Non-Applicability of Pseudobasis for Designing Perfectly Secure Message Transmission Protocols Against Non-Threshold Adversary

Ashish Choudhury[1] **Kaoru Kurosawa**[2]
Arpita Patra[3]

[1]Applied Statistics Unit
Indian Statistical Institute Kolkata India
partho31@gmail.com, partho_31@yahoo.co.in

[2]Department of Computer and Information Sciences
Ibaraki University, Hitachi, Ibaraki Japan
kurosawa@mx.ibaraki.ac.jp

[3]Department of Computer Science
Aarhus University, Denmark
arpitapatra10@gmail.com, arpita@cs.au.dk

Abstract

In EUROCRYPT 2008, Kurosawa and Suzuki introduced a very fundamental and interesting concept in coding theory called *pseudobasis*, which was used to design the first ever efficient and communication optimal two round *perfectly secure message transmission* (PSMT) protocol tolerating a t-active *threshold adversary*. Recently in ASIACRYPT 2010, Yang and Desmedt designed first ever *efficient* two round and three round PSMT protocols tolerating *non-threshold adversary* in both directed and undirected network settings. The key idea behind their protocol is adaptation of the concept of pseudobasis from threshold settings to non-threshold settings. However, in this paper, we show that the concept of pseudobasis will not work in non-threshold settings. This automatically implies that the protocols presented by Yang et al. will fail to provide perfect reliability.

As additional contribution, we present *efficient* two round and three round PSMT protocols tolerating non-threshold adversary in undirected networks. Our two round protocol sends *single* message while our three round protocol can send *multiple* messages *concurrently*. Our protocols are conceptually simpler, without using pseudobasis and also does not use the idea of converting a *linear secret sharing scheme* (LSSS) into a *linear code*, as done in the protocols of Yang et al.

Keywords: Information Theoretic Security, Non-Threshold Adversary, Efficiency.

1 Introduction

In the problem of *perfectly secure message transmission* (PSMT), there exists a sender $\mathbf{S}$ and a receiver $\mathbf{R}$, who are connected by n *disjoint bi-directional* channels, denoted by $\mathcal{W} = \{w_1, \ldots, w_n\}$, which are also called as *wires*. There exists a *computationally unbounded* adversary who can listen and modify communication over "some" of these n channels, which are unknown to $\mathbf{S}$ and $\mathbf{R}$. Sender has a message $m^{\mathbf{S}}$, which is a sequence of ℓ elements 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 $m^{\mathbf{R}} = m^{\mathbf{S}}$ correctly (*perfect reliability*). Moreover, adversary should not get any information about $m^{\mathbf{S}}$ in *information theoretic sense* (*perfect secrecy*). Furthermore, this should hold irrespective of the behavior of the adversary over the channels under his control. PSMT problem was formulated by Dolev et al. [7] who assumed the presence of a t-active *threshold adversary*, denoted by $\mathcal{A}_t$, who can *actively* control at most t channels throughout the protocol. The work of Dolev et al. was

followed by [15, 5, 17, 1, 14, 8, 12, 4, 2]. However, all these works consider threshold setting. In the literature, there exists another adversarial setting called *non-threshold* setting.

1.1 Non-Threshold Adversarial Setting

Modeling the adversary by a threshold helps in easy characterization of protocols and it also helps in analyzing protocols. However, as mentioned in [9], modeling the (dis)trust in the network as a threshold adversary is not always appropriate because threshold protocol requires more *stringent* requirements than the reality [10]. A non-threshold adversary, also called as *general adversary*, denoted by $\mathcal{A}$ is characterized by an *adversary structure* Γ, which is a collection of subsets of channels that the adversary $\mathcal{A}$ can *potentially* corrupt. That is,

$$\Gamma = \{B \subset \mathcal{W} \mid \mathcal{A} \text{ can corrupt } B\}.$$

Moreover, we assume that if $B \in \Gamma$ and if $B' \subset B$, then $B' \in \Gamma$. It is easy to see that a threshold adversary $\mathcal{A}_t$ is a *special case* of non-threshold adversary, such that $|B| \leq t$, for each $B \in \Gamma$.

Definition 1 ($\mathcal{Q}^k$ Condition [9]) *We say that $\mathcal{A}$ characterized by Γ, satisfies $\mathcal{Q}^k$ condition with respect to $\mathcal{W}$, if there exists no k sets in Γ, which adds upto the whole set $\mathcal{W}$. That is:*

$$\forall B_1, \ldots, B_k \in \Gamma : B_1 \cup \ldots \cup B_k \neq \mathcal{W}.$$

Any PSMT protocol is analyzed by the following parameters: (a) **Round Complexity**: It is the total number of rounds taken by the protocol, where a round means a communication from **S** to **R** or vice-versa; (b) **Communication Complexity**: It is the total number of field elements exchanged between **S** and **R** during the protocol; (c) **Computational Complexity**: It is the amount of computation done by **S** and **R** during the protocol.

A PSMT protocol is called *efficient*, if its round complexity, communication complexity and computational complexity is *polynomial* in n and the size of adversary structure Γ.

1.2 PSMT Tolerating Non-Threshold Adversary

Designing PSMT protocols in non-threshold settings is comparatively difficult than in threshold settings and due to this, designing efficient PSMT protocols tolerating non-threshold adversary got relatively less attention. PSMT problem in non-threshold settings was first studied by Kumar et al. [10], where they have shown that any *r-round* PSMT protocol, where $r \geq 2$, is possible iff $\mathcal{A}$ satisfies $\mathcal{Q}^2$ condition with respect to $\mathcal{W}$. On the other hand, Desmedt et al. in [6] have shown that *single round* PSMT is possible iff $\mathcal{A}$ satisfies $\mathcal{Q}^3$ condition with respect to $\mathcal{W}$.

We now summarize constant round protocols against Q^2 adversary in Table. 1, 2 and 3.

Table 1: 3-Round PSMT against Q^2 adversary

Reference	Number of Rounds	message size	Comments
[11]	3	Single field element	Efficient
3-Round-Protocol-I [18]	3	Single field element	Efficient
Proposed in this paper	3	Multiple field elements	Efficient. Same communication complexity as previous two protocols

1.3 Motivation of Our Work and Our Contributions

From the above tables, we find that [18] showed four constant round PSMT protocols against $\mathcal{Q}^2$ adversary. The key ideas behind their protocols are as follows:

Table 2: 2-Round PSMT against Q^2 adversary

Reference	Number of Rounds	message size	comment
[11]	2	Single field element	Inefficient
2-Round-Protocol-I [18]	2	Single field element	Efficient
Proposed in this paper	2	Single field element	Efficient. Conceptually simpler than 2-Round-Protocol-I

Table 3: Flawed PSMT based on pseudobasis (see next section for details)

Reference	Number of Rounds	message size
3-Round-Protocol-II [18]	3	Multiple field elements
2-Round-Protocol-II [18]	2	Multiple field elements

- To design **2-Round-Protocol-I** and **3-Round-Protocol-I**, they used *linear secret sharing scheme* (LSSS) [3]. However, instead of directly using LSSS, they converted LSSS into linear codes [13] and then used the properties of the linear code. As a result, these protocols are conceptually involved. This motivates us to design efficient and constant round PSMT protocols tolerating non-threshold adversary, which are conceptually simpler.
- To design **2-Round-Protocol-II** and **3-Round-Protocol-II**, they again used the idea of converting a LSSS into a linear code. But in addition, they used the idea of *pseudobasis*. Here pseudobasis is a fundamental concept, which was introduced in [12] to design first ever *computationally efficient* and *communication optimal* two round PSMT protocol tolerating a *threshold adversary*. In [18], the authors claimed that the idea of pseudobasis (which was originally introduced in [12] for threshold settings) can be easily generalized in non-threshold settings also. However, they have not given any proof for this. This motivates us to check whether the concept of pseudobasis can indeed be generalized in non-threshold settings.

Our Contributions: In this paper, we make the following contributions:

1. We show that the concept of pseudobasis cannot be generalized to non-threshold settings, in contrary to what is claimed in [18]. This automatically implies that protocols **2-Round-Protocol-II** and **3-Round-Protocol-II** fail to provide perfect reliability and hence they are not PSMT protocols (see Table 3).
2. We present an *efficient and simple* 3 round PSMT protocol which can send *multiple* messages *concurrently*. Note that the previous 3 round correct protocols, namely the protocol of [11] and **3-Round-Protocol-I**, can send only a single message. Furthermore, our protocol involves the *same communication complexity* as theirs (see Table 1).
3. We present an *efficient and simple* 2 round PSMT protocol which can send a *single* message. Our protocol is *conceptually simpler* than **2-Round-Protocol-I**. Moreover, our protocol is *efficient* while the 2 round protocol of [11] is *inefficient* (see Table 2).

Open Problems: In this paper, we are only considering bidirectional channels. In [18], the authors have also considered directed graph, where certain channels (n channels) are directed from **S** to **R** and certain channels (u channels) are directed from **R** to **S**. They presented a *three* round protocol in this settings tolerating non-threshold adversary and the protocol uses the concept of pseudobasis. However, due to our contribution 1, we find that this three round protocol in directed network is not a PSMT protocol. It is left as an open problem to design an efficient and constant round PSMT protocol in directed networks tolerating a non-threshold adversary.

2 Linear Secret Sharing Scheme (LSSS)

In a *secret sharing scheme*, a dealer D distributes a secret $s \in \mathbb{F}$, to a set of n parties $\mathcal{P} = \{P_1, \ldots, P_n\}$ in such a way that some subsets of the participants (called as access sets) can reconstruct s from their shares, while the other subsets of the participants (called forbidden sets) have no information about s from their shares. The family of access sets is called an *access structure*. Moreover, we assume that access structure is monotone, which is defined as follows:

Definition 2 *An access structure Σ is monotone if $A \in \Sigma$ and $A' \supseteq A$, then $A' \in \Sigma$.*

Corresponding to the access structure Σ, we have the adversary structure $\Gamma = \Sigma^c$, where c denotes the complement. The sets in Γ are called as *forbidden* sets. There exists a *computationally unbounded* adversary $\mathcal{A}$, who can control any set in Γ.

A secret sharing scheme for any monotone access structure Σ can be realized by a linear secret sharing scheme (LSSS) [3] as follows: Let $\mathcal{M}$ be a $d \times e$ matrix over $\mathbb{F}$ and $\quad : \{1, \cdots, d\} \to \{1, \cdots, n\}$ be a labeling function, where $d \geq e$ and $d \geq n$.

Sharing algorithm:

1. To share a secret $s \in \mathbb{F}$, D first chooses a random vector $\vec{\rho} \in \mathbb{F}^{e-1}$ and compute a vector
$$\vec{v} = (v_1, \cdots, v_d)^T = \mathcal{M} \cdot \begin{pmatrix} s \\ \vec{\rho} \end{pmatrix}. \tag{1}$$
2. Let
$$\mathsf{LSSS}(s, \vec{\rho}) = (\mathtt{share}_1, \cdots, \mathtt{share}_n), \tag{2}$$
where $\mathtt{share}_i = \{v_j \mid \quad (j) = i\}$. The dealer gives $\mathtt{share}_i$ to P_i as a share for $i = 1, \cdots, n$.

Reconstruction algorithm: A set of parties $A \in \Sigma$ can reconstruct the secret s if and only if $(1, 0, \cdots, 0)$ is in the linear span of
$$\mathcal{M}_A = \{m_j \mid \quad (j) \in A\},$$
where m_j denotes the jth row of $\mathcal{M}$. If this is indeed the case then there exists a vector $\vec{\alpha}_A$ called *recombination vector*, such that $\vec{\alpha}_A \cdot \mathcal{M}_A = (1, 0, \ldots, 0)$. Let $\vec{s}_A$ denote the set of shares corresponding to the parties in A. Then the parties in A can reconstruct s by computing $s = \langle \vec{\alpha}_A, \vec{s}_A^T \rangle$, where $\langle x, y \rangle$ denotes *dot product* of x and y and x^T denotes transpose of x.

Definition 3 (Monotone Span Programme (MSP) [3]) *We say that the above $(\mathcal{M}, \ )$ is a monotone span program which realizes Σ. The size of the MSP is the number of rows d in M.*

Theorem 1 ([3]) *The above algorithm constitutes a valid secret sharing scheme.*

2.1 Constructing Linear Codes from LSSS [18]

In [18], the authors have given the following construction to show that for every MSP, there exists a *linear code* [13]. We now recall this construction. Let $(\mathcal{M}, \ )$ be an MSP, where $\mathcal{M}$ is of size $d \times e$. Let k be the rank of $\mathcal{M}$, where $k \leq e$. Without loss of generality, let the first k rows of $\mathcal{M}$ be *linearly independent*. Notice that $\quad (1, \ldots, k) \notin \Gamma$. Otherwise, the adversary $\mathcal{A}$ recover all the n shares of any secret from the shares of that secret given to the corrupted parties.

Remark 1 (Regarding Rank of Rows of $\mathcal{M}$ Corresponding to an Access Set) *Note that it is not necessary that the rows in $\mathcal{M}$ assigned to the parties corresponding to an access set has full rank, namely k. This further implies that in an LSSS in non-threshold settings, the parties corresponding to an access set can only reconstruct the secret using their shares. But using these shares, they may or may not be able to reconstruct the shares of any other party. This is in contrast to threshold secret sharing schemes like Shamirs' [16], where the parties corresponding to any access*

set (namely any set of $t+1$ parties) can reconstruct back not only the secret, but also the shares of all other parties. Thus in an LSSS*, we cannot claim that share of every party can be expressed as a linear combination of the shares corresponding to the parties in an access set, until and unless the rows of $\mathcal{M}$ corresponding to the parties in the access set has full rank. It is this restriction, which will later let us to show that the concept of pseudobasis will not work in non-threshold settings.*

Definition 4 (Linear Codes) *[13, 18] A linear code C over $\mathbb{F}$ is defined by a $k \times d$ generating matrix G in standard form $G = [I_k|A]$, where I_k denotes $k \times k$ identity matrix and A is a $k \times (d-k)$ matrix. The codewords of C are determined by encoding function $EC : \mathbb{F}^k \rightarrow \mathbb{F}^d$, such that given a vector $\vec{r} \in \mathbb{F}^k$,*

$$EC(\vec{r}) = \vec{r} \cdot G = \vec{c},$$

where $\vec{c} \in \mathbb{F}^d$ is a codeword of C.

Linking LSSS with a Linear Code: Let $(\mathcal{M}_{d \times e}, \ )$ be an MSP, where the rank of $\mathcal{M}$ is k and where the first k rows of $\mathcal{M}$ are independent. Let $\mathcal{M}_k$ be the $k \times e$ submatrix of $\mathcal{M}$, consisting of first k rows of $\mathcal{M}$. So rank of $\mathcal{M}_k$ is k. Now we can construct a linear code C with generator matrix G from $\mathcal{M}$ as follows: G will be an $k \times d$ matrix. Let col_i be the ith column of G, for $i = 1, \ldots, d$. Then col_i has the following property:

$$(col_i)^T \times \mathcal{M}_k = row_i,$$

where row_i is the ith row of $\mathcal{M}$, for $i = 1, \ldots, d$.

The above construction is possible because $\mathcal{M}_k$ has rank k and thus every row_i of $\mathcal{M}$ is in the row span of $\mathcal{M}_k$. It is easy to see that $col_1, \ldots, col_k$ are nothing but the columns of I_k. Thus G generates a linear code, say C. Now one can easily verify that

$$\mathsf{LSSS}(s, \vec{\rho}) = (s_1, \ldots, s_k, \ldots, s_d) = EC(s_1, \ldots, s_k) \in C,$$

for any $s \in \mathbb{F}$ and $\vec{\rho} \in \mathbb{F}^{e-1}$. Therefore, the set of all possible shares generated by $\mathcal{M}$, namely $\{\mathsf{LSSS}(s, \vec{\rho}) | s \in \mathbb{F}, \vec{\rho} \in \mathbb{F}^{e-1}\}$, is a subset of the linear code C.

Now using the code C, the authors in [18] presented their constant round protocols. In the next section, we recall the notion of pseudobasis from [12] and show why it cannot be generalized in non-threshold settings.

3 Pseudobasis in Threshold Settings

We now recall the notion of pseudobasis from [12]. The idea was introduced in [12] to design a two round efficient and communication optimal PSMT protocol tolerating threshold adversary $\mathcal{A}_t$. The setting is as follows: $\mathbf{S}$ and $\mathbf{R}$ are connected by n bi-directional channels, denoted by $\mathcal{W} = \{w_1, \ldots, w_n\}$. Here $n \geq 2t+1$, which is a necessary and sufficient condition for the existence of any r-round PSMT protocol tolerating $\mathcal{A}_t$, where $r \geq 2$ [7]. For the current discussion, we fix $n = 2t+1$. Let $\mathcal{C}$ be the set of all possible Reed-Solomon (RS) codewords [13] of length n over $\mathbb{F}$, which are encoded using polynomials of degree t over $\mathbb{F}$. Moreover, let $\alpha_1, \ldots, \alpha_n$ be the publicly known elements from $\mathbb{F}$, at which the polynomials are evaluated to compute the RS codewords in $\mathcal{C}$. Now suppose $\mathbf{S}$ sends several codewords, say γ codewords $C_1, \ldots, C_\gamma \in \mathcal{C}$ to $\mathbf{R}$ over the n channels by transmitting i^{th} component of all the codewords over i^{th} channel w_i. During the transmission, $\mathcal{A}_t$ can control t channels and can change the component(s) of the codeword(s) sent over those channels. However, the locations at which error(s) occur in the codewords are not random. This is because for all the codewords, the errors always occur at the same t (or a smaller subset) locations. The concept of pseudo-basis is based on this simple and interesting observation.

Let $\mathbf{R}$ receive the set of vectors $\mathcal{Y} = \{Y_1 \ldots, Y_\gamma\}$ over the n wires, such that for $i = 1, \ldots, \gamma$,

$$Y_i = C_i + E_i, \tag{3}$$

where $E_i = (e_{i1}, \ldots, e_{in})$ is an error vector introduced by the adversary. Since at most t errors could occur during the transmission, each E_i may have at most t non-zero components. Let

$$support(E_i) = \{j \mid e_{ij} \neq 0\}. \tag{4}$$

Then there exists a set of t wires $\{w_{j_1}, \ldots, w_{j_t}\} \subset \mathcal{W}$ that are under the control of $\mathcal{A}_t$, such that each error vector E_i satisfies

$$support(E_i) \subseteq \{j_1, \ldots, j_t\} \tag{5}$$

This means that the space E spanned by the error vectors $E_1, \ldots, E_\gamma$ has dimension at most t. The notion of pseudo-basis exploits this idea. For two vectors $V_1, V_2 \in \mathbb{F}^n$, we write

$$V_1 = V_2 \; mod \, \mathcal{C} \tag{6}$$

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

$$Y_i = E_i \; mod \, \mathcal{C} \quad \text{holds, as } Y_i - E_i = C_i \in \mathcal{C}$$

Definition 5 (Linearly Pseudo-Express [12]) *We say that a vector $Y \in \mathcal{Y}$ is linearly pseudo-expressed by $\{B_1, \ldots, B_k\}$ if there exists some $\alpha = (a_1, \ldots, a_k)$, such that*

$$Y = a_1 B_1 + \ldots + a_k B_k \; mod \, \mathcal{C}$$

Lemma 1 ([12]) *Let $\mathcal{E} = \{E_1, \ldots, E_\gamma\}$ be the error vector set of $\mathcal{Y}$. Then $E_i \in \mathcal{E}$ is linearly expressed by $\{E_{j1}, \ldots, E_{jk}\} \subseteq \mathcal{E}$ iff Y_i is linearly pseudo-expressed by $\{Y_{j1}, \ldots, Y_{jk}\}$.*

The above lemma is very crucial and for the sake of completeness, we present the proof of the above lemma in **APPENDIX A**. *In the next section, we will show that if try to generalize pseudobasis (defined in the sequel) to non-threshold settings, then the above lemma need not hold. And it is due to this reason, pseudobasis will fail to achieve its properties in non-threshold settings.*

Definition 6 (Pseudo-Span [12]) *We say that $\{Y_{j1} \ldots, Y_{jk}\} \subset \mathcal{Y}$ pseudo-spans $\mathcal{Y}$ if each $Y_i \in \mathcal{Y}$ can be linearly pseudo-expressed by $\{Y_{j1} \ldots, Y_{jk}\}$. That is, each $Y_i = (a_1 Y_{j1} + \ldots + a_k Y_{jk}) \; mod \, \mathcal{C}$, for some $a_1, \ldots, a_k$.*

Definition 7 (Pseudo-Basis [12]) *We say that $\{Y_{j1} \ldots, Y_{jk}\} \subset \mathcal{Y}$ is a pseudo-basis of $\mathcal{Y}$ if it is the minimum subset of $\mathcal{Y}$ which pseudo-spans $\mathcal{Y}$.*

Definition 8 (Pseudo-Dimension [12]) *If $\{Y_{j1} \ldots, Y_{jk}\} \subset \mathcal{Y}$ is a pseudo-basis of $\mathcal{Y}$ and $k = |\{Y_{j1} \ldots, Y_{jk}\}|$, then we say that $\mathcal{Y}$ has pseudo-dimension k.*

Theorem 2 ([12]) *Let $\mathcal{E} = \{E_1, \ldots, E_\gamma\}$ be the error vector set of $\mathcal{Y}$. Then $\mathcal{B}_e = \{E_{j1}, \ldots, E_{jk}\} \subset \{E_1, \ldots, E_\gamma\}$ is a basis of $\mathcal{E}$ iff $\mathcal{B}_y = \{Y_{j1}, \ldots, Y_{jk}\} \subset \mathcal{Y}$ is a pseudo-basis of $\mathcal{Y}$ (Note that $\mathcal{B}_e$ and $\mathcal{B}_y$ have same indices). In particular, the pseudo-dimension of $\mathcal{Y}$ is equal to the dimension of E.*

Note that Lemma 1 is very crucial to prove the above theorem. Let $\mathcal{E} = \{E_1, \ldots, E_\gamma\}$ be the error vector set of $\mathcal{Y}$ and let $\{C_1, \ldots, C_\gamma\}$ be the corresponding original codewords which **S** sent. Then define

$$CORRUPTED = \cup_{i=1}^{\gamma} support(E_i) \tag{7}$$

Then $CORRUPTED$ is the set of wires that $\mathcal{A}_t$ has corrupted.

Theorem 3 ([12]) *Let $\mathcal{B}_y = \{Y_{j1}, \ldots, Y_{jk}\}$ be the pseudo-basis of $\mathcal{Y}$ and let $C_{j1}, \ldots, C_{jk}$ be the corresponding original codewords. Then $CORRUPTED = \cup_{i=1}^{i=k} support(Y_{ji} - C_{ji})$.*

Theorem 4 ([12]) *The pseudo-dimension of $\mathcal{Y}$ is at most t.*

How Pseudobasis can be used to design efficient PSMT protocols: From the above discussion, we find that if **R** can efficiently find the pseudobasis and correctly send it back to **S**, then **S** can find the identity of all corrupted wires. In [12], the authors have shown how both these tasks can be done efficiently. This along with several other steps, helps both **S** and **R** to identify all corrupted wires and by using this knowledge they can securely exchange the message.

4 Non-Applicability of Pseudobasis in Non-Threshold Settings

In [18] the authors claimed the following: consider a non-threshold adversary $\mathcal{A}$, characterized by an adversary structure Γ over $\mathcal{W}$, such that $\mathcal{A}$ satisfies $\mathcal{Q}^2$ condition with respect to $\mathcal{W}$ (this is a generalization of the condition $n \geq 2t+1$). Then the concept of pseudobasis (discussed in the previous section) can be generalized to non-threshold settings. Specifically, Lemma 1, Theorem 2 and Theorem 3 will also hold against $\mathcal{A}$. However, we now show that this need not be the case. We first discuss a counter example.

A Counter Example: Let $n = 5$ and consider a non-threshold adversary $\mathcal{A}$, specified by adversary structure $\Gamma = \{\{w_1\}, \{w_2\}, \{w_3\}, \{w_4, w_5\}, \{w_1, w_4\}, \{w_1, w_5\}, \{w_2, w_4\}, \{w_2, w_5\}, \{w_3, w_4\}, \{w_3, w_5\}\}$. It is easy to verify that $\mathcal{A}$ satisfies $\mathcal{Q}^2$ condition. The corresponding monotone access structure $\Sigma = \{\{w_1, w_2\}, \{w_1, w_3\}, \{w_2, w_3\}, \{w_1, w_4, w_5\}, \{w_2, w_4, w_5\}, \{w_3, w_4, w_5\}\}$. Now consider the following LSSS: to share a secret s, select a random polynomial $f(x) = s + ax$, where a is random. In addition, select random b, c and d, subject to the condition that $s = b + c + d$. Now the shares corresponding to $w_1, \ldots, w_5$ are $\texttt{share}_1, \ldots, \texttt{share}_5$ respectively, where $\texttt{share}_1 = (f(1), b) = (s+a, b)$, $\texttt{share}_2 = (f(2), b) = (s + 2a, b)$, $\texttt{share}_3 = (f(3), b) = (s + 3a, b)$, $\texttt{share}_4 = (c)$ and $\texttt{share}_5 = (d)$. It is easy to see that this sharing can be represented by following 8×5 matrix:

$$\mathcal{M} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 3 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Now as shown in Section 2, we can construct a linear code C from $\mathcal{M}$, where

$$C = \{(s + a, b, s + 2a, b, s + 3a, b, c, d)\}.$$

If $(s, a, b, c, d) = (0, 0, 0, 1, -1)$, then the corresponding codeword is $C_1 = (0, 0, 0, 0, 0, 0, 1, -1)$. On the other hand, if $(s, a, b, c, d) = (-1, 1, 0, -1, 1)$, then the corresponding codeword is $C_2 = (0, 0, 1, 0, 2, 0, -1, 1)$. Now suppose C_1, C_2 are transmitted over the channels, with $\texttt{share}_i$ being transmitted over channel w_i, for $i = 1, \ldots, 5$. Let $\mathcal{A}$ controls the set $\{w_4, w_5\}$ from Γ. Let $\mathbf{R}$ receive $Y_1 = (0, 0, 0, 0, 0, 0, 1, 0)$ and $Y_2 = (0, 0, 1, 0, 2, 0, 0, 1)$. So the corresponding error vectors are $E_1 = (0, 0, 0, 0, 0, 0, 0, 1)$ and $E_2 = (0, 0, 0, 0, 0, 0, 1, 0)$.

Now it is easy to see that $Y_2 - Y_1 = (0, 0, 1, 0, 2, 0, -1, 1) = C_2 \in C$. So Y_1 is the pseudobasis of $\mathcal{Y} = \{Y_1, Y_2\}$. So according to Lemma 1 and Theorem 2, E_1 is the basis of error vectors $\mathcal{E} = \{E_1, E_2\}$. However, this is not so as E_1 and E_2 are independent. So if $\mathbf{R}$ reliably sends back the pseudobasis Y_1 to $\mathbf{S}$, then from the corresponding error vector E_1, $\mathbf{S}$ will conclude that only w_5 is corrupted, which is incorrect because w_4 is also corrupted.

4.1 The Technical Reason for Non-Applicability of Pseudobasis

The mean reason why pseudobasis cannot be generalized in non-threshold settings is that Lemma 1 cannot be generalized against non-threshold adversary. Specifically, in non-threshold settings, if any $Y_i \in \mathcal{Y}$ is linearly pseudo-expressed by $\{Y_{j1}, \ldots, Y_{jk}\} \subset \mathcal{Y}$, then it is not true that the error vector E_i can be linearly expressed by error vectors $\{E_{j1}, \ldots, E_{jk}\}$. To see this, we recall the proof of this fact (in threshold settings) from **APPENDIX A** and analyze where the proof goes wrong in non-threshold settings.

So let Y_i be linearly pseudo-expressed by $\{Y_{j1}, \ldots, Y_{jk}\}$. Then in mod $\mathcal{C}$, we have

$$Y_i = (a_1 Y_{j1} + \ldots + a_k Y_{jk}) \; mod \; \mathcal{C} \tag{8}$$

for some $a_1, \ldots, a_k$. The above equation can be written as

$$(\mathcal{C}_i + E_i) = \{a_1(\mathcal{C}_{j1} + E_{j1}) + \ldots + a_k(\mathcal{C}_{jk} + E_{jk})\} \; mod \; \mathcal{C}$$
$$\Rightarrow (\mathcal{C}_i - a_1\mathcal{C}_{j1} - \ldots - a_k\mathcal{C}_{jk}) + (E_i - a_1E_{j1} - \ldots - a_kE_{jk}) \in \mathcal{C}$$

Now $\mathcal{C}_i, \mathcal{C}_{j1}, \ldots, \mathcal{C}_{jk}$ are valid *linear* codewords. So from linearity property, $(\mathcal{C}_i - a_1\mathcal{C}_{j1} - \ldots - a_k\mathcal{C}_{jk}) \in \mathcal{C}$. The linearity property also implies that: $(E_i - a_1E_{j1} - \ldots - a_kE_{jk}) \in \mathcal{C}$. In threshold settings (where each codeword is nothing but n distinct points on some degree-t polynomial), we could claim that since each error vector can have non-zero value in at most t locations, the codeword $(E_i - a_1E_{j1} - \ldots - a_kE_{jk})$ will have at least $n - t \geq t + 1$ zero components and hence $(E_i - a_1E_{j1} - \ldots - a_kE_{jk})$ is an all zero vector (0-codeword). This is because there is only one codeword, namely 0-codeword $(0, \ldots, 0)$, which has at least $t + 1$ zero components.

However, in non-threshold settings, we can claim only the following: each error vector $E_i, E_{j1}, \ldots, E_{jk}$ will have 0's in the locations corresponding to some access set, say Acc_1 and non-zero values in the locations corresponding to some forbidden set from Γ. This further implies that the codeword $(E_i - a_1E_{j1} - \ldots - a_kE_{jk})$ will have 0's in the locations corresponding to Acc_1. But from this, we cannot claim that $(E_i - a_1E_{j1} - \ldots - a_kE_{jk})$ is an all 0-codeword. This is because rows in the MSP $\mathcal{M}$, corresponding to Acc_1 may not have full rank (see Remark 1). We can only claim that the secret s, corresponding to which $(E_i - a_1E_{j1} - \ldots - a_kE_{jk})$ is the codeword, is zero.

In fact, in the counter example that we have presented in the previous section, we can verify that $E_2 - E_1 = (0, 0, 0, 0, 0, 0, 1, -1) \in C$. Specifically, $(0, 0, 0, 0, 0, 0, 1, -1)$ is the codeword corresponding to $(s, a, b, c, d) = (0, 0, 0, 1, -1)$. So though the codeword $(0, 0, 0, 0, 0, 0, 1, -1)$ has zero at all the locations corresponding to the access set $\{w_1, w_2, w_3\} \in \Sigma$, still $E_2 - E_1$ is a non-zero codeword. This is because the rows in $\mathcal{M}$, corresponding to w_1, w_2 and w_3 (namely the first six rows of $\mathcal{M}$) does not have full rank.

5 Three Round PSMT Sending Multiple Messages Concurrently

In[18], the authors have presented a three round efficient PSMT protocol (namely protocol **3-Round-Protocol-II**) tolerating $\mathcal{Q}^2$ adversary structure. As mentioned earlier, this protocol uses pseudobasis to identify the corrupted channels. But as discussed in the previous section, pseudobasis may not help to identify all corrupted channels in non-threshold settings. So we now a present a new and simple three round efficient PSMT protocol which can send multiple messages *concurrently*. Before that, we present few tools and notations used in our protocol.

5.1 Notations and Tools Used in Our Protocols

Let $\mathcal{A}$ be a non-threshold adversary, specified by an adversary structure Γ over the set of channels $\mathcal{W}$. Moreover, let $\Sigma = \Gamma^c$ be the corresponding access structure over $\mathcal{W}$. Furthermore, let $\mathcal{A}$ satisfies $\mathcal{Q}^2$ condition with respect to $\mathcal{W}$. Let $(\mathcal{M}, \;\;)$ be the MSP realizing the access structure Σ. *Without loss of generality and for simplicity, we assume that only i^{th} row of $\mathcal{M}$ is assigned to channel w_i, for $i = 1, \ldots, n$.* Thus,

$$\mathcal{M} = \begin{pmatrix} \vec{m_1} \\ \vdots \\ \vec{m_n} \end{pmatrix}$$

is an $n \times e$ matrix over $\mathbb{F}$. *However, our protocols will also work when more than one row of $\mathcal{M}$ is assigned to some w_i.* Finally we use the following notation in our protocols:

Notation 1 *Let $\mathcal{Q}$ be any subset of $\mathcal{W}$ i.e $\mathcal{Q} \subseteq \mathcal{W}$. Then $\mathcal{M}_\mathcal{Q}$ denotes the matrix containing the rows of $\mathcal{M}$ corresponding to the channels in $\mathcal{Q}$. For example, if $\mathcal{Q} = \{w_1, \ldots, w_t\}$, then*

$$\mathcal{M}_\mathcal{Q} = \begin{pmatrix} \vec{m_1} \\ \vdots \\ \vec{m_t} \end{pmatrix}.$$

Definition 9 (Size of $\mathcal{A}$ [18]) *The size of $\mathcal{A}$, denoted by $sz^{\mathcal{A}}$, is defined as*

$$sz^{\mathcal{A}} = \max\{\textit{size of } B | B \in \Gamma\},$$

where size of any $B \in \Gamma$ is defined as $|B|$. Thus, $sz^{\mathcal{A}}$ denotes the maximum number of channels that can be corrupted by the adversary. It is easy to see that $sz^{\mathcal{A}} = \mathcal{O}(n)$.

Definition 10 (Broadcasting a Value [11]) *Suppose* $\mathbf{S}$ *(*$\mathbf{R}$*) wants to reliably send some value x to* $\mathbf{R}$ *(*$\mathbf{S}$*) even in the presence of $\mathcal{A}$. Then* $\mathbf{S}$ *can do so by simply broadcasting x, which means he sends x over all the channels. In [11], Kurosawa has given an efficient algorithm, which allows* $\mathbf{R}$ *(*$\mathbf{S}$*) to recover x correctly. For details, see [11]. The communication complexity will be $n|x|$.*

Definition 11 (Extranding Randomness (EXTRAND) [17, 1]) *Suppose* $\mathbf{S}$ *and* $\mathbf{R}$ *have some how agreed on a sequence of N elements $X = [X_1, \ldots, X_N]$, such that $N - F$ elements of X are known to the adversary and the remaining F elements are information theoretically secure, where $F < N$. However, neither* $\mathbf{S}$ *nor* $\mathbf{R}$ *knows which elements are information theoretically secure. Now both* $\mathbf{S}$ *and* $\mathbf{R}$ *wants to locally compute a sequence of F values, namely $Y = [Y_1, \ldots, Y_F]$, from X, such that the elements of Y are information theoretically secure. In [17], the authors have presented an efficient algorithm, called $EXTRAND$ to solve this problem. We use this algorithm as $Y = EXTRAND_{N,F}(X)$.*

5.2 Our Three Round Protocol

We now present our three round efficient protocol called 3-Round-Multiple, which can *concurrently* send a message $m^{\mathbf{S}}$ containing $(n - sz^{\mathcal{A}})$ field elements. We first explain the idea of the protocol.

The Underlying Idea: During first round, $\mathbf{S}$ selects n random secrets and generates n shares for each of these secrets according to the LSSS $\mathcal{M}$. Now i^{th} secret, along with the randomness used with it to compute the n shares for the ith secret is sent over channel w_i. In addition, j^{th} share of each of the n secrets is sent over w_j. After receiving the values, $\mathbf{R}$ will try to find out all possible inconsistencies in the receive information (how this is done is shown in the protocol). Then during second round, $\mathbf{R}$ will broadcast all inconsistencies to $\mathbf{S}$, which will be correctly received by $\mathbf{S}$. Now after comparing the inconsistent values (as received by $\mathbf{R}$), with the original values which $\mathbf{S}$ has sent, $\mathbf{S}$ will identify all secrets which were incorrectly received by $\mathbf{R}$ during the first round. Now neglecting those secrets, $\mathbf{S}$ will consider all the remaining secrets which he knows has been correctly received by $\mathbf{R}$. Now $\mathbf{S}$ knows that there exists at least $(n - sz^{\mathcal{A}})$ secrets among those retained secrets, which were sent over the wires not under the control of adversary and hence are information theoretically secure. However, $\mathbf{S}$ does not know their exact identity. So $\mathbf{S}$ applies EXTRAND on the retained secrets to get a random and information theoretically secure one time pad consisting of $(n - sz^{\mathcal{A}})$ elements. Now using the pad, $\mathbf{S}$ masks elements of $m^{\mathbf{S}}$. Finally, during the third round, $\mathbf{S}$ broadcasts the masked message, along with the indexes of the secrets which are used by $\mathbf{S}$ to generate the pad. Now using this information, $\mathbf{R}$ will also be able to correctly generate the pad and hence can get back $m^{\mathbf{S}}$ by unmasking the masked message with the pad.

A similar idea as above, is used in the three round protocol of [14] in threshold settings. So our protocol can be viewed as a generalization of the protocol of [14]. The protocol is given in Fig. 1.

We now prove the properties of the protocol.

Lemma 2 (Security) *In protocol of Fig. 1, message $m^{\mathbf{S}}$ will be information theoretically secure.*

PROOF: To prove the lemma, we have to show that the pad $Y^{\mathbf{S}}$ will be information theoretically secure. This can be proved by showing that adversary will have no information about $s_i^{\mathbf{S}}$'s which are sent over the channels not under his control. The rest will follow from the properties of EXTRAND algorithm and the fact that there are at least $n - sz^{\mathcal{A}}$ such channels. So let w_i be a channel not under the control of $\mathcal{A}$. During **Round I**, $\mathcal{A}$ will get at most $sz^{\mathcal{A}}$ shares of $s_i^{\mathbf{S}}$, which are sent over the channels under his control. However, from the secrecy property of LSSS, these shares will not reveal any information about $s_i^{\mathbf{S}}$ to $\mathcal{A}$. During **Round II**, $\mathcal{A}$ will know the list $Conflict$ as

Figure 1: Efficient Three Round Protocol Sending a Message of Size $(n - sz^{\mathcal{A}})$ Field Elements

Round I: S to R:

1. **S** selects n random elements $s_1^{\mathbf{S}}, \ldots, s_n^{\mathbf{S}}$ from $\mathbb{F}$ and computes $\mathsf{LSSS}(s_i^{\mathbf{S}}, \vec{\rho_i}^{\mathbf{S}}) = (\mathtt{share}_{i1}^{\mathbf{S}}, \cdots, \mathtt{share}_{in}^{\mathbf{S}})$, for $i = 1, \ldots, n$, where $\vec{\rho_i}^{\mathbf{S}}$'s are the randomness.
2. For $i = 1, \ldots, n$, **S** sends the following to **R** over channel w_i:

$$\text{(a) } (s_i^{\mathbf{S}}, \vec{\rho_i}^{\mathbf{S}}) \quad \text{and} \quad \text{(b) } \mathtt{share}_{ji}^{\mathbf{S}}, \text{ for } j = 1, \ldots, n.$$

Round II: R to S:

1. For $i = 1, \ldots, n$, let **R** receive the following from **S** over channel w_i [a]

$$\text{(a) } (s_i^{\mathbf{R}}, \vec{\rho_i}^{\mathbf{R}}) \quad \text{and} \quad \text{(b) } \mathtt{share}_{ji}^{\mathbf{R}}, \text{ for } j = 1, \ldots, n.$$

2. For $i = 1, \ldots, n$, **R** computes $\mathsf{LSSS}(s_i^{\mathbf{R}}, \vec{\rho_i}^{\mathbf{R}}) = (\mathtt{share}'^{\mathbf{R}}_{i1}, \cdots, \mathtt{share}'^{\mathbf{R}}_{in})$.
3. For $i = 1, \ldots, n$, **R** checks $\mathtt{share}'^{\mathbf{R}}_{ij} \stackrel{?}{=} \mathtt{share}_{ij}^{\mathbf{R}}$, for $j = 1, \ldots, n$. If the test fails then **R** adds a conflicting tuple $(i, j, \mathtt{share}'^{\mathbf{R}}_{ij}, \mathtt{share}_{ij}^{\mathbf{R}})$ to a list $Conflict$, which is initially $\emptyset$.
4. Finally, **R** broadcasts the list $Conflict$, containing all conflicting values to **S**.

Round III: S to R:

1. **S** correctly receives the list $Conflict$. **S** initializes a list called $Faulty = \emptyset$.
2. For each $(i, j, \mathtt{share}'^{\mathbf{R}}_{ij}, \mathtt{share}_{ij}^{\mathbf{R}}) \in Conflict$, **S** does the following:
   (a) **S** checks $\mathtt{share}'^{\mathbf{R}}_{ij} \stackrel{?}{=} \mathtt{share}_{ij}^{\mathbf{S}}$. If the test fails then **S** concludes that **R** has received $s_i^{\mathbf{R}} \neq s_i^{\mathbf{S}}$ over w_i and adds w_i to list $Faulty$.
   (b) **S** checks $\mathtt{share}_{ij}^{\mathbf{R}} \stackrel{?}{=} \mathtt{share}_{ij}^{\mathbf{S}}$. If the test fails then **S** concludes that **R** has received $\mathtt{share}_{ij}^{\mathbf{R}} \neq \mathtt{share}_{ij}^{\mathbf{S}}$ over w_j and adds w_j to list $Faulty$.
3. **S** constructs a list $X^{\mathbf{S}}$, which is the collection of all $s_i^{\mathbf{S}}$'s, such that $w_i \notin Faulty$. **S** then computes $Y^{\mathbf{S}} = EXTRAND_{|X^{\mathbf{S}}|,(n-sz^{\mathcal{A}})}(X^{\mathbf{S}})$.
4. Let $Y^{\mathbf{S}} = (y_1^{\mathbf{S}}, \ldots, y_\ell^{\mathbf{S}})$ and the message $m^{\mathbf{S}} = (m_1^{\mathbf{S}}, \ldots, m_\ell^{\mathbf{S}})$, where $\ell = (n - sz^{\mathcal{A}})$. **S** computes $C^{\mathbf{S}} = (c_1^{\mathbf{S}}, \ldots, c_\ell^{\mathbf{S}}) = m^{\mathbf{S}} \oplus Y^{\mathbf{S}}$, where $c_l^{\mathbf{S}} = y_l^{\mathbf{S}} \oplus m_l^{\mathbf{S}}$, for $l = 1, \ldots, \ell$.
5. Finally **S** broadcasts $C^{\mathbf{S}}$ and the list $Faulty$ to **R** and terminates the protocol.

Message Recovery by R:

1. **R** correctly receives $C^{\mathbf{R}} = C^{\mathbf{S}}$ and the list $Faulty$.
2. **R** constructs a list $X^{\mathbf{R}}$, which is the collection of all $s_i^{\mathbf{R}}$'s, such that $w_i \notin Faulty$. **R** then computes $Y^{\mathbf{R}} = EXTRAND_{|X^{\mathbf{R}}|,(n-sz^{\mathcal{A}})}(X^{\mathbf{R}})$.
3. Finally **R** computes $m^{\mathbf{R}} = C^{\mathbf{R}} \oplus Y^{\mathbf{R}}$ and terminates the protocol.

[a] If channel w_i is not under the control of $\mathcal{A}$ then $(s_i^{\mathbf{R}}, \vec{\rho_i}^{\mathbf{R}}) = (s_i^{\mathbf{S}}, \vec{\rho_i}^{\mathbf{S}})$ and $\mathtt{share}_{ji}^{\mathbf{R}} = \mathtt{share}_{ji}^{\mathbf{S}}$, for $j = 1, \ldots, n$.

it is broadcasted. Now suppose there exists a tuple $(i, j, \mathtt{share}'^{\mathbf{R}}_{ij}, \mathtt{share}_{ij}^{\mathbf{R}}) \in Conflict$, involving w_i. We claim that in this case, w_j is corrupted (i.e., under the control of adversary) and hence adversary already knows the correct share $\mathtt{share}_{ij}^{\mathbf{S}} = \mathtt{share}'^{\mathbf{R}}_{ij} \neq \mathtt{share}_{ij}^{\mathbf{R}}$. This is because no two honest channels (which are not under the control of the adversary) will conflict with each other. So knowing $Conflict$ does not add any new information about $s_i^{\mathbf{S}}$ to the view of the adversary. Finally, during **Round III**, **S** broadcasts $C^{\mathbf{S}}$ and the list $Faulty$, which does not add new information about $s_i^{\mathbf{S}}$ to the view of the adversary. Hence $s_i^{\mathbf{S}}$ will be information theoretically secure. □

Lemma 3 (Reliability) *In protocol of Fig. 1,* **R** *will correctly recover* $m^{\mathbf{R}} = m^{\mathbf{S}}$.

PROOF: To prove the lemma, we have to show that $X^{\mathbf{R}} = X^{\mathbf{S}}$. The rest will follow from the properties of EXTRAND. To show that $X^{\mathbf{R}} = X^{\mathbf{S}}$, we will show that if w_i is a *corrupted* channel, under the control of the adversary, such that $s_i^{\mathbf{R}} \neq s_i^{\mathbf{S}}$, then $w_i \in Faulty$. So let w_i be a corrupted channel, such that $s_i^{\mathbf{R}} \neq s_i^{\mathbf{S}}$. We now claim that there exists at least one *honest channel* (not under the control of $\mathcal{A}$) say w_j, such that $\mathtt{share}_{ij}^{\mathbf{R}} = \mathtt{share}_{ij}^{\mathbf{S}}$ and $\mathtt{share}'^{\mathbf{R}}_{ij} \neq \mathtt{share}_{ij}^{\mathbf{R}}$ and hence the tuple $(i, j, \mathtt{share}'^{\mathbf{R}}_{ij}, \mathtt{share}_{ij}^{\mathbf{R}}) \in Conflict$. Now since **R** broadcasts $Conflict$ to **S**, it will be received

correctly by $\mathbf{S}$, who will come to know that $s_i^{\mathbf{R}} \neq s_i^{\mathbf{S}}$, as the test $\mathtt{share}'^{\mathbf{R}}_{ij} \stackrel{?}{=} \mathtt{share}^{\mathbf{S}}_{ij}$ will fail.

So we now proceed to prove the above claim by contradiction. Let $v_i^{\mathbf{S}} = (s_i^{\mathbf{S}}, \vec{\rho}_i^{\mathbf{S}})$ and $v_i^{\mathbf{R}} = (s_i^{\mathbf{R}}, \vec{\rho}_i^{\mathbf{R}})$. Since $s_i^{\mathbf{S}} \neq s_i^{\mathbf{R}}$, we have $v_i^{\mathbf{R}} \neq v_i^{\mathbf{S}}$. Without loss of generality, let $w_1, \ldots, w_t$ be the honest channels and hence $\{w_1, \ldots, w_t\}$ is an *access set*. Let $\mathcal{M}_{Ho}$ be the rows in the MSP $\mathcal{M}$ corresponding to $w_1, \ldots, w_t$. Thus, M_{Ho} consists of first t rows of $\mathcal{M}$. Now we know the following:

$$(\mathtt{share}^{\mathbf{R}}_{i1}, \ldots, \mathtt{share}^{\mathbf{R}}_{it})^T = (\mathtt{share}^{\mathbf{S}}_{i1}, \ldots, \mathtt{share}^{\mathbf{S}}_{it})^T \tag{9}$$

This is because $w_1, \ldots, w_t$ are honest channels. Also

$$(\mathtt{share}^{\mathbf{R}}_{i1}, \ldots, \mathtt{share}^{\mathbf{R}}_{it})^T = \mathcal{M}_{Ho} \cdot (v_i^{\mathbf{S}})^T \tag{10}$$

Now the following holds:

$$(\mathtt{share}'^{\mathbf{R}}_{i1}, \ldots, \mathtt{share}'^{\mathbf{R}}_{it})^T = \mathcal{M}_{Ho} \cdot (v_i^{\mathbf{R}})^T \tag{11}$$

If $(\mathtt{share}'^{\mathbf{R}}_{i1}, \ldots, \mathtt{share}'^{\mathbf{R}}_{it})^T = (\mathtt{share}^{\mathbf{R}}_{i1}, \ldots, \mathtt{share}^{\mathbf{R}}_{it})^T$ holds then it implies the following

$$(\mathtt{share}'^{\mathbf{R}}_{i1}, \ldots, \mathtt{share}'^{\mathbf{R}}_{it})^T - (\mathtt{share}^{\mathbf{R}}_{i1}, \ldots, \mathtt{share}^{\mathbf{R}}_{it})^T = (0, \ldots, 0(t \text{ times}))^T = \mathcal{M}_{Ho} \cdot ((v_i^{\mathbf{R}})^T - (v_i^{\mathbf{S}})^T)$$

Since $\{w_1, \ldots, w_t\}$ is an *access set*, there exists a *recombination vector*, say $\vec{\alpha}_{Ho}$, such that $\vec{\alpha}_{Ho} \cdot \mathcal{M}_{Ho} = (1, 0, \ldots, 0)$. So multiplying both sides of the above equation by $\vec{\alpha}_{Ho}$, we get

$$\vec{\alpha}_{Ho} \cdot (0, \ldots, 0(t \text{ times}))^T = \vec{\alpha}_{Ho} \cdot \mathcal{M}_{Ho} \cdot ((v_i^{\mathbf{R}})^T - (v_i^{\mathbf{S}})^T)$$

The above equation further implies that

$$(0, \ldots, 0(t \text{ times}))^T = (1, 0, \ldots, 0) \cdot ((v_i^{\mathbf{R}})^T - (v_i^{\mathbf{S}})^T) = (s_i^{\mathbf{R}} - s_i^{\mathbf{S}}, 0, \ldots, 0)^T$$

Thus, we get $s_i^{\mathbf{R}} - s_i^{\mathbf{S}} = 0$, which implies that $s_i^{\mathbf{R}} = s_i^{\mathbf{S}}$, which is a contradiction. □

Theorem 5 *The protocol in Fig. 1 is an efficient three round PSMT protocol, which sends a message containing $(n - sz^{\mathcal{A}})$ field elements by communicating $\mathcal{O}(n^3)$ field elements.*

Proof: The fact that the protocol is a PSMT protocol follows from Lemma 3 and Lemma 2. During **Round I**, $\mathbf{S}$ sends $\mathcal{O}(n^2)$ field elements. During **Round II**, $\mathbf{R}$ broadcasts $Conflict$, which requires communicating $\mathcal{O}(n^3)$ field elements. During **Round III**, $\mathbf{S}$ broadcasts $C^{\mathbf{S}}$ and the list $Faulty$, which requires communicating $\mathcal{O}(n^3)$ field elements. So total communication complexity is $\mathcal{O}(n^3)$. Now it is easy to verify that the computation and communication complexity of the protocol is polynomial in size of Γ, n and the size of underlying LSSS. □

6 A Simple Two Round Protocol Sending Single Message

We now present a simple and efficient two round protocol which sends a message containing *one* field element and involves a communication complexity of $\mathcal{O}(n^3)$. If we compare this protocol with the three round protocol of previous section, then we find that the number of rounds now decreases by one. However, on the negative size, our two round protocol can send *only one* field element, while our three round protocol can send *multiple field* elements *concurrently*, with the same communication complexity ($\mathcal{O}(n^3)$). In [18], a two round efficient protocol is presented, having the same communication complexity as ours. However, the protocol of [18] uses the idea of converting the underlying LSSS into linear code (as explained in Sec. 2.1) and uses the properties of encoding and decoding of the linear code. On the other hand, we directly use the underlying LSSS and hence our protocol is conceptually simpler. We now present the underlying idea of our protocol.

Underlying Idea: The first round of the protocol is same as in our three round protocol, except that first round is now executed by $\mathbf{R}$, instead of $\mathbf{S}$. Thus $\mathbf{R}$ sends n random secrets and their shares (generated according to the LSSS) across the n channels. $\mathbf{S}$ on receiving the secrets and the shares,

finds the inconsistency among the received values and reliably sends back these inconsistencies to $\mathbf{R}$, in the form of a conflict list. $\mathbf{S}$ is sure that on receiving the conflict list, $\mathbf{R}$ can find out all the secrets which are received incorrectly by $\mathbf{S}$. Sender $\mathbf{S}$ also knows that the honest channels have correctly delivered the secrets and these secrets will be information theoretically secure. Now to securely communicate the message $m^{\mathbf{S}}$, $\mathbf{S}$ computes the shares of $m^{\mathbf{S}}$ according to the LSSS and mask each individual i^{th} share of $m^{\mathbf{S}}$ with the ith secret received by $\mathbf{S}$ and broadcasts the masked shares to $\mathbf{R}$. To recover the message, $\mathbf{R}$ first find out all the secrets which were received incorrectly by $\mathbf{S}$ from the list $Conflict$. $\mathbf{R}$ neglects all those secrets and the corresponding masked shares of $m^{\mathbf{S}}$. The remaining masked shares of $m^{\mathbf{S}}$ are then unmasked by $\mathbf{R}$ and $\mathbf{R}$ knows that they are the correct shares of $m^{\mathbf{S}}$. Moreover these shares correspond to an access set. So by applying the reconstruction algorithm of the LSSS to these shares, $\mathbf{R}$ recovers the message. A similar approach is used in two round protocol of [4] tolerating threshold adversary. Thus our protocol can be viewed as a generalization of the protocol of [4]. The protocol is now formally given in Fig. 2.

Figure 2: Simple and Efficient Two Round Protocol Sending a Single Field Element

Round I: R to S:

1. $\mathbf{R}$ selects n random elements $s_1^{\mathbf{R}}, \ldots, s_n^{\mathbf{R}}$ from $\mathbb{F}$ and computes $\mathsf{LSSS}(s_i^{\mathbf{R}}, \vec{\rho_i}^{\mathbf{R}}) = (\mathtt{share}_{i1}^{\mathbf{R}}, \cdots, \mathtt{share}_{in}^{\mathbf{R}})$, for $i = 1, \ldots, n$, where $\vec{\rho_i}^{\mathbf{R}}$'s are the randomness.
2. For $i = 1, \ldots, n$, $\mathbf{R}$ sends the following to $\mathbf{S}$ over channel w_i:

$$\text{(a) } (s_i^{\mathbf{R}}, \vec{\rho_i}^{\mathbf{R}}) \quad \text{and} \quad \text{(b) } \mathtt{share}_{ji}^{\mathbf{R}}, \text{ for } j = 1, \ldots, n.$$

Round II: S to R:

1. For $i = 1, \ldots, n$, let $\mathbf{S}$ receive the following from $\mathbf{R}$ over channel w_i:

$$\text{(a) } (s_i^{\mathbf{S}}, \vec{\rho_i}^{\mathbf{S}}) \quad \text{and} \quad \text{(b) } \mathtt{share}_{ji}^{\mathbf{S}}, \text{ for } j = 1, \ldots, n.$$

2. For $i = 1, \ldots, n$, $\mathbf{S}$ computes $\mathsf{LSSS}(s_i^{\mathbf{S}}, \vec{\rho_i}^{\mathbf{S}}) = (\mathtt{share}'^{\mathbf{S}}_{i1}, \cdots, \mathtt{share}'^{\mathbf{S}}_{in})$.
3. For $i = 1, \ldots, n$, $\mathbf{S}$ checks $\mathtt{share}'^{\mathbf{S}}_{ij} \stackrel{?}{=} \mathtt{share}_{ij}^{\mathbf{S}}$, for $j = 1, \ldots, n$. If the test fails then $\mathbf{S}$ adds a conflicting tuple $(i, j, \mathtt{share}'^{\mathbf{S}}_{ij}, \mathtt{share}_{ij}^{\mathbf{S}})$ to a list $Conflict$, which is initially $\emptyset$.
4. $\mathbf{S}$ computes $\mathsf{LSSS}(m^{\mathbf{S}}, \vec{\rho}^{\mathbf{S}}) = (\mathtt{share}_1^{\mathbf{S}}, \cdots, \mathtt{share}_n^{\mathbf{S}})$, where $\vec{\rho}^{\mathbf{S}} \in \mathbb{F}^{e-1}$ is a random vector. $\mathbf{S}$ also computes $C^{\mathbf{S}} = (c_1^{\mathbf{S}}, \ldots, c_n^{\mathbf{S}})$, where $c_i^{\mathbf{S}} = \mathtt{share}_i^{\mathbf{S}} \oplus s_i^{\mathbf{S}}$, for $i = 1, \ldots, n$.
5. Finally, $\mathbf{S}$ broadcasts the list $Conflict$ and $C^{\mathbf{S}}$ to $\mathbf{R}$ and terminates the protocol.

Message Recovery by R:

1. $\mathbf{R}$ correctly receives the list $Conflict$ and $C^{\mathbf{R}} = C^{\mathbf{S}}$. $\mathbf{R}$ initializes a list called $Faulty = \emptyset$.
2. For each $(i, j, \mathtt{share}'^{\mathbf{S}}_{ij}, \mathtt{share}_{ij}^{\mathbf{S}}) \in Conflict$, $\mathbf{R}$ does the following:
   (a) $\mathbf{R}$ checks $\mathtt{share}'^{\mathbf{S}}_{ij} \stackrel{?}{=} \mathtt{share}_{ij}^{\mathbf{R}}$. If the test fails then $\mathbf{R}$ concludes that $\mathbf{S}$ has received $s_i^{\mathbf{S}} \neq s_i^{\mathbf{R}}$ over w_i and adds w_i to list $Faulty$.
   (b) $\mathbf{R}$ checks $\mathtt{share}_{ij}^{\mathbf{S}} \stackrel{?}{=} \mathtt{share}_{ij}^{\mathbf{R}}$. If the test fails then $\mathbf{R}$ concludes that $\mathbf{S}$ has received $\mathtt{share}_{ij}^{\mathbf{S}} \neq \mathtt{share}_{ij}^{\mathbf{R}}$ over w_j and adds w_j to list $Faulty$.
3. $\mathbf{R}$ neglects all $c_i^{\mathbf{S}}$'s from $C^{\mathbf{S}}$, such that $w_i \in Faulty$. Using the remaining $c_i^{\mathbf{S}}$'s, such that $w_i \notin Faulty$, $\mathbf{R}$ computes $\mathtt{share}_i^{\mathbf{R}} = c_i^{\mathbf{S}} \oplus s_i^{\mathbf{R}}$.
4. Finally, $\mathbf{R}$ applies reconstruction algorithm of the LSSS to the above computes $\mathtt{share}_i^{\mathbf{R}}$'s, recovers $m^{\mathbf{R}}$ and terminates the protocol.

We now briefly sketch the properties of the protocol.

Lemma 4 (Secrecy) *In the protocol of Fig. 2, $m^{\mathbf{S}}$ will be information theoretically secure.*

Proof: We prove the claim for the worst case, when adversary passively controls (i.e., only listen the communication over the channels without modifying them) the channels under his control. In this case, $\mathcal{A}$ will know all the secrets $s_i^{\mathbf{R}}$'s which are sent over the channels under his control and hence $\mathtt{share}_i^{\mathbf{S}}$'s corresponding to those channels (by computing $\mathtt{share}_i^{\mathbf{S}} = c_i^{\mathbf{S}} \oplus s_i^{\mathbf{R}}$). However, the remaining $s_i^{\mathbf{R}}$'s, which are sent over the remaining channels (not under the control of $\mathcal{A}$) will remain information theoretically secure (follows using same argument as in Lemma 2). So even though $\mathcal{A}$

will know the $c_i^{\mathbf{S}}$'s corresponding to those *honest* channels (as they are broadcasted), he will have no information about the corresponding $\mathtt{share}_i^{\mathbf{S}}$'s (as the corresponding $s_i^{\mathbf{R}}$'s are unknown) and hence $m^{\mathbf{S}}$, as the channels under the control of $\mathcal{A}$ does not constitute an access set. □

Lemma 5 (Secrecy) *In the protocol of Fig. 2,* $\mathbf{R}$ *will correctly recover* $m^{\mathbf{R}} = m^{\mathbf{S}}$.

Proof: We first claim that if adversary changes some $s_i^{\mathbf{R}}$ sent over a channel under his control such that $s_i^{\mathbf{S}} \neq s_i^{\mathbf{R}}$, then $w_i \in Faulty$. This follows using same argument as used in the proof of Lemma 3. Also the channels w_i's which are not under the control of the adversary will constitute an access set. Now it is easy to see that $\mathtt{share}_i^{\mathbf{R}} = \mathtt{share}_i^{\mathbf{S}}$ will hold for each $w_i \notin Faulty$, as $s_i^{\mathbf{S}} = s_i^{\mathbf{R}}$ will hold for those w_i's. Thus by applying reconstruction algorithm of LSSS to $\mathtt{share}_i^{\mathbf{R}}$'s, where $w_i \notin Faulty$, $\mathbf{R}$ will correctly recover $m^{\mathbf{R}} = m^{\mathbf{S}}$, as these $\mathtt{share}_i^{\mathbf{R}}$'s form an access set. □

Theorem 6 *The protocol in Fig. 2 is an efficient two round PSMT protocol, which sends a message containing single field element by communicating* $\mathcal{O}(n^2)$ *field elements.*

Proof: Follows from Lemma 5, Lemma 4 and the description of the protocol. □

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] A. Choudhury. Protocols for reliable and secure message transmission. Cryptology ePrint Archive, Report 2010/281, 2010.

[3] R. Cramer, I. Damgård, and U. M. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 316–334, 2000.

[4] Y. Desmedt, S. Erotokritou, and R. Safavi-Naini. Simple and communication complexity efficient almost secure and perfectly secure message transmission schemes. In D. J. Bernstein and T. Lange, editors, *Progress in Cryptology - AFRICACRYPT 2010, Third International Conference on Cryptology in Africa, Stellenbosch, South Africa, May 3-6, 2010. Proceedings*, volume 6055 of *Lecture Notes in Computer Science*, pages 166–183. Springer, 2010.

[5] 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.

[6] Y. Desmedt, Y. Wang, and M. Burmester. A complete characterization of tolerable adversary structures for secure point-to-point transmissions without feedback. In *Algorithms and Computation, 16th International Symposium, ISAAC 2005, Sanya, Hainan, China, December 19-21, 2005, Proceedings*, volume 3827 of *Lecture Notes in Computer Science*, pages 277–287. Springer Verlag, 2005.

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

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

[9] M. Hirt and U. M. Maurer. Complete Characterization of Adversaries Tolerable in Secure Multi-Party Computation. In *Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing, Santa Barbara, California, USA, August 21-24, 1997*, pages 25–34. ACM Press, 1997.

[10] 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.

[11] K. Kurosawa. Round-efficient perfectly secure message transmission scheme against general adversary. Cryptology ePrint Archive, Report 2010/450, 2010.

[12] 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.

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

[14] 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.

[15] H. Sayeed and H. Abu-Amara. Efficient perfectly secure message transmission in synchronous networks. *Information and Computation*, 126(1):53–61, 1996.

[16] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[17] 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 CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 545–561. Springer, 2004.

[18] Q. Yang and Y. Desmedt. General perfectly secure message transmission using linear codes. In *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 448–465, 2010.

APPENDIX A: Proof of Properties of Pseudobasis in Threshold Settings

Lemma 1 [12]: *Let $\mathcal{E} = \{E_1, \ldots, E_\gamma\}$ be the error vector set of $\mathcal{Y}$. Then $E_i \in \mathcal{E}$ is linearly expressed by $\{E_{j1}, \ldots, E_{jk}\} \subseteq \mathcal{E}$ iff Y_i is linearly pseudo-expressed by $\{Y_{j1}, \ldots, Y_{jk}\}$.*

Proof: Let E_i be linearly expressed by $\{E_{j1}, \ldots, E_{jk}\}$. This implies that

$$E_i = a_1 E_{j1} + \ldots + a_k E_{jk}$$

for some $a_1, \ldots, a_k$. Since $\{E_1, \ldots, E_\gamma\}$ is error vector set of $\mathcal{Y}$, it implies that for each i, $Y_i = \mathcal{C}_i + E_i$.

Then in mod $\mathcal{C}$, we have:

$$\begin{aligned} & Y_i - (a_1 Y_{j1} + \ldots + a_k Y_{jk}) \\ &= (\mathcal{C}_i + E_i) - a_1(\mathcal{C}_{j1} + E_{j1}) - \ldots - a_k(\mathcal{C}_{jk} + E_{jk}) \\ &= (\mathcal{C}_i - a_1\mathcal{C}_{j1} - \ldots - a_k\mathcal{C}_{jk}) + (E_i - a_1 E_{j1} - \ldots - a_k E_{jk}) \\ &= (\mathcal{C}_i - a_1\mathcal{C}_{j1} - \ldots - a_k\mathcal{C}_{jk}) + 0 \\ &= (\mathcal{C}_i - a_1\mathcal{C}_{j1} - \ldots - a_k\mathcal{C}_{jk}) \end{aligned}$$

Now notice that $\mathcal{C}_i, \mathcal{C}_{j1}, \ldots, \mathcal{C}_{jk}$ are valid RS codewords belonging to $\mathcal{C}$ and hence are encoded using polynomials of degree t. So from the linearity property of RS codewords, $\mathcal{C}_i - a_1\mathcal{C}_{j1} - \ldots - a_k\mathcal{C}_{jk}$ will also be a valid RS codeword belonging to $\mathcal{C}$ and hence is encoded using polynomial of degree t. So $\mathcal{C}_i - a_1\mathcal{C}_{j1} - \ldots - a_k\mathcal{C}_{jk} \in \mathcal{C}$. Therefore, $Y_i - (a_1 Y_{j1} + \ldots + a_k Y_{jk}) \in \mathcal{C}$. Thus if E_i is linearly expressed by $\{E_{j1}, \ldots, E_{jk}\}$ then Y_i is linearly pseudo-expressed by $\{Y_{j1}, \ldots, Y_{jk}\}$.

Next suppose that Y_i is linearly pseudo-expressed by $\{Y_{j1}, \ldots, Y_{jk}\}$. Then in mod $\mathcal{C}$, we have

$$Y_i = (a_1 Y_{j1} + \ldots + a_k Y_{jk}) \; mod \; \mathcal{C} \tag{12}$$

for some $a_1, \ldots, a_k$. The above equation can be written as

$$\begin{aligned} & (\mathcal{C}_i + E_i) = \{a_1(\mathcal{C}_{j1} + E_{j1}) + \ldots + a_k(\mathcal{C}_{jk} + E_{jk})\} \; mod \; \mathcal{C} \\ \Rightarrow & (\mathcal{C}_i - a_1\mathcal{C}_{j1} - \ldots - a_k\mathcal{C}_{jk}) + (E_i - a_1 E_{j1} - \ldots - a_k E_{jk}) \in \mathcal{C} \end{aligned}$$

Now $\mathcal{C}_i, \mathcal{C}_{j1}, \ldots, \mathcal{C}_{jk}$ are valid RS codewords, encoded using polynomials of degree t. So from the linearity property of RS codes, we have

$$(\mathcal{C}_i - a_1\mathcal{C}_{j1} - \ldots - a_k\mathcal{C}_{jk}) \in \mathcal{C} \tag{13}$$

The linearity property of RS codes also implies:

$$(E_i - a_1 E_{j1} - \ldots - a_k E_{jk}) \in \mathcal{C}$$

as the sum of a valid RS codeword (encoded using polynomial of degree t) with another vector can be a valid RS codeword (encoded using polynomial of degree t) only if the other vector is also a valid RS codeword (encoded using polynomial of degree t). As the number of non-zero components in each $E_i, E_{j1}, \ldots, E_{jk}$ is at most t, the vector $(E_i - a_1 E_{j1} - \ldots - a_k E_{jk})$ will have at least $n - t \geq t + 1$ zero components. However, in $\mathcal{C}$, there is only one codeword, namely 0-codeword (i.e. a tuple of length n containing all 0's) $(0, \ldots, 0)$, which has at least $t + 1$ zero components. This is because each element of $\mathcal{C}$ represents n distinct points on a t degree polynomial and $t + 1$ zero's uniquely define the zero polynomial $f(x) = \sum_{i=0}^{t} 0x^i$. These two facts together imply that the vector $(E_i - a_1 E_{j1} - \ldots - a_k E_{jk})$ is an all zero vector (0-codeword) which further implies that

$$E_i = a_1 E_{j1} + \ldots + a_k E_{jk}$$

This means that if Y_i is linearly pseudo-expressed by $\{Y_{j1}, \ldots, Y_{jk}\}$, then E_i is linearly expressed by $\{E_{j1}, \ldots, E_{jk}\}$. □