Asymptotically Optimal Two-Round Perfectly Secure Message Transmission

Saurabh Agarwal[1], Ronald Cramer[2] and Robbert de Haan[3]

[1] Basic Research in Computer Science (http://www.brics.dk), funded by Danish National Research Foundation. EMAIL: saurabh@daimi.au.dk
[2] CWI, Amsterdam & Mathematical Institute, Leiden University, The Netherlands. URL: http://www.cwi.nl/~cramer, http://www.math.leidenuniv.nl/~cramer
[3] CWI, Amsterdam, The Netherlands. URL: http://www.cwi.nl/~haan

Abstract. The problem of perfectly secure message transmission concerns two synchronized non-faulty processors sender ($\mathcal{S}$) and receiver ($\mathcal{R}$) that are connected by a synchronous network of $n \geq 2t+1$ noiseless 2-way communication channels. Their goal is to communicate privately and reliably, despite the presence of an adversary that may actively corrupt at most t of those channels. These properties should hold information theoretically and without error.
We propose an asymptotically optimal solution for this problem. The proposed protocol consists of two communication rounds, and a total of $O(\ell n)$ bits are exchanged in order to transmit a message of ℓ bits. Earlier, at CRYPTO 2004, an equally optimal solution has been claimed. However, we give a counter-example showing that their result is not perfectly reliable. The flaw seems to be fundamental and non-trivial to repair. Our approach is overall entirely different, yet it also makes essential use of their neat communication efficient technique for reliably transmitting conflict graphs.
What distinguishes our approach from previous ones is a technique that allows to identify *all actively corrupted channels*, initially trading it off against privacy. A perfectly secure and reliable secret key is then distilled by privacy amplification.

Keywords: reliable and private transmission, information theoretic security, zero-error protocols, communication efficiency.

1 Introduction

The problem of perfectly secure message transmission (PSMT) was first introduced in [2]. In its more general description, it concerns two synchronized non-faulty processors sender ($\mathcal{S}$) and receiver ($\mathcal{R}$) that are connected by a synchronous network of n noiseless 2-way communication channels. The goal is for $\mathcal{S}$ to communicate a secret message M, drawn from a finite field K, to $\mathcal{R}$. This should be done in such a way that for any set of at most t channels that is controlled and coordinated by an active adversary ($\mathcal{A}$), the adversary is neither able to disrupt the transmission of M to $\mathcal{R}$, nor is he able to obtain any new information about M. Moreover, these properties should hold information theoretically and without error. Of course $\mathcal{S}$ and $\mathcal{R}$ have no a priori knowledge of which particular channels are under the control of $\mathcal{A}$.

In general, such perfect communication is not possible for every selection of t and n. The good values for t and n depend on whether communication is 1-way (only from $\mathcal{S}$ to $\mathcal{R}$) or 2-way ($\mathcal{S}$ and $\mathcal{R}$ converse). It has been established that $n \geq 3t + 1$ is necessary and sufficient for 1-way communication and $n \geq 2t + 1$ is necessary and sufficient for 2-way communication [2].

The efficiency of any protocol solving the PSMT problem is typically measured in three parameters; the number of channels t that can be controlled by the adversary, the number of rounds[4] r of the protocol and the number of bits sent to reliably and privately communicate one bit of actual message from $\mathcal{S}$ to $\mathcal{R}$. The last parameter is also known as the communication complexity of the protocol.

Clearly a protocol which can tolerate the strongest adversary, uses the minimum number of rounds and which has a minimal communication complexity is preferred. For $r = 1$ and $n = 3t + 1$, a protocol with optimal communication complexity is known [2,5]. On the other hand, no protocol with optimal communication complexity for $r > 1$ and $n = 2t + 1$ is known. In this paper we give an asymptotically optimal protocol for $r = 2$ and $n = 2t + 1$. The authors of an earlier paper [5] claim to have found an optimal protocol for $r = 2$ and $n = 2t + 1$, but as we note in Section 3.3 the protocol of [5] is not perfectly reliable and therefore not a perfectly secure message transmission protocol.[5]

1.1 Organization of the paper

In Section 2 we isolate some of the known basic techniques that are used throughout the paper. In Section 3, we give an overview of prior work in this area and in particular give a counter-example that breaks the perfect reliability of the protocol proposed in [5]. In Section 4, we introduce the new techniques which lead to our asymptotically optimal protocol using the communication efficient technique for reliably transmitting conflict graphs from [5]. The latter is then described in Section 5, where also the communication complexity is worked out.

2 Preliminaries

2.1 Shamir Sharing

Let K be a finite field with $|K| > n$. A selection $\{s_1, s_2, \ldots, s_n\}$ of shares according to Shamir's (t, n)-threshold secret sharing scheme [4] over the field K has the following properties,

[4] A round or a phase is a single sided communication from $\mathcal{S}$ to $\mathcal{R}$ or vice versa.

[5] This paper also presents a protocol which ends in a single round with high probability. However as the authors of [5] have noted in their presentation at CRYPTO 2004, this protocol is incorrect. Failure can be enforced with probability close to 1.

(1) any $t+1$ shares fix all other shares in the selection and
(2) given any subset of $t+1-e$ shares, no information can be obtained about any disjoint subset of e shares.

Definition 1. *We say that a set of shares is* consistent *when there is a selection in the Shamir secret sharing scheme that leads to this set of shares. A set of shares which is not consistent is called* inconsistent*.*

Definition 2. *We say that a channel is* corrupted *if information sent on the channel is changed before it reaches its destination in any round of the protocol. Otherwise the channel is called* uncorrupted*. In other words, the information sent on an uncorrupted channel is unchanged in any round of the protocol.*

Definition 3. *We say that a value is* broadcast *if it is simultaneously sent over all communication channels. Since the value can then be correctly determined using majority voting on the other end, such values are always perfectly reliably transmitted. Since corruptions that occur during broadcasts are easy to detect, in the sequel we assume without loss of generality that broadcasts occur without any corruptions on the channels.*

2.2 Protocol Π_i

There is a two-round subprotocol Π_i that has implicitly been used both in the protocol of [3] and also in the incorrect protocol of [5]. During this protocol, $\mathcal{R}$ attempts to privately transmit a value $s \in K$ to $\mathcal{S}$ over the i^{th} channel and obtains feedback about the result afterwards. It has the following functionality:

- If channel i is corrupted, it is disqualified by $\mathcal{R}$ after the second round.
- If channel i is not corrupted, $\mathcal{R}$ is certain that $\mathcal{S}$ correctly received s.
- If channel i is not under the control of $\mathcal{A}$, $\mathcal{A}$ obtains no information about the value s.

We now briefly describe the details of the protocol. Assume that $|K| > n$. First, $\mathcal{R}$ selects an arbitrary set $\{s_1, s_2, \ldots, s_n\}$ of shares in Shamir's (t,n)-threshold secret sharing scheme where the i^{th} share is s, corresponding to a randomly chosen secret. $\mathcal{R}$ then sends all the shares over channel i and the share s_j over every other channel j. We denote the shares received on channel i by $\{s'_1, s'_2, \ldots, s'_n\}$ and the shares received on the other channels j by t'_j. This completes the first round. If the received set $\{s'_1, s'_2, \ldots, s'_n\}$ of shares is not consistent, $\mathcal{S}$ disqualifies channel i and broadcasts a notification to $\mathcal{R}$. Otherwise, for every pair of values such that $s'_j \neq t'_j$, $\mathcal{S}$ broadcasts[6] j and s'_j. Finally, $\mathcal{R}$ verifies for all received values whether $s'_j = s_j$ and disqualifies channel i if this is not the case or if $\mathcal{S}$ disqualified channel i. The properties now follow from a straightforward application of (1) and (2) for the value s.

[6] In [5] these shares are actually not transmitted using broadcast, but the functionality is the same.

The symmetry of conflicts in Π_i We now describe an interesting property of the first round of the protocol Π_i that is used to break the perfect reliability of the protocol from [5] in Section 3.3. Let $\{s_1, s_2, \ldots, s_n\}$ be the set of shares in Shamir's (t, n)-threshold secret sharing scheme that has been transmitted by $\mathcal{R}$ over the i^{th} channel in the first round of protocol Π_i. We denote the set of shares received on channel i by $\{u_1, u_2, \ldots, u_n\}$ and the shares received over the other channels j by v_j. We say that channel j *conflicts with* channel i if $u_j \neq v_j$.

Assume that channel i is under the control of the adversary and that the set $\{u_1, u_2, \ldots, u_n\}$ is a consistent set of shares that differs from the original set $\{s_1, s_2, \ldots, s_n\}$. If t of the original values s_j that $\mathcal{R}$ sent over the uncontrolled channels were kept intact and the shares sent over the remaining $t-1$ controlled channels were changed to u_j, there will be only one pair u_j, v_j for which $u_j \neq v_j$ $(= s_j)$, i.e., there will be only one uncontrolled channel j that conflicts with channel i. Note that this is also the minimal number of conflicts possible when the set of shares sent over channel i is altered, since the shares sent over the $t+1$ uncontrolled channels completely fix the original set of shares and therefore cannot all be consistent with the altered set of shares.

Now consider the situation where channel i is not under the control of the adversary and where $\{u_1, u_2, \ldots, u_n\}$ is the original set of shares. Furthermore, assume that only the share on the controlled channel j is modified, resulting in an altered share s_j. Then $u_j \neq s_j$ and the received shares are exactly the same as in the previous situation. This implies that $\mathcal{S}$ cannot distinguish between the two situations.

2.3 Information Reconciliation

In this section we describe an information reconciliation technique that is based on an idea by Sayeed and Abu-Amara [3]. We assume that $\mathcal{S}$ has a vector consisting of $n = 2t+1$ uniformly random values and that at least $t+1$ of these values are known by $\mathcal{R}$. Furthermore, suppose that the adversary $\mathcal{A}$ knows at most t of these values and nothing else about the other values. The goal is for $\mathcal{S}$ to transmit enough information to allow $\mathcal{R}$ to recover the random vector, without allowing $\mathcal{A}$ to do the same.

Concretely, let K be a finite field with $|K| > n+t$ and assume that $\mathcal{S}$ has a uniformly random vector $\underline{v} = (v_1, v_2, \ldots, v_n) \in K^n$. We now consider the vector $\underline{v}$ as a set of values for the first n shares in Shamir's $(n-1, n+t+1)$-threshold secret sharing scheme. By property (1), these shares fix the remaining $t+1$ shares in the scheme. Let $\mathcal{S}$ broadcast t of these remaining shares to $\mathcal{R}$.

$\mathcal{R}$ now knows at least n different shares in the scheme, which completely fix all the shares in the scheme and in particular the first n shares. Since $\mathcal{A}$ can learn the value of at most $n-1$ different shares, it follows from (2) that $\mathcal{A}$ cannot completely reconstruct the first n shares in the scheme. Therefore, the requirements are met.

2.4 Privacy Amplification

We now describe a well-known technique for perfect privacy amplification, that is very well-suited for use in PSMT protocols. Suppose $\mathcal{S}$ and $\mathcal{R}$ share b uniformly random elements in K and that it is promised that $a < b$ of these elements are completely unknown to the adversary. Then there is a simple technique that allows $\mathcal{S}$ and $\mathcal{R}$ to non-interactively generate a random elements about which $\mathcal{A}$ has no information.

Assume that $|K| > a + b$. Then we can view the b shared random elements as the first b shares in a Shamir's $(b - 1, a + b)$-threshold secret sharing scheme. Again these shares fix all the other shares in the selection and by property (2), $\mathcal{A}$ has no information about the values of the a 'new' shares. These shares can therefore be taken as the outcome of the privacy amplification.

3 Earlier Protocols for PSMT

3.1 Overview

The main known protocols for perfectly secure message transmission all roughly have the same structure.

1. First $\mathcal{S}$ and $\mathcal{R}$ interact in such a manner that they obtain sufficiently correlated information, about which $\mathcal{A}$ has sufficient uncertainty.
2. Then $\mathcal{S}$ and $\mathcal{R}$ perform information reconciliation, i.e., they agree upon certain information that is not completely known to $\mathcal{A}$.
3. Subsequently $\mathcal{S}$ and $\mathcal{R}$ non-interactively perform privacy amplification on this information and obtain a random string which is completely unknown to $\mathcal{A}$.
4. This string now serves as a one-time pad, $\mathcal{S}$ encrypts the *actual* message with it and communicates the result to $\mathcal{R}$.

For simplicity we leave out the encryption part of the procedure in the sequel and only focus on establishing the one-time pad. The first known two round protocol for PSMT is due to Dolev et al. [2]. The communication complexity of this protocol is $O(2^n)$, and therefore it is not efficient in terms of communication complexity. The article by Sayeed and Abu-Amara [3] presents the first efficient two round PSMT protocol, which achieves a communication complexity of $O(n^3)$. In [5], the authors claim to present a protocol with a communication complexity of $O(n)$, which can be shown to be optimal. However, the protocol in [5] is not perfectly reliable as is shown in Section 3.3. Furthermore, it seems to be nontrivial to repair the protocol, indicating that some new techniques may be required. As a consequence, the problem of finding two-round PSMT protocols with better communication complexity is still open. In the sequel we demonstrate some new techniques, and show that at least *asymptotically* a linear communication complexity can be obtained.

3.2 Protocol by Sayeed and Abu-Amara

The protocol due to Sayeed and Abu-Amara [3] is easily explained in terms of the techniques described in Section 2. Initially, Π_i is executed in parallel for every channel $i \in \{1, 2, \ldots, n\}$. This results in n random values $\{v_1, v_2, \ldots, v_n\}$ that are received by $\mathcal{S}$, of which at least $t+1$ are equal to values that were originally transmitted by $\mathcal{R}$. Furthermore, $\mathcal{R}$ finds out in the second round which values were correctly received. Also, $\mathcal{A}$ knows at most t of the values received by $\mathcal{S}$, which correspond to the channels that are under his control. $\mathcal{S}$ and $\mathcal{R}$ can now apply the information reconciliation technique from Section 2.3 and the privacy amplification technique from Section 2.4 to obtain a completely secret element v, which can then be used as a one-time pad.

3.3 Protocol of CRYPTO 2004

We demonstrate in this section how the protocol Π_i is used to obtain a two round protocol in [5]. To make it clear that the protocol is not perfectly reliable, we show a strategy for $\mathcal{A}$ so that $\mathcal{R}$ cannot decide without error probability between two different random pads at the end of the protocol. Since the adversary can guess the information sent over the channels that are not under his control and the protocol should have zero error probability, we can assume without loss of generality that $\mathcal{A}$ also has full knowledge of the information that is transmitted over the uncontrolled channels.

To further simplify matters we only discuss the core functionality of the protocol, and only for the situation where $> 2t/3$ of the channels are corrupted in the first round. We can do this, since the protocol in [5] applies different techniques depending on whether $\leq 2t/3$ of the channels are corrupted in the first round or more channels are corrupted. Furthermore, we discuss the protocol using the notation and techniques of this paper.

The protocol from [5] starts by executing the protocol Π_i in parallel for every channel i. The protocol then continues in the second round as follows:

1. An arbitrary set $\{q_1, q_2, \ldots, q_n\}$ of shares in Shamir's $(\lfloor 4t/3 \rfloor, n)$-threshold secret sharing scheme is selected according to a randomly chosen secret.
2. Privacy amplification is applied to these shares, leading to a random vector $\underline{y} = (y_1, \ldots, y_{\lfloor t/3 \rfloor})$ that is later used as a one-time pad to mask and send a message $M \in K^{\lfloor t/3 \rfloor}$.
3. For every channel i, the first round of Π_i is executed again with an arbitrary set $U_i = \{(u_1)_i, (u_2)_i, \ldots, (u_n)_i\}$ of shares for which $(u_i)_i = q_i$, where this time $\mathcal{S}$ is the party transmitting.
4. For every conflict $(s'_j)_i \neq (t'_j)_i$ that occurred during the first execution of Π_i, the value $(u_j)_i$ is broadcast by $\mathcal{S}$. [7]

[7] In [5] these shares are actually not transmitted using broadcast, but the functionality is the same.

We now set $t = 4$ and $n = 9$ and demonstrate the claimed strategy for the adversary so that $\mathcal{R}$ cannot decide without error probability at the end of the protocol between two different one-time pads y_1 and y_2. It then follows that the protocol is not perfectly reliable.

Assume that the channels 1, 7, 8 and 9 are under control of the adversary. Let Q_1 and Q_2 be two sets of shares in Shamir's $(5, 9)$-threshold secret sharing scheme that lead to different pads y_1 and y_2 after privacy amplification, but for which the shares are the same in the 3^{th}, 4^{th}, 5^{th} and 6^{th} position. It is straightforward to show that such sets always exist regardless of the choice of Q_1 and in fact such sets can be found for any choice of y_1 and y_2. We demonstrate a strategy for the adversary such that $\mathcal{R}$ cannot determine whether Q_1 or Q_2 was the original set of shares, due to the fact that $\mathcal{R}$ cannot determine at the end of the protocol whether channel 1 has been corrupted during the protocol or channel 2. Let Q_1 be the set of shares that is selected by $\mathcal{S}$ in step 1. The strategy, which consists of a number of instances of the technique demonstrated in Section 2.2, then works as follows:

- In the first round, the adversary precisely creates conflicts in two directions between the channels 6 and 7, 6 and 8 and 6 and 9 using the technique described in Section 2.2.
- In the second round, the adversary precisely creates conflicts in two directions between the channels 1 and 2, 5 and 7, 5 and 8 and 5 and 9. This is done in such a way that the shares sent on the channels 1, 3, 4, 5 and 6 are consistent with the set Q_2. Since the shares on the uncontrolled channels are not altered, the shares sent on the channels 2, 3, 4, 5 and 6 remain consistent with the set Q_1. This can be done as follows.
  - For $j \in \{1, 2, 7, 8, 9\}$, choose a share $(u'_1)_j$ in such a way that the set of shares that is defined by the shares $(u'_1)_j$, $(u_3)_j$, $(u_4)_j$, $(u_5)_j$ and $(u_6)_j$ contains as the j^{th} share the j^{th} share of Q_2. This is going to ensure that the shares sent on the channels 1, 3, 4, 5 and 6 are consistent with the set Q_2.
  - Let U'_1 be the set of shares that will replace the set U_1, defined by the shares $(u'_1)_1$, $(u_3)_1$, $(u_4)_1$, $(u_5)_1$ and $(u_6)_1$. The corresponding shares sent over the controlled channels are replaced by the new shares in this set. This causes channel 2 to conflict with channel 1.
  - Replace the share $(u_1)_2$ sent over channel 1 by $(u'_1)_2$. This causes channel 1 to conflict with channel 2.
  - For $j \in \{7, 8, 9\}$, let U'_j be the set of shares that will replace the set U_j, defined by the shares $(u'_1)_j$, $(u_2)_j$, $(u_3)_j$, $(u_4)_j$ and $(u_6)_j$. The corresponding shares sent over the controlled channels are replaced by the new shares in these sets. This causes channel 5 to conflict with the channels 7, 8 and 9.
  - Replace the shares $(u_7)_5, (u_8)_5, (u_9)_5$ sent over the channels 7, 8 and 9 by arbitrary different values. This causes the channels 7, 8 and 9 to conflict with channel 5.

In the first round more than $2t/3$ of the channels are corrupted, since the sets of shares are changed on the channels 7, 8 and 9, so the example matches the assumed setting. These three channels will be disqualified at the end of the second round. However, as described in Section 2.2, $\mathcal{R}$ cannot determine after the second application of the first round of Π_i whether channel 1 has been corrupted or channel 2. By design, the received shares on the channels could have come from the set Q_2 if the channels 2, 7, 8 and 9 had been corrupted and the shares on the actual uncorrupted channels are consistent with Q_1. Therefore, $\mathcal{R}$ cannot distinguish between these cases. As a final remark, note that the broadcast shares $(u_j)_i$ from step 4 are superfluous in this scenario, as these shares were already correctly received in the second round.

4 Alternative two round PSMT

In this section we describe the new two round protocol. Using the technique from Section 5.1 and given a large enough message M, it introduces a communication cost of $O(\ell n)$ bits. To this end, we introduce the protocol $\hat{\Pi}_i$, that is based on some completely new techniques. As already noted in Section 3.1, we leave out the encryption part of the procedure and only focus on establishing the one-time pad.

4.1 Protocol $\hat{\Pi}_i$

The main contribution of this paper is the replacement of the protocol Π_i from Section 2.2 by a stronger two-round protocol $\hat{\Pi}_i$ with the following functionality:

- $\mathcal{S}$ and $\mathcal{R}$ both obtain a uniformly random vector $\mathcal{Z}_i = (z_1, z_2, \ldots, z_d) \in K^d$. However, they do not necessarily control which vector this is.
- If channel i is not under the control of $\mathcal{A}$, $\mathcal{A}$ obtains no information about the vector $\mathcal{Z}_i$.

Here $d \in \mathbb{Z}_{>0}$ is some constant value that can be selected before the start of the protocol.

We compare the protocol $\hat{\Pi}_i$ with the protocol Π_i. As shown in the protocol due to Sayeed and Abu-Amara (Section 3.2), after Π_i has been invoked once for every channel, up to t of the values that were actually received by $\mathcal{S}$ may be unknown to $\mathcal{R}$. Therefore, almost all privacy had to be sacrificed during information reconciliation. However, when the protocol $\hat{\Pi}_i$ finishes, information reconciliation has already occurred.

Furthermore, we will show that by choosing a message M of sufficiently large size, the relative amount of privacy that has to be given up during the information reconciliation in the new protocol can be made arbitrarily small, whereas in the protocol due to Sayeed and Abu-Amara this amount is always proportional to the message size. Additionally incorporating the technique from Section 5.1 allows us to obtain the desired communication complexity.

4.2 Sketch of the techniques used

From the discussion in Section 2.2, we see that controlled channels can change their share in order to match an altered set of shares sent over another channel in the protocol Π_i, while these channels cannot be detected due to insufficient feedback. We demonstrate a completely different technique that allows *all* corrupted channels to be detected, even if only a single share has been altered on such a channel. In Section 4.3 we then demonstrate that if $\mathcal{S}$ sends some appropriate additional information, which is similar to the conflict information that is broadcast during the protocol Π_i, $\mathcal{R}$ can reconstruct the altered set $\{s'_1, s'_2, \ldots, s'_n\}$ of shares that $\mathcal{S}$ received.

The key to detecting all corrupted channels is the fact that there are always $t+1$ channels that are not controlled by the adversary. The shares corresponding to these channels completely determine the original set of shares and therefore any combination of shares consisting of one altered share together with the shares corresponding to the $t+1$ uncontrolled channels has to be inconsistent. Such an inconsistent combination allows $\mathcal{R}$ to detect a corrupted channel, since $\mathcal{S}$ can send the altered share to $\mathcal{R}$ and $\mathcal{R}$ can verify whether this share has been altered. However, since it is not known which channels are under the control of $\mathcal{A}$, we need to perform this procedure for all subsets consisting of $t+1$ channels to make sure that the proper subset of $t+1$ channels has been attempted.

4.3 Details of protocol $\hat{\Pi}_i$

Assume that $|K| > n+t$ and let $\mathcal{N} = \{1, 2, \ldots, n\}$. The first round is just as in the protocol Π_i, except that instead of one random set of shares, m sets of shares are initially selected and transmitted.

Round 1 In the first round, $\mathcal{R}$ selects m arbitrary sets of shares according to Shamir's (t, n)-threshold secret sharing scheme for m randomly chosen secret values. We use notation (s_j) to denote the vector of all j^{th} shares and denote the set consisting of all shares by $\{(s_1), (s_2), \ldots, (s_n)\}$. Furthermore, the notation $(s_j)_l$ is used to denote the j^{th} share in the l^{th} set of shares. $\mathcal{R}$ sends the set $\{(s_1), (s_2), \ldots, (s_n)\}$ over the channel i and the vector (s_j) over every other channel j. This completes the first round.

Round 2 Assume that $\mathcal{S}$ receives the set $\{(s'_1), (s'_2), \ldots, (s'_n)\}$ on channel i and vectors (t'_j) on the other channels j, where we define $(t'_i) := (s'_i)$. Without loss of generality, we may again assume that all the m sets of received shares are consistent, since otherwise $\mathcal{S}$ can disqualify channel i and notify $\mathcal{R}$.

We now perform the verification as described in Section 4.2, where for every channel j and every combination of $t+1$ channels $j_1, j_2, \ldots, j_{t+1}$ not including channel j we try to find an index l for which the combination of the shares $(t'_{j_1})_l, (t'_{j_2})_l, \ldots, (t'_{j_{t+1}})_l$ and $(t'_j)_l$ is not consistent. For every such selection of channels, if such an index l exists $\mathcal{S}$ broadcasts one such share $(t'_j)_l$ and its index l to $\mathcal{R}$, who can then verify whether this share is correct. In Lemma 1, we show that this approach allows $\mathcal{R}$ to identify all corrupted channels j.

It is clear that in the above procedure many shares are broadcast that were not initially known to $\mathcal{A}$. In order to remove all information that $\mathcal{A}$ may have gained due to this, all sets of shares corresponding to the broadcast shares are discarded. Of the remaining sets, the i^{th} shares are then kept. Due to the similarities with the protocol Π_i, it now follows that these shares are known to $\mathcal{A}$ only if channel i is under his control, whereas $\mathcal{A}$ gains no information about these shares otherwise.

Concretely, for $j = 1, 2, \ldots, n$, let the set $\mathcal{Q}_j = \{V_{j1}, V_{j2}, \ldots, V_{jw}\}$ consist of all combinations of $t+1$ channels that do not include channel j, i.e., $\mathcal{Q}_j = \{V \subset \mathcal{N}\backslash\{j\} \ : \ |V| = t+1\}$. Then for all members of the set $\mathcal{Q}_j$ $(1 \leq j \leq n)$ the corresponding shares determine the full set of shares and every set $\mathcal{Q}_j$ has the same number of elements (namely $w = \binom{n-1}{t+1}$ elements).

The following protocol now specifies the verification step that is performed after the first round.

Protocol 1 (Classify channels)

1. *Let $j \in \mathcal{N}$ and $k \in \{1, \ldots, w\}$. Then either the received shares in the $(t+2)$-sized set of shares $W_{jkl} := \{(t'_j)_l\} \cup \{(t'_z)_l \ : \ z \in V_{jk}\}$ are consistent for every $l \in \{1, \ldots, m\}$, or there is a smallest integer l_{jk} such that the shares in $W_{jkl_{jk}}$ are not consistent.*
   Taking $l_{jk} = 0$ when the shares in W_{jkl} are consistent for every l, we let $L_j = (l_{j1}, \ldots, l_{jw})$ be the vector containing all such smallest indices, $I_j = \{l_{j1}, \ldots, l_{jw}\}\backslash\{0\}$ be the corresponding set of indices and define
   $$E_j := ((t'_j)_{l_{jm}})_{m \in \{1,\ldots,w\}: l_{jm} \neq 0}.$$
2. *For $j = 1, \ldots, n$, $\mathcal{S}$ broadcasts L_j and E_j. Furthermore, $\mathcal{S}$ defines*
   $$\mathcal{Z}_i := ((s'_i)_l)_{l \in \{1,\ldots,m\} \backslash (\bigcup_{j=1}^{n} I_j)}.$$

During Protocol 2 almost the same 'conflict information' is transmitted as in the second round of Π_i, with as its main difference that now whenever $(s'_j)_l \neq (t'_j)_l$ both conflicting values are returned instead of only the value $(s'_j)_l$. However, this information is used in a completely different way. Whereas in previous protocols this information was required to discover channels that have been corrupted, that functionality is now completely superfluous due to the previous protocol. Instead, the information transmitted during Protocol 2 is exactly sufficient to allow for complete information reconciliation by $\mathcal{R}$, in the sense that it helps $\mathcal{R}$ to completely determine what $\mathcal{S}$ received in the first round.

Protocol 2 (Gather reconciliation information)

1. *Define $\mathcal{C}_i := \{(l, j, (s'_j)_l, (t'_j)_l) : (s'_j)_l \neq (t'_j)_l, j \in \mathcal{N}\backslash\{i\}, l \in \{1, \ldots, m\}\}$.*
2. *$\mathcal{S}$ broadcasts $\mathcal{C}_i$.*

After $\mathcal{S}$ has finished transmitting, $\mathcal{R}$ can now execute the following protocol to reconstruct $\mathcal{Z}_i$. Note that nothing needs to be transmitted anymore at this point.

Protocol 3 (Reconcile)
According to Lemma 1, a channel j has been corrupted iff there exists an entry $(t'_j)_l$ in E_j such that $(t'_j)_l \neq (s_j)_l$. This allows $\mathcal{R}$ to completely split up the set of channels in a set U_c of channels that have been corrupted and a set U_u of uncorrupted channels.

1. *First assume that $i \in U_u$. Then*

$$\mathcal{Z}_i = ((s'_i)_l)_{l \in \{1,\ldots,m\} \setminus (\bigcup_{j=1}^n I_j)} = ((s_i)_l)_{l \in \{1,\ldots,m\} \setminus (\bigcup_{j=1}^n I_j)},$$

   which is a vector known to $\mathcal{R}$.
2. *Now assume that $i \in U_c$. Fix $l \in \{1,\ldots,m\}$, let $H \subset U_u$ be a set of $t+1$ uncorrupted channels and take $j \in H$. Then either $(t'_j)_l = (s'_j)_l$ or $(l, j, (s'_j)_l, (t'_j)_l) \in \mathcal{C}_i$. In the first case, $(t'_j)_l$ (which is then equal to $(s_j)_l$, which is known to $\mathcal{R}$) gives a share in the l^{th} set of shares $\{(s'_1)_l, \ldots, (s'_n)_l\}$, and in the second case $(s'_j)_l$ gives a share in the l^{th} set of shares.*

   Since the l^{th} set of shares is consistent, the $t+1$ shares that can be obtained in this way fix all the shares in this set and in particular the share $(s'_i)_l$. Therefore, $\mathcal{R}$ can obtain $\mathcal{Z}_i$ in this case as well.

It follows that $\mathcal{R}$ and $\mathcal{S}$ both obtain the same vector $\mathcal{Z}_i$ at the end of the protocol. If $\mathcal{Z}_i$ is not empty, the values in the vector are either completely known to the adversary or he has no information about these values, depending only on whether the adversary controls channel i or not. This completes the description of protocol $\hat{\Pi}_i$.

Two round PSMT Assume that we execute the protocol $\hat{\Pi}_i$ in parallel for all n channels. Without loss of generality we may assume that all vectors $\mathcal{Z}_i$ have the same length, since otherwise $\mathcal{S}$ and $\mathcal{R}$ can just remove entries according to some predetermined method. Also, it should be clear that for any i that the protocol is executed for, the set $\bigcup_{j=1}^n I_j$ can contain at most $nw = n\binom{n-1}{t+1}$ indices. Therefore, by choosing m large enough, the length of the vectors $\mathcal{Z}_i$ can in fact be fixed to any nonzero value, so we can assume that the vectors $\mathcal{Z}_i$ have nonzero length.

At most t of the vectors $\mathcal{Z}_i$ are known to the adversary at the end of the n parallel executions of $\hat{\Pi}_i$, whereas he has no information about the remaining vectors. Therefore, applying a parallel version of the privacy amplification technique from Section 2.4 on these n vectors gives $t+1$ completely secret vectors. The values in these vectors can then be used in the second round by $\mathcal{S}$ to one-time-pad encrypt message elements from K.

Proofs We now provide the results that support our claims. The following lemma shows that $\mathcal{R}$ can determine for a channel j whether it has been corrupted in the first round of $\hat{\Pi}_i$ by comparing the received values in the set E_j with the original values that were transmitted on channel j in the first round.

Lemma 1. *Fix any $j \in \mathcal{N}$. Then $(t'_j)_l \neq (s_j)_l$ for some $l \in \{1, \ldots, m\}$ iff there is a value $l' \in \{1, \ldots, m\}$ such that $(t'_j)_{l'} \neq (s_j)_{l'}$ and $(t'_j)_{l'}$ is an entry of E_j.*

Proof. ($\Leftarrow$) Trivial.

($\Rightarrow$) If $(t'_j)_l \neq (s_j)_l$ for some $l \in \{1, \ldots, m\}$, then there is a set V_{jk} of $t+1$ uncontrolled channels that does not contain channel j. In particular, the shares in the set W_{jkl} are inconsistent, since the shares corresponding to these uncontrolled channels lead to share $(s_j)_l$ for j, which is different from $(t'_j)_l$.

Now let l' be the smallest value for which the shares in the set $W_{jkl'}$ are inconsistent. Since the channels in V_{jk} are not under control of the adversary, the shares in the set $W_{jkl'} \backslash \{(t'_j)_{l'}\}$ lead to share $(s_j)_{l'}$ for channel j, where $(t'_j)_{l'} \neq (s_j)_{l'}$ since otherwise the shares would be consistent. Since $(t'_j)_{l'}$ is an entry of E_j by definition, the lemma follows.

At first sight, it may seem that $\mathcal{A}$ can deduce information from the minimum collision indices l_{jk} that are broadcast during Protocol 2. However, the lemma below shows that this is not the case.

Lemma 2. *The values l_{jk} in Protocol 1 completely depend on the actions of the adversary in the first round. In particular, these values are known to the adversary even before they are broadcast in the second round.*

Proof. By Lagrange's theorem, a unique linear relation $\sum_{i=1}^{t+1} \lambda_i s_i = s_{t+2}$ necessarily holds for a consistent set $\{s_1, \ldots, s_{t+2}\}$ of shares in Shamir's (t, n)-threshold secret sharing scheme, where the λ_i's are publicly known constants that only depend on the a priori fixed evaluation points on the used secret sharing polynomial. Lets assume that the first $e <= t$ shares are replaced by values s'_i. It is straightforward to verify that the new set $\{s'_1, .., s'_e, s_{e+1}, .., s_{t+2}\}$ is consistent iff $\sum_{i=1}^{e} \lambda_i s'_i + \sum_{i=e+1}^{t+1} \lambda_i s_i = s_{t+2}$. This is the case iff $\sum_{i=1}^{e} \lambda_i (s_i - s'_i) = 0$. However, the values $s_i - s'_i$ are chosen by and known to the adversary. Therefore, the adversary already knows beforehand whether any particular received subset of $t+2$ shares is consistent or not and can in particular predict all the minimum indices l_{jk}.

Proposition 1. *If $(l, j, (s'_j)_l, (t'_j)_l) \in \mathcal{C}_i$, then at least one of the channels i and j has been corrupted. Furthermore, both $(s'_j)_l$ and $(t'_j)_l$ were already known to the adversary at the end of the first round.*

Assume that $\mathcal{A}$ does not control channel i. Since the first round of $\hat{\Pi}_i$ is a parallel version of the first round of Π_i, it is clear that $\mathcal{A}$ obtains no information about the i^{th} shares in the sets of shares in the first round. The following lemma states that the adversary does not learn anything new about the values of the entries of $\mathcal{Z}_i$ (provided that $\mathcal{Z}_i$ has any entries at all) in the second round of $\hat{\Pi}_i$. This shows that the proposed protocol is perfectly private.

Lemma 3. *If channel i is not under control of the adversary and $\mathcal{Z}_i$ contains a nonzero number of entries, then the adversary obtains no new information about the values of the entries of $\mathcal{Z}_i$ in the second round.*

Proof. According to Lemma 2, the indices that are transmitted during Protocol 1 are selected by (and therefore known to) the adversary before the execution of the second round. Furthermore, the shares that are transmitted during Protocol 1 are completely uncorrelated with the vector $\mathcal{Z}_i$, since the corresponding sharings are discarded before the vector $\mathcal{Z}_i$ is constructed. Finally, Proposition 1 shows that only information that is already known to the adversary is transmitted during Protocol 2. Therefore, the adversary does not learn anything new about the values of the entries of $\mathcal{Z}_i$ in the second round.

5 Towards linear overhead

5.1 Improved reliable transmission

The most expensive transmission in the described protocols so far, both for the previous protocols as well as for the new one, is the transmission of the collisions in the second round, i.e., the transmission of the values s'_j and/or t'_j for which $s'_j \neq t'_j$. In this section we demonstrate a technique used in [5], which can reduce the communication cost by switching from broadcast to a combination of broadcast and error correcting. We describe this technique for a single value of $l \in \{1, \dots, m\}$ in the new protocol, where we group all shares received on the channels that relate to a set of shares that has index l. It is straightforward to improve the technique we describe by taking all shares into account at once, but this has no impact on the obtained communication complexity.

We start by regrouping the vectors in the sets $\mathcal{C}_1, \dots, \mathcal{C}_n$ as follows: For $l = 1, \dots, m$, define $C_l := \{(i, j, (s'_j)_l, (t'_j)_l) : (l, j, (s'_j)_l, (t'_j)_l) \in \mathcal{C}_i, i \in \{1, 2, \dots, n\}\}$. Then the set C_l contains all conflicts arising from the distribution of the n l^{th} sets of shares. We then move to a more efficient method to reliably transmit the sets C_l. Define the undirected graph $\mathcal{G}_l = (\mathcal{P}, \mathcal{E}_l)$ by

$$(i, j) \in \mathcal{E}_l \Leftrightarrow (i, j, (s'_j)_l, (t'_j)_l) \in C_l \vee (j, i, (s'_i)_l, (t'_i)_l) \in C_l$$

and let M_l be the size of a maximum matching on $\mathcal{G}_l$. From Proposition 1 it follows that every edge in this graph involves at least one channel that has been corrupted. Therefore there are at least M_l channels that have been corrupted in total.

These channels can all be detected by $\mathcal{R}$ using the information from Protocol 1. This implies that $\mathcal{R}$ will be able to discard the shares received on at least M_l channels during the reliable transmission. Therefore, we can use an error correcting code with codewords of length n that can handle M_l erasures and $t - M_l$ errors for the transmission. Using codewords of length n, $n - M_l - 2(t - M_l) = M_l + 1$ shares can now be transmitted.

Since every edge in the graph involves at least one channel that is in the maximum matching, there can be at most $2M_l n$ edges in the graph. In particular, this implies that every set C_l can contain at most $4M_l n$ vectors. Using an error correcting code, every set C_l can thus be transmitted by sending $O(n^2)$ shares over the channels.

5.2 Complexity analysis

Choose a field K such that its elements can be represented using bit strings of length $O(\log(n))$ and assume that m is such that $m > nw \log_n(m)$. The length $\max\{0, m - nw\}$ of the vectors $\mathcal{Z}_i$ can be chosen to be of size $\geq cm$ for any constant c in the interval $(0, 1)$, by enlarging m as necessary. As the privacy amplification section shows, this implies that we can obtain a secret key of size $(t+1)cm = O(nm)$, i.e., of $O(mn \log(n))$ bits. In order to have a PSMT-protocol with a linear communication complexity, the total number of shares transmitted in each round should be $O(mn^2)$ or, stated equivalently, the total number of bits transmitted in each round should be $O(mn^2 \log(n))$. Let us now analyze the communication complexity of the parts of the new protocol.

The first round. For every i, $\mathcal{R}$ sends mn elements over channel i and m elements over every other channel j. This sums up to a total of $O(mn)$ shares that are sent over each channel and therefore to $O(mn^2)$ shares in total that are transmitted in the first round.

Protocol: Classify channels. For every $i \in \{1, 2 \ldots, n\}$, at most $O(w)$ indices and field elements are broadcast at the end of Protocol 1 for every j, so that in total $O(n^2 w(\log(m) + \log(n))$ bits have to be broadcast. This gives $O(n^3 w(\log(m) + \log(n))$ bits that are transmitted during Protocol 1. Our assumption implies that $m > n$, so that this can be rewritten to $O(n^3 w \log(m))$. Furthermore, by assumption $m > nw \log_n(m)$, so that $m \log(n) > nw \log(m)$, i.e., $n^3 w \log(m) = O(mn^2 \log(n))$.

Protocol: Gather reconciliation information. Assume that we regroup the sets C_i during the Protocol 2 as described in Section 5.1 and obtained the sets C_l. Then, using some appropriate padding between the vectors C_l, all information can be transmitted by communicating only $O(mn^2)$ shares.

This completes the analysis.

6 Conclusion

In this paper we have given an asymptotically optimal two round PSMT protocol for $n = 2t + 1$. It is not difficult to extend the protocol for $n > 2t + 1$ as well. In particular, there exists a protocol that asymptotically achieves a constant communication overhead when $t = cn$ for any $0 < c < 1/2$.

The main difference when compared to earlier two round PSMT protocols is the ability to completely isolate corrupted channels. However, this comes at the expense of a high computational cost to both sender and receiver. We do not know whether a similar protocol can exist where sender and receiver are restricted to polynomial time computations (in terms of the number of channels) only.

7 Acknowledgements

The authors would like to thank Serge Fehr, Dennis Hofheinz, Eike Kiltz, Jesper Buus Nielsen and Carles Padró for useful discussions and comments.

References

1. Y. Desmedt and Y. Wang. Perfectly Secure Message Transmission Revisited. In *EUROCRYPT'02*, volume 2332 of *LNCS*, pages 502–517. Springer-Verlag, 2002.
2. D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly Secure Message Transmission. *JACM*, 40(1):17–47, January 1993.
3. H. Sayeed and H. Abu-Amara. Efficient Perfectly Secure Message Transmission in Synchronous Networks. *Information and Computation*, 126(1):53–61, 1996.
4. A. Shamir. How to Share a Secret. *Communications of the ACM*, 22:612–613, 1979.
5. S. Srinathan, A. Narayanan, and C. Pandu Rangan. Optimal Perfectly Secure Message Transmission. In *CRYPTO'04*, pages 545–561, Santa Barbara, California, 2004.