocol **PRMT-I** *sends L field elements by communicating $O(\frac{LT}{X})$ field elements, where $L \geq N^2$.*

5.3 A Worst Case $O(\mathcal{D})$ Phase PRMT Protocol

Here **S** and **R** are connected by $n \geq 2t_b + t_f + 1$ wires. The protocol **PRMT-II** has the following properties: (i) If $X \geq t_f$, then it sends m in three phases by communicating $O(|m|)$ (recall $b \geq |m| = \ell$) field elements. (ii) If $X < t_f$, then either it reliably sends the message m or reduces the number unknown fail-stop errors from t_f to X, both in $O(\mathcal{D})$ phases by communicating $O(b)$ field elements, where $|m| = \ell$, $\mathcal{D} = \frac{\log(\frac{t_f}{n-t_f})}{\log(\frac{cb}{\ell})}$ and $b \geq |m|$. In the later case when fail-stop faults reduces from t_f to X, the message is sent using three phase PRMT protocol **PRMT-I**. To design the protocol, we use protocol **SP-REL** (Chapter 3) as a black-box. *In* **PRMT-II**, *N and T denote the number of active wires (which are used in the protocol)*

Protocol PRMT-I - A Three Phase PRMT Protocol

Phase I: S to R

1. **S** divides M into blocks $B_1, B_2, \ldots, B_z$, each of size $(X - 2t_b + \frac{t_b}{2})$, where $z = \lceil \frac{|M|}{X-2t_b+\frac{t_b}{2}} \rceil$. For each $B_j, 1 \leq j \leq z$, **S** computes a RS codeword of size N, denoted by $[C_{j1}\ C_{j2}\ \ldots\ C_{jN}]$. Finally, through wire $w_i, 1 \leq i \leq N$, **S** sends $C_{ji}, 1 \leq j \leq z$.

Phase II: R to S

1. **R** receives information through N' wires where $N - T \leq N' \leq N$. **R** applies RS decoding algorithm to each of the received codewords and tries to correct $\frac{t_b}{2}$ errors and simultaneously detects additional $\frac{t_b}{2}$ errors in each codeword.
2. IF the decoding algorithm does not detect additional faults (after correcting $\frac{t_b}{2}$ errors) in any codeword, **R** correctly recovers each B_j, concatenates them, recovers M and terminates the protocol by broadcasting "SUCCESS" signal (see Theorem 5.2.1).
3. ELSE if $\exists e \in \{1, 2, \ldots, z\}$, such that decoding algorithm detects additional errors (after correcting $\frac{t_b}{2}$ errors) in the e^{th} received codeword, then **R** broadcasts an "ERROR" signal, the e^{th} received codeword and index e. **R** also sends an n length bit-vector, indicating the index of the wires over which it has not received any information. /* If there are more than one e, then **R** randomly selects one. */

IF **S** receives "SUCCESS" signal, then **S** terminates the protocol. ELSE **S** receives "ERROR" signal, index e, e^{th} codeword as received by **R** in **Phase I** and initiates **Phase III** as follows:

Phase III: S to R

1. **S** compares the original and the received (from **R**) e^{th} codeword to identify more than $\frac{t_b}{2}$ faulty wires (which delivered incorrect information to **R** during **Phase I**). **S** saves the identity of corrupted wires in a list L_{fault}. **S** broadcasts L_{fault} to **R**, re-sends M by executing **PRU-SP**$(M, |M|, N, t_b, T, |L_{fault}|)$ and terminates the protocol.

Local Computation by R (If it has Sent "ERROR" Signal During Phase II)

R correctly receives L_{fault} and comes to know the identity of $|L_{fault}| > \frac{t_b}{2}$ Byzantine corrupted wires. **R** finally recovers M as in **PRU-SP**$(M, |M|, N, t_b, T, |L_{fault}|)$ and terminates the protocol.

and the number of unknown fail-stop errors respectively during the execution of the protocol. Notice that N and T always maintain the relationship $N - T = n - t_f = X$. Both N and T are global variables which are updated by **S** *and* **R** *in parallel, after every disclosure of fail-stop corrupted wires. Initially $N = n$ and $T = t_f$.* **S** and **R** also maintain a list of active wires which is periodically updated during the execution of **PRMT-II**. We now describe **PRMT-II**.

Size of ℓ: If **PRMT-II** calls **PRMT-I** to send M', then M' will contain at least one chunk of m of size $\frac{\ell}{\mathcal{D}}$. Since **PRMT-I** requires the minimum message size (L) to be N^2 (where $N \leq n$), we take $|m| = \ell = n^3$, which ensures that $|M'| = \frac{n^3}{\mathcal{D}} \geq n^2 \geq N^2$.

Protocol PRMT-II (m, n, t_b, t_f)**: S and R are connected by** $n \geq 2t_b + t_f + 1$ **wires**

1. IF $X \geq t_f$, then **S** sends m by executing the three phase **PRMT-I** protocol (see Lemma 5.3.1). /* $X = n - t_f$ */
2. IF $(X < t_f)$ AND $(\mathcal{D} < 1)$, then **S** sends m by executing the three phase PRMT protocol **PRMT-I** (see Lemma 5.3.2)
3. IF $(X < t_f)$ AND $(\mathcal{D} \geq 1)$, then **S** and **R** does the following:
   (a) **S** and **R** initializes $N = n, i = 1$ and $T = t_f$. **S** breaks m into chunks $B_1, B_2, \ldots, B_q$ each of size $\frac{\ell}{\mathcal{D}}$ (so $q = \mathcal{D}$).
   (b) While $(i \leq q)$ AND $(T > X)$ DO /* X is now $N - T = n - t_f$ */
      i. **S** sets $k_f = T\frac{\ell}{cb}$ and executes **SP-REL**(B_i, N, t_b, T, k_f) by using the block size as $k = (X - 2t_b) + \frac{t_b}{2} + k_f$ and waits for feed-back. /* Recall that k is the block-size which is used in **SP-REL**.*/
      ii. **Success:** If **S** receives "SUCCESS" signal, then **S** increments i and continue with the next iteration. /* **S** concludes that **R** has received B_i correctly */
      iii. **Failure because more than** $T - k_f$ **wires are blocked by the adversary:** If **S** receives "ERROR1" signal, a value a and an n length bit vector (whose i^{th} position equal to 0 indicates that **R** has received no information over wire w_i), then **S** and **R** globally sets $N = N - a, T = T - a$. **S** and **R** also remove the a wires (whose corresponding bit value in the bit-vector is 0) from the list of active wires
      iv. **Failure due to the occurrence of more than** $\frac{t_b}{2}$ **Byzantine Fault:** If **S** receives "ERROR2" signal, index e and e^{th} codeword (as received by **R**), then after doing local verification, **S** identifies more than $\frac{t_b}{2}$ Byzantine corrupted wires, which it saves in a list L_{fault}. **S** then broadcasts L_{fault} to **R** and then both **S** and **R** globally set $N = N - |L_{fault}|$ and remove the wires in L_{fault} from the list of active wires. Now number of Byzantine faulty wires in the set of active wires is less than $\frac{t_b}{2}$. /* T remains unchanged in this case. */
   (c) If $i > q$ then terminate the protocol. Else **S** sends the remaining portion of the message, say M', consisting of chunks $B_i, B_{i+1}, \ldots, B_q$ by executing the three phase PRMT protocol **PRMT-I**.

The analysis of **PRMT-II** is divided into two cases: (i) When $X \geq t_f$ (ii) When $X < t_f$. The second case has further two sub-cases, depending upon whether $\mathcal{D} \geq 1$ or $\mathcal{D} < 1$.

Lemma 5.3.1 *If* $X \geq t_f$*,* **PRMT-II** *sends* m *in three phases by communicating* $O(\ell)$ *field elements.*

PROOF: Follows from Corollary 5.2.2(1), by substituting $N = n, T = t_f$ and $L = |m|$, where $|m| = n^3$. □

Lemma 5.3.2 *If* $X < t_f$ *and* $\mathcal{D} < 1$ *then* **PRMT-II** *sends* m *in three phases by communicat-*

ing $O(b)$ field elements.

PROOF: If $X < t_f$ then $n = \Theta(t_f)$ because $n = X + t_f$. Also if $\mathcal{D} < 1$ then it implies $\log(\frac{cb}{\ell}) > \log(\frac{t_f}{n-t_f}) \Rightarrow \frac{cb}{\ell} > \frac{t_f}{n-t_f} \Rightarrow b > \frac{t_f \ell}{c(n-t_f)} \Rightarrow b > \frac{t_f \ell}{cX}$. Thus, we require to send ℓ field elements by communicating $O\left(\frac{t_f \ell}{X}\right)$ field elements. Since $\ell \geq n^3$, **PRMT-II** have to send n^3 field elements by communicating $O\left(\frac{n^3 t_f}{X}\right)$ field elements. Now substituting $T = t_f, N = n, L = n^3$ and $N = \Theta(T)$ in Corollary 5.2.2(2), we find that this can be done in three phases by executing **PRMT-I**. □

Lemma 5.3.1 and Lemma 5.3.2 prove the properties of **PRMT-II** for two cases. Now we analyze the properties of **PRMT-II** for the case when $X < t_f$ and $\mathcal{D} \geq 1$. In this case, the execution sequence is as follows: **S** sequentially selects a chunk B_i of size $\frac{\ell}{\mathcal{D}}$ from the message m. **S** then executes **SP-REL** to send B_i using $(X - 2t_b) + \frac{t_b}{2} + k_f$ as the block-size, where $k_f = T\frac{\ell}{cb}$. In this process either (a) **R** recovers B_i or (b) **S** identifies at least $T - k_f + 1$ fail-stop corrupted wires or more than $\frac{t_b}{2}$ Byzantine corrupted wires, which subsequently **S** and **R** removes from their list of active wires. In the first case **R** receives B_i and then **S** selects the next chunk B_{i+1} and repeats the same process. In the later cases, depending upon the type of identified faulty wires, **S** re-sets the block-size and tries to re-send the same B_i by executing **SP-REL**. Note that if more than $\frac{t_b}{2}$ Byzantine corrupted wires are identified, then the same block size is used to re-send B_i, whereas if more than $T - k_f$ fail-stop corrupted wires are identified then block-size becomes $(X - 2t_b) + \frac{t_b}{2} + k_f$, where k_f is reduced at least by a factor of $(\frac{cb}{\ell})$. This process will continue until the entire m is received by **R** or the number of unknown fail-stop faults T becomes less than or equal to X. When number of unknown fail-stop faults T becomes less than or equal to X, **S** sends the remaining portion of m, say M', by executing the three phase PRMT protocol **PRMT-I**, which will optimally send M' by communicating $O(|M'|)$ field elements. Thus the principle of the protocol is as follows: In each phase, **S** tries to send a fixed portion of m using **SP-REL**. Now either **SP-REL** will be successful or both **S** and **R** comes to know the identity of some faulty wires. After repeating this process for $O(\mathcal{D})$ iterations **S** will be able to send m to **R**.

Remark 8 *In* **PRMT-II***, when* **SP-REL** *gets executed successfully,* **R** *does not communicate the identity of the wires which stops communication (at most $T - k_f$). Therefore, the list of*

active wires remains unchanged at both ends. This does not affect communication complexity and phase complexity of the protocol because if in some phase, **SP-REL** *is executed successfully, then* **S** *sends the next $\frac{\ell}{\mathcal{D}}$ portion of m using the same value of k, in the next phase. So if the number of errors remains same this time also,* **SP-REL** *will be executed successfully, otherwise more than $T - k_f$ errors will be revealed.*

Lemma 5.3.3 *In* **PRMT-II***, if* **SP-REL** *fails due to more than $\frac{t_b}{2}$ Byzantine errors, then in the remaining phases,* **SP-REL** *can fail only due to occurrence of more than $T - k_f$ fail-stop errors.*

PROOF: In the protocol, once **SP-REL** fails due to the occurrence of more than $\frac{t_b}{2}$ Byzantine errors, then both **S** and **R** will know the identity of these wires. Now in the remaining executions of **SP-REL**, at most $\frac{t_b}{2} - 1$ Byzantine errors can occur. If in all these remaining executions, each time at most $T - k_f$ fail-stop errors occur, then **R** will receive $n' = N - (T - k_f)$ values for each codeword, which are RS encoded using a polynomial of degree $k - 1 = (X - 2t_b) + \frac{t_b}{2} + k_f - 1$. Putting these values in Theorem 2.2.6, we find that decoding algorithm will be able to correct $c = \frac{t_b}{2}$ errors in the codewords and hence **SP-REL** will be successful. □

Lemma 5.3.4 *In* **PRMT-II***, if during any phase* **SP-REL** *fails due to occurrence of more than $T - k_f$ fail-stop errors, then T is reduced at least by a factor of $\frac{cb}{\ell}$.*

PROOF: From Lemma 3.1.4, if **SP-REL** fails due to the occurrence of more than $T - k_f$ fail-stop errors, then the number of unknown fail-stop errors reduces to at least $T - (T - k_f + 1) = k_f - 1 = T\frac{\ell}{cb} - 1$. □

Theorem 5.3.5 *If $X < t_f$ and $\mathcal{D} \geq 1$ then* **PRMT-II** *terminates in $O(\mathcal{D})$ phases.*

PROOF: In this case, the phase complexity of **PRMT-II** is bounded by the number of times **SP-REL** protocol is executed in the protocol. There are $q = \mathcal{D}$ chunks of m and **SP-REL** is executed at least once for each of them. If each chunk is sent successfully in a single attempt, then the phase complexity of **PRMT-II** is $O(\mathcal{D})$. On the other hand, from Lemma 5.3.4, each un-successful execution of **SP-REL** reduces the number of unknown fail-stop errors T at least by a factor of $\frac{cb}{\ell}$. Also, from Lemma 5.3.3, **SP-REL** can fail only *once* due to Byzantine errors. Hence the number of un-successful executions of **SP-REL** required to bring the num-

ber of unknown fail-stop errors T, from its initial value of t_f to $n - t_f (= X)$ is bounded by $O(\frac{\log(\frac{t_f}{n-t_f})}{\log(\frac{cb}{\ell})}) = O(\mathcal{D})$. Once, T becomes less than or equal to X, then the remaining message is sent in three phases by executing **PRMT-I**. Thus even if adversary alternately allows some (un)successful executions of **SP-REL** followed by some unsuccessful(successful) executions of **SP-REL**, the phase complexity of **PRMT-II** remains as $O(\mathcal{D})$. □

Theorem 5.3.6 *The communication complexity of* **PRMT-II** *is* $O(b)$*, where* $\ell \leq b < n\ell$ *and* $\ell \geq n^3$.

PROOF: In **PRMT-II**, if step 1 is executed, then from Lemma 5.3.1, the communication complexity of the protocol is $O(\ell) = O(b)$. Similarly, if step 2 is executed, then from Lemma 5.3.2, the communication complexity of the protocol is $O(b)$. However, if step 3 is executed, then the communication complexity of **PRMT-II** is computed as follows:

To send a chunk B_i of size $\frac{\ell}{\mathcal{D}}$, **S** executes **SP-REL** with block-size $k = (X - 2t_b) + \frac{t_b}{2} + k_f = (X - 2t_b) + \frac{t_b}{2} + T\frac{\ell}{cb}$. Also at every stage of the protocol $N = T + X$. From Lemma 3.1.5, the number of field elements sent by **S** during an execution of **SP-REL** is given by $\frac{\frac{\ell}{\mathcal{D}}}{(X-2t_b)+\frac{t_b}{2}+T\frac{\ell}{cb}} * (X + T)$. Since, $T > X$, by increasing the numerator and decreasing the denominator, the above expression is bounded by $\frac{\ell}{\mathcal{D}} * \frac{dT}{T\frac{\ell}{cb}} = \frac{dcb}{\mathcal{D}} = O\left(\frac{b}{\mathcal{D}}\right)$, where d is a positive constant. From Theorem 5.3.5, the overall phase complexity of **PRMT-II** is $O(\mathcal{D})$. Thus, the total number of field elements communicated by **S** is $O(b)$. In the protocol, each time **SP-REL** fails, **R** broadcasts an n length bit vector, which requires communicating $O\left(\frac{n^2}{\log(|\mathbb{F}|)}\right)$ (see Lemma 3.1.5 and Remark 5). Since, in **PRMT-II**, the number of failures of **SP-REL** is bounded by $O(\mathcal{D})$, the total number of field elements communicated by **R** in the protocol is $O\left(\frac{n^2\mathcal{D}}{\log(|\mathbb{F}|)}\right)$. Also, if **SP-REL** fails due to the occurrence of more than $\frac{t_b}{2}$ Byzantine errors, then **R** broadcasts a codeword, incurring a communication complexity of $O(n^2)$. Moreover, from Lemma 5.3.3, this will happen only once. Since $|m| = n^3$ and $b \geq |m|$, the overall communication complexity of **PRMT-II** is $O(b) + O\left(\frac{n^2\mathcal{D}}{\log(|\mathbb{F}|)}\right) + O(n^2) = O(b)$. □

CHAPTER 6

Conclusion

6.1 Conclusion

The efficiency of any PRMT protocol is expressed using three parameters; the connectivity of the underlying network (n), the number of phases (r) and the number of field elements (b) communicated during the execution of the protocol. One of the major contributions of this thesis is a novel way to establish a lower bound for the number of phases and bring out a relation describing trade-off existing between connectivity, communication complexity and phase complexity. The second major contribution is a protocol that works on any *sufficiently* connected network that simultaneously are phase optimal and communication optimal. As a significant and highly non-intuitive corollary is that any PRMT protocol requires $\Omega\left(\log(\frac{t_f}{t_b})\right)$ phases, to send ℓ field elements with communication complexity $O(\ell)$, in a network that is $2t_b + t_f + 1$ -connected. The lower bound in some sense shows that a very highly fault tolerant network cannot be built without an increase either in communication complexity or number of rounds needed for the communication.

6.2 Open Problems and Future Directions

We believe that the proof for lower bound is of independent interest which can be applied to solve problems in other models. The result of this work is negative. It is worthwhile to look into the following problems.

- Generalize this result to different models such as non-threshold model, hyper-graphs, random fault models, multi-cast. In the random fault model one looks at the expected behavior instead of worst case behavior. The result to multi-cast can be easily generalized if there is a efficient way to find deterministic network coding with certain nice

properties. Certain work for non-threshold adversary from a polynomial set has been done in [15].

- In this result we get a lower bound and a corresponding protocol albeit to some constant factor. To get a Protocol and a lower bound to exact constant would be worthwhile and would reveal additional insights into the problem. Also the minimum message size is quite large for the proposed protocol. One more important direction to pursue would be to bring this down as well as prove better lower bounds for very small message sizes.
- The result in this work concentrates on phase complexity. There has been some research done in the direction of round complexity in [2]. This problem is also important.

References

[**1**] Ashwinkumar B. V, Arpita Patra, Ashish Choudhary, Kannan Srinathan, and C. Pandu Rangan. On tradeoff between network connectivity, phase complexity and communication complexity of reliable communication tolerating mixed adversary.

[**2**] K. Srinathan. Secure distributed communication. PhD Thesis, IIT Madras, 2006.

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

[**4**] K. Menger. Zur allgemeinen kurventheorie. *Fundamenta Mathematicae*, 10:96–115, 1927.

[**5**] Y. Desmedt and Y. Wang. Perfectly secure message transmission revisited. In *Proc. of Advances in Cryptology: Eurocrypt 2002*, LNCS 2332, pages 502–517. Springer-Verlag, 2003.

[**6**] M. Franklin and R. Wright. Secure communication in minimal connectivity models. *Journal of Cryptology*, 13(1):9–30, 2000.

[**7**] K. Srinathan, A. Narayanan, and C. Pandu Rangan. Optimal perfectly secure message transmission. In *Proc. of Advances in Cryptology: CRYPTO 2004*, LNCS 3152, pages 545–561. Springer-Verlag, 2004.

[**8**] A. Patra, A. Choudhary, K. Srinathan, and C. Pandu Rangan. Constant phase bit optimal protocols for perfectly reliable and secure message transmission. In *Proc. of INDOCRYPT 2006*, volume 4329 of *LNCS*, pages 221–235. Springer Verlag, 2006.

[**9**] S. Agarwal, R. Cramer, and R. de Haan. Asymptotically optimal two-round perfectly secure message transmission. In C. Dwork, editor, *Proc. of Advances in Cryptology: CRYPTO 2006*, LNCS 4117, pages 394–408. Springer-Verlag, 2006.

[**10**] A. Patra, A. Choudhary, and C. Pandu Rangan. Constant phase efficient protocols for

secure message transmission in directed networks. In *ACM PODC*, pages 322–323, 2007.

[**11**] Nancy A. Lynch. *Distributed Algorithms*. HARCOURT ASIA PTE LTD, 2000.

[**12**] A. Narayanan, K. Srinathan, and C. Pandu Rangan. Perfectly reliable message transmission. *Information Processing Letters*, 11(46):1–6, 2006.

[**13**] A. Patra, AshwinKumar B. V, A. Choudhary, K. Srinathan, and C. Pandu Rangan. Bit optimal and phase optimal protocols for perfectly reliable message transmission in the presence of mixed adversary. Manuscript. Available at www.cs.iitm.ernet.in/∼ashishc/OPRMT.pdf.

[**14**] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error Correcting Codes*. North-Holland Publishing Company, 1978.

[**15**] Ravishankar Krishnaswamy and Pandurangan Chandrasekaran. Fault tolerant network coding. *ACM Symposium on Theory of Computing(STOC): Student Research Competition(SRC)*, 2007.

List of Publications from this Thesis

- [1] Ashwinkumar B.V, Arpita Patra, Ashish Choudhary, Kannan Srinathan and C. Pandu Rangan. On Tradeoff Between Network Connectivity, Phase Complexity and Communication Complexity of Reliable Communication Tolerating Mixed Adversary. To appear at ACM Symposium on Principles of Distributed Computing(PODC) 2008.