# Synchronous Perfectly Secure Message Transmission with Optimal Asynchronous Fallback Guarantees

Giovanni Deligios[1(✉)] and Chen-Da Liu-Zhang[2]

[1] ETH Zurich, Zurich, Switzerland
giovanni.deligios@inf.ethz.ch
[2] HSLU and Web3 Foundation, Zug, Switzerland
chen-da.liuzhang@hslu.ch

**Abstract.** Secure message transmission (SMT) constitutes a fundamental network-layer building block for distributed protocols over incomplete networks. More specifically, a sender **S** and a receiver **R** are connected via $\ell$ disjoint paths, a subset of which are controlled by the adversary.

*Perfectly-secure* SMT protocols in synchronous and asynchronous networks are resilient up to $\ell/2$ and $\ell/3$ corruptions respectively. In this work, we ask whether it is possible to achieve a perfect SMT protocol that simultaneously tolerates $t_s < \ell/2$ corruptions when the network is synchronous, and $t_a < \ell/3$ when the network is asynchronous.

We completely resolve this question by showing that perfect SMT is possible if and only if $2t_a + t_s < \ell$. In addition, we provide a concretely round-efficient solution for the (slightly worse) trade-off $t_a + 2t_s < \ell$.

As a direct application of these results, following the recent work by Appan, Chandramouli, and Choudhury [PODC'22], we obtain an $n$-party perfectly-secure multi-party computation protocol with asynchronous fallback over any network with connectivity $\ell$, as long as $t_a + 3t_s < n$ and $2t_a + t_s < \ell$.

## 1 Introduction

### 1.1 Motivation

Secure message transmission (SMT) is a fundamental building block that allows to run more complex distributed protocols over incomplete networks (e.g. consensus protocols, secret-sharing, or secure computation protocols). It allows a sender **S** and a receiver **R** of an incomplete network of point-to-point channels to communicate securely [9]. Justified by the fact that in a $\ell$-connected graph there are at least $\ell$ disjoint paths among any two nodes [15], one often considers the abstraction in which **S** and **R** are simply connected via $\ell$ channels (also called wires), representing vertex-disjoint paths in the network graph. Assuming

C.-D. Liu-Zhang—This work was partially carried out while the author was at Carnegie Mellon University, USA. Supported in part by the NSF award 1916939, DARPA SIEVE program, a gift from Ripple, a DoE NETL award, a JP Morgan Faculty Fellowship, a PNC center for financial services innovation award, and a Cylab seed funding award.


F. Baldimtsi and C. Cachin (Eds.): FC 2023, LNCS 13950, pp. 77–93, 2024.
https://doi.org/10.1007/978-3-031-47754-6_5

an adversary that can corrupt at most $t$ parties in the network, this translates to at most $t$ of the $\ell$ wires being under the control of the adversary (the ones containing a corrupted node), while the remaining $\ell - t$ wires can be considered secure channels. In other words, the secure message transmission problem asks to construct a secure channel between $\mathbf{S}$ and $\mathbf{R}$ from $\ell$ channels of which an unknown subset of $t$ is under full control of the adversary.

Protocols for $\mathsf{SMT}$ can be classified with respect to the underlying communication model. Two prominent models in the literature are the synchronous and asynchronous models. In the synchronous model, channels are guaranteed to deliver messages within a known delay. In contrast, in the asynchronous model, the delivery of messages can be delayed arbitrarily by the adversary. As a consequence, parties cannot wait to receive messages from all parties to proceed in the protocol execution, as there is no way to distinguish a corrupted party who does not send a message from an honest party whose message is delayed.

Perfectly secure $\mathsf{SMT}$ can be achieved in the synchronous model if up to $t_s < \ell/2$ wires are corrupted [8,9,13], while perfectly secure $\mathsf{SMT}$ in the asynchronous model can only tolerate up to $t_a < \ell/3$ corrupted wires. It is therefore natural to investigate whether there is a protocol that achieves (simultaneously) security guarantees in both network models. More concretely, we ask the following question:

> *Under what conditions does there exist a perfectly-secure message transmission protocol that tolerates up to $t_s$ wires to be corrupted if the network is synchronous, and also up to $t_a$ if the network is asynchronous?*

We completely resolve this question by providing several feasibility and impossibility results. More concretely, we show that $2t_a + t_s < \ell$ is necessary and sufficient for a perfectly-secure message transmission protocol that tolerates up to $t_s$ (resp. $t_a$) corrupted wires if the network is synchronous (resp. asynchronous).

Together with the result by Appan, Chandramouli, and Choudhury [1] on perfectly-secure synchronous multi-party computation (MPC) with asynchronous fallback, we obtain an $n$-party perfectly-secure synchronous MPC with asynchronous fallback over any network with connectivity $\ell$, as long as $t_a + 3t_s < n$ and $2t_a + t_s < \ell$.

Finally, as a result of independent interest, we show that assuming the slightly worse trade-off[1] of $t_a + 2t_s < \ell$, we can achieve a similar perfectly secure message transmission protocol, but that runs in 3 rounds when the network is synchronous. This round complexity is essentially optimal, given that in the purely synchronous setting the optimal number of rounds is 2 [17,19].

### 1.2 Technical Overview

**Feasibility.** Our feasibility result has three main ingredients:

[1] This trade-off is worse given that $t_a \leq t_s$. Note that any protocol with asynchronous security is also secure when run over a synchronous network.

- A *compiler*, which given black-box access to a synchronous (enhanced) secure message transmission protocol and an asynchronous one, provides a protocol with security in both synchronous (up to $t_s$ corruptions) and asynchronous (up to $t_a \leq t_s$ corruptions) networks, assuming the trade-off $2t_a + t_s < \ell$. Intuitively, the synchronous (respectively asynchronous) protocol should provide most of the security guarantees if the network is synchronous (respectively asynchronous). The synchronous protocol either runs successfully or guarantees that the sender detects that the network is asynchronous, and can fallback on the asynchronous protocol. The main challenge is ensuring that, if the network is synchronous, the adversary cannot convince the sender to run the asynchronous protocol, which only tolerates a lower corruption threshold.
- A *Synchronous* SMT *protocol* with the additional guarantees that, if the network is asynchronous, either the protocol succeeds or the sender *is sure* that the network is asynchronous. The construction is round-based. Intuitively, the sender tries to send a secret pad to the receiver by secret sharing the pad and sending each share over one of the $\ell$ wires. If too many errors were introduced by the adversary, the receiver cannot reconstruct the pad, but can inform the sender (via a reliable public channel that also needs to be constructed, which we denote by RMT). The sender can then detect a faulty wire and repeat the process excluding this wire (with a fresh pad and a lower degree sharing). If the sender and the receiver successfully share a secret pad, the actual message can be one-time-pad encrypted and sent over the public channel.
  The main challenge to overcome is properly dealing with erasures (that can originate from faulty wires or by delays on honest wires). In our model when the network is asynchronous, the adversary can convince the sender to exclude an honest wire by simply delaying a message along this wire by longer than the round time. If the sender excludes too many (honest) wires and decreases the degree of the sharing accordingly, eventually the shares on the $t_a$ *actually corrupted* wires determine the secret pad, and secrecy is lost. This is where the trade-off comes into play: we only allow the sender to eliminate up to $t_s - t_a$ wires. This fixes the problem in the asynchronous setting because the starting degree is $t_s$, so after removing $t_s - t_a$ wires, the remaining degree is still $t_s - (t_s - t_a) = t_a$. Moreover, if the network is synchronous, it is guaranteed that the protocol succeeds at the latest after the last wire is excluded: there are $\ell - (t_s - t_a) = \ell - t_s + t_a$ non-excluded wires (among which $t_a$ are corrupted), and the sharing has degree $t_s - (t_s - t_a) = t_a$. Since $2t_a < \ell - t_s$, the reconstruction is successful. In turn, if at this point the protocol does not succeed, the sender *is sure* that the network is asynchronous. Therefore, the resulting protocol runs in at most $t_s - t_a$ rounds when the network is synchronous.
- An *Asynchronous* SMT *protocol*. This protocol does not require any additional properties for the higher synchronous corruption threshold of $t_s$, and therefore any protocol from the literature can be used in a black-box fashion. Due to space constraints, we report a known construction (using our notation) and prove its security only in the full version of this paper [7].

**Impossibility.** We prove that our feasibility result is tight, by showing that the trade-off assumption $2t_a + t_s < \ell$ on the corruption thresholds we made up to this point is not only sufficient, but also necessary to achieve secure message transmission in this hybrid model. Towards contradiction, consider $2t_a + t_s = n$. Partition the channels into three sets $K, A, B$ of sizes $|A| = |B| = t_a$ and $|K| = t_s$. At a high level, the idea is as follows: the information travelling over the channels in $A$ and $B$ must completely determine the message being transmitted (even if no information is transmitted over $K$). This is because in the synchronous setting, the transmission succeeds when there are $t_s$ corruptions. However, if the network is asynchronous, the adversary can delay all the information via the channels in $K$, and control *half* of the remaining channels, which are enough to tamper with the output of the receiver. Proving this precisely requires a carefully designed scenario-based argument, that can be found in the full version of this paper [7].

**Round-Efficient Synchronous SMT with Sub-optimal Trade-Off.** We slightly strengthen these assumptions to $t_a + 2t_s < n$ to achieve a protocol that almost achieves the optimal round complexity of protocols in the purely synchronous model. Intuitively, the stronger trade-off helps for the following reason: if the network is asynchronous, the adversary can delay messages on up to $t_s$-wires (and change those on up to $t_a$), and the receiver can still not be sure the network is asynchronous (the $t_s$ erasures could also originate on wires in a synchronous network). During the transmission of a secret pad, this results in $t_s + t_a$ *actual* wrong shares. Under the stronger assumption, $t_s + t_a < n - t_s$, which is the number of wrong shares that can be tolerated (in the sense of at least detected) in the purely synchronous setting. Therefore, erasures can simply be treated as wrong values, greatly reducing the need for interaction between **S** and **R**.

### 1.3 Related Work

**Synchronous Protocols with Asynchronous Fallback.** A recent line of works [1–4,6,11,16] has investigated the feasibility and efficiency of distributed protocols (consensus and secure computation protocols) that are secure in both synchronous and asynchronous networks. All these works assume a complete network of point-to-point channels among the parties. Our work expands upon this line by considering the simplest building block for distributed protocols over incomplete networks.

**Secure Message Transmission.** The problem of SMT in synchronous networks has been widely investigated [9,10,14,18–20]. Perfectly-secure SMT can be achieved, allowing multiple rounds of interaction between the sender and the receiver, if and only if $t < \ell/2$ channels are under control of the adversary [9]. Several works focused on improving the round complexity, achieving optimal 2-round constructions [17,19]. In the asynchronous model, the number of corrupted channels tolerated decreases to $t < n/3$ for perfect security, but interestingly it

is still possible up to $t < \ell/2$ corruptions [5] when allowing a small probability of error. In [12] the authors investigate SMT in a model where some channels are synchronous and some are asynchronous. They prove that PSMT in this model is impossible *unless the synchronous channels alone already allow* for PSMT. This is in contrast to our model in which the parties are unaware of the network conditions at execution time. The argument they use is similar to the one in Sect. 4.

## 2 Preliminaries

### 2.1 Model

**Adversary.** We consider an active threshold adversary which is allowed to adaptively (based on the information gathered during the execution of the protocol) corrupt a subset of at most $t$ of the parties (in the secure message transmission abstraction, this amounts to corrupting $t$ channels). We assume that the adversary is computationally unbounded and we consider information theoretic security for our protocols.

**Network Topology.** We consider an incomplete network of point-to-point secure channels among parties. We identify the network as a graph, where parties represent vertices and channels represent edges. We say a graph is $\ell$-connected if $\ell$ is the minimum number of edges that must be removed in order to disconnect any two vertices in the graph (two vertices are disconnected if there is no path with these vertices as endpoints). The connectivity $\ell$ is equal to the number of disjoint paths between any two given vertices [15]. We assume that the network topology is fixed and known to the parties before executing a protocol.

**Communication Model.** We consider a model in which parties have access to local clocks and are not a priori aware of the network conditions when executing a protocol. We distinguish two possibilities: the *synchronous model* and the *asynchronous model*.

In the synchronous model, the local clocks are synchronized, and messages are guaranteed to be delivered within some known time bound $\Delta$. The communication can then naturally be described as proceeding in rounds, where for $\mathbb{N} \ni r \geq 1$, each message received in the time slot $[r\Delta, (r+1)\Delta)$ (according to the local clock of each party) is regarded as a round $r$ message.

In the asynchronous model, parties do not have access to synchronized clocks. The adversary is allowed to schedule the delivery of messages arbitrarily, but each message sent by honest parties must eventually be delivered (this guarantee is needed if one wishes to make statements about protocol termination). In this setting, one describes protocols in a message-driven fashion. This means that, upon receiving a message, a party adds this message to a pool of received messages and checks weather a list of conditions specified from the protocol is satisfied to decide on its next action (sending a message, producing output, terminating, etc.).

In our model, both descriptions can be adopted. In a round-based protocol, if a message is received outside of the time allocated for a certain round, it is ignored. In the secure message transmission abstraction, the assumptions on the communication network directly translate into assumptions on the $\ell$ wires connecting $\mathbf{S}$ and $\mathbf{R}$. However, the assumed maximum delay on the resulting channels needs to account for the delays of all channels in the corresponding paths (meaning each wire will have a delay of $d \cdot \Delta$, where $d$ denotes the diameter of the network graph).

### 2.2 Definitions

A secure message transmission protocol allows two parties, connected by multiple channels (wires), to communicate securely even when a subset of the channels is under the control of an adversary.

This abstraction captures the scenario in which two parties part of an incomplete network of secure channels wish to communicate securely. Disjoint paths in the network graph serve as channels. A channel is corrupted if at least one of the parties (nodes) on the path is corrupted. Notice that all guarantees are lost if either the sender or the receiver do not follow the protocol.

We slightly deviate from usual definitions by requiring that the sender protocol also produces a Boolean output. Intuitively, the output is 1 if the sender *knows* the protocol succeeded. Similarly, the receiver is allowed to output a value $\perp$. Intuitively, this means they could not produce a valid output.

**Definition 1** *(Secure Message Transmission). Let $\Pi$ be a protocol executed between $\mathbf{S}$ (the sender) with input $m \in \mathbb{F}$ and randomness $r_1$ and output $b \in \{0,1\}$ and $\mathbf{R}$ (the receiver) with randomness $r_2$ and output $v \in \mathbb{F} \cup \{\perp\}$, connected by channels $(c_1, \ldots, c_\ell)$. We say $\Pi$ is a protocol for $\mathsf{SMT}$ achieving:*

- *($t$-**correctness**) if whenever up to $t$ channels are under control of the adversary, if $\mathbf{S}$ has input $m$, then $\mathbf{R}$ outputs $v = m$ and $\mathbf{S}$ outputs $b = 1$;*
- *($t$-**perfect**[2] **privacy**) if for all $m, m'$, for all $k \geq 1$, for all $\mathcal{I} \subseteq \{1, \ldots, \ell\}$ such that $|\mathcal{I}| \leq t$, the distributions of $T^k_{\mathcal{I},m}$ and $T^k_{\mathcal{I},m'}$ are equal, where $T^k_{\mathcal{I},m}$ denotes the random variable whose values are the $k$-th messages travelling on the channels $\{c_i\}_{i \in \mathcal{I}}$ when the sender has input $m$;*
- *($t$-**termination**) if whenever up to $t$ channels are under control of the adversary, $\mathbf{S}$ and $\mathbf{R}$ terminate;*
- *($t$-**weak correctness**) if whenever up to $t$ channels are under control of the adversary, if $\mathbf{S}$ has input $m$, then*
  - *$\mathbf{R}$ outputs $m$ or $\perp$;*
  - *$\mathbf{R}$ outputs $m$ or $\mathbf{S}$ outputs $0$.*

*If $\Pi$ achieves $t$-correctness, $t$-perfect privacy and $t$-termination, we say that $\Pi$ is $t$-perfectly secure.*

[2] By requiring the distributions $T^k_{\mathcal{I},m}$ and $T^k_{\mathcal{I},m'}$ to be statistically close or computationally indistinguishable one obtains the notion of statistical security and computational security. In this paper, we are only concerned with perfect security.

In what follows, unless otherwise stated, an SMT protocol is to be understood as *perfectly* secure. Depending on the assumptions made on the channels $c_i$, we will consider two cases. If the channels are synchronous (cf. Sect. 2.1), we will talk about synchronous SMT (sSMT); if the channels are asynchronous we will talk about asynchronous SMT (aSMT).

## 3 Secure Message Transmission with Fallback

Throughout this section, we work in the abstract setting of an honest sender and receiver connected by $\ell$ channels, $t$ of which are under full control of the adversary, and the remaining $\ell - t$ are secure channels.

We show an SMT protocol which is secure regardless of whether the sender and the receiver are connected by synchronous or asynchronous channels. The protocol tolerates up to $t_s < \ell/2$ channels to be under the control of the adversary if the channels are synchronous, and up to $t_a < \ell/3$ if the channels are asynchronous, under optimal trade-offs on the corruption thresholds $2t_a + t_s < \ell$ (optimality of the trade-offs is discussed in Sect. 4).

### 3.1 Compiler

First, we present a compiler that combines a synchronous sSMT protocol and an asynchronous aSMT protocol to obtain a protocol that is secure in both communication models. The synchronous component needs to provide certain guarantees even the channels are asynchronous, while the asynchronous one does not require any additional guarantees. More specifically let $\Pi_{\mathsf{sSMT}} = (\mathbf{S}_s, \mathbf{R}_s)$ be an SMT protocol with the following properties:

- If $(c_1, ..., c_\ell)$ are synchronous channels: $t_s$-security.
- If $(c_1, ..., c_\ell)$ are asynchronous channels: $t_a$-(perfect) privacy, $t_a$-weak correctness, $t_a$-termination.

Moreover, let $\Pi_{\mathsf{aSMT}} = (\mathbf{S}_a, \mathbf{R}_a)$ be an SMT protocol with the following properties:

- If $(c_1, ..., c_\ell)$ are asynchronous channels: $t_a$-security.

The sender and the receiver first run the synchronous protocol. If the network is synchronous, then $t_s$-security guarantees that the protocol succeeds. In this case, the asynchronous protocol is not run. If the network is asynchronous, $t_a$-weak correctness guarantees that any output by the receiver matches the message sent by the sender. However, in this case the protocol might also fail and the receiver might not produce output. If this happens, $t_a$-weak correctness of the synchronous protocol comes to the rescue again: the sender can detect that something went wrong and run the asynchronous protocol. Asynchronous secure message transmission does not require interaction: if the receiver has already produced output while running the synchronous protocol, they simply ignore

any further messages. Otherwise, $t_a$-security of the asynchronous component guarantees that the protocol terminates successfully. Notice that even when the network is asynchronous, the synchronous protocol still $t_a$-provides privacy. This idea is formalized in the following protocol. Lemmas 1 and 2 are proven in the full version of this paper.

**Protocol** $\Pi_{\mathsf{hSMT}}(\Pi_{\mathsf{sSMT}}, \Pi_{\mathsf{aSMT}})$

**Code for** $\mathbf{S}_h(m, r_1)$**:**

$b \leftarrow \mathbf{S}_s(m, r_1)$;
**if** $b = 1$ **then**
  **return** $b$;
**else**
  $b \leftarrow \mathbf{S}_a(m, r_1)$;
  **return** $b$;
**end if**

**Code for** $\mathbf{R}_h(r_2)$**:**

$v \leftarrow \mathbf{R}_s(r_2)$;
**if** $v \neq \perp$ **then**
  **return** $v$;
**else**
  $v \leftarrow \mathbf{R}_a(r_2)$;
  **return** $v$;
**end if**

**Lemma 1.** *If $(c_1, ..., c_\ell)$ are synchronous channels and at most $t_s$ channels are under control of the adversary, then $\Pi_{\mathsf{sSMT}}$ achieves $t_s$-security.*

**Lemma 2.** *If $(c_1, ..., c_\ell)$ are asynchronous channels and at most $t_a$ channels are under control of the adversary then $\Pi_{\mathsf{hSMT}}$ achieves $t_a$-security.*

### 3.2 Synchronous RMT with Asynchronous Detection

Before describing our construction for $\Pi_{\mathsf{sSMT}}$, it will be useful to discuss the weaker primitive of *Robust Message Transmission* (RMT). Intuitively, an RMT protocol is an SMT protocol that provides no privacy guarantees (i.e. a public channel between the sender and the receiver that the adversary cannot tamper with). More formally, an RMT protocol is a protocol satisfying the correctness and termination properties of Definition 1. In the context of secure message transmission, such a primitive is often referred to as *broadcast*.

Consider the scenario where a sender $\mathbf{S}$ and a receiver $\mathbf{R}$ are connected by $\ell$ channels $(c_1, \ldots, c_\ell)$ of which at most $t < \ell/2$ under control of the adversary and the remaining $\ell - t$ are secure channels. Here RMT can be achieved by $\mathbf{S}$ sending the same message over all channels, and $\mathbf{R}$ taking a majority decision over the received messages (this is the same as encoding and decoding using an $(1, \ell)$-repetition code).

We use RMT as a building block in our synchronous sSMT protocols. To provide the security guarantees we are after in our synchronous model with asynchronous fallback, we require enhanced RMT protocols. More specifically, when the channels are asynchronous and up to $t_a$ are under control of the adversary, we still require that either $\mathbf{S}'s$ message is correctly delivered to $\mathbf{R}$, or that $\mathbf{S}$ detects that something went wrong. This is formalized in the following protocol and lemmas. Please refer to the full version of this paper for proofs [7].

**Protocol** $\Pi^{t_s}_{\mathsf{sRMT}}$

**Code for S**$(m)$**:**

Initialize $b := 0$;
**Round** 1**:** send $m$ over $c_i$ for all $1 \leq i \leq \ell$;
**Round** 2: if `ok` is received over at least $t_s + 1$ channels, set $b := 1$; output $b$ and terminate;

**Code for R**$()$**:**

Initialize $v := \perp$;
**Round** 1: if there is $m \in \mathbb{F}$ received over at least $t_s + 1$ channels, set $v := m$ and send `ok` over $c_i$ for all $1 \leq i \leq \ell$;
**Round** 2: output $v$ and terminate;

**Lemma 3.** *Assume that $t_a \leq t_s < \ell/2$. If $(c_1, ..., c_\ell)$ are synchronous channels and at most $t_s$ channels are under control of the adversary, then $\Pi_{\mathsf{sRMT}}$ achieves $t_s$-correctness and $t_s$-termination.*

**Lemma 4.** *Assume that $t_a \leq t_s < \ell/2$. If $(c_1, ..., c_\ell)$ are asynchronous channels and at most $t_a$ channels are under control of the adversary, then $\Pi_{\mathsf{sRMT}}$ achieves $t_a$-weak correctness and $t_a$-termination.*

### 3.3 Synchronous SMT with Asynchronous Detection

We show a sSMT protocol which is $t_s$-secure when the network is synchronous and $t_a$-secure when the network is asynchronous, under the (provably optimal) trade-off assumption $2t_a + t_s < \ell$.

The protocol takes after one of the first synchronous constructions introduced by Dolev et al. [9]. The idea is the following: the sender $\mathbf{S}$ selects a random pad and secret shares it using a $(\ell, t_s)$-threshold secret sharing scheme, sending each share over a distinct channel. The receiver $\mathbf{R}$ tries to reconstruct the secret from the received shares. If reconstruction fails because too many shares were tampered with by the adversary, the receiver $\mathbf{R}$ sends the received messages back to $\mathbf{S}$ via sRMT (the roles of sender and receiver are reversed in this sub-protocol). The sender $\mathbf{S}$ identifies at least one corrupted channel, and the process is then repeated (with a fresh pad and a lower degree sharing) excluding this faulty channel.

In a purely synchronous setting, in each round of interaction the number of corrupted channels strictly decreases, so that after at most $(t_s + 1)$-rounds $\mathbf{R}$ receives a pad correctly. Once a pad has been transmitted successfully, in the following round $\mathbf{S}$ can use the pad to one-time-pad encrypt the message and send it to $\mathbf{R}$ via $\mathsf{sRMT}$.

In our setting things are more complicated. If the channels are asynchronous, the adversary could convince $\mathbf{S}$ that a certain channel is corrupted by simply delaying the message on this channel by longer than the round time $\Delta$. By doing so, the adversary can force $\mathbf{S}$ to eliminate honest channels one-at-a-time, until the degree of the sharing of the pad is low enough that the $t_a$ known shares determine the secret pad, thus violating privacy.

To overcome this problem, one must keep $\mathbf{S}$ from removing too many channels (at most $t_s - t_a$), so that the degree of the sharing is never smaller than $t_a$. This solves a problem but creates others: since now we can never eliminate all the corrupted channels even if the network is synchronous, how do we guarantee correctness? Our trade-off assumption $2t_a + t_s < \ell$ plays a crucial role here. To be consistent with the rest of the presentation, we explain the protocol using the language of error-correcting codes. Lemma 9 guarantees that, for all $\ell$ and $i < \ell$ there exists a pair $(\mathcal{C}^{(i)}, \mathbf{h}^{(i)})$ where $\mathcal{C}^{(i)}$ is an $(\ell - i, t_s + 1 - i, \ell - t_s)$-linear MDS code such that for all $\mathbf{x} \in \mathcal{C}^{(i)}$ the scalar product $\mathbf{h}^{(i)}\mathbf{x}^T$ is uniformly random in $\mathbb{F}$ even when up to $t_s - i$ symbols of $\mathbf{x}$ are known. Let $\mathsf{decode}_{\mathcal{C}^{(i)}}(\mathbf{y})$ be an (efficient) decoding algorithm for $\mathcal{C}^{(i)}$ returning a pair $(b, \mathbf{x})$. If decoding is successful, then $b = 1$ and $\mathbf{x} \in \mathcal{C}$, otherwise $b = 0$. To ease the notation, we consider that the $\mathsf{sRMT}$ protocol runs in 1 round. Lemmas 5 and 6 are proven in the full version of this paper [7].

**Protocol** $\Pi^{t_s,t_a}_{\mathsf{sSMT}}\left(\Pi^{t_s}_{\mathsf{sRMT}}, \left\{\mathcal{C}^{(k)}, \mathbf{h}^{(k)}\right\}_{k=1}^{t_s-t_a}\right)$

**Code for** $\mathbf{S}(m, r_1)$

$\mathtt{elimChannels} \leftarrow \varnothing$;
$b \leftarrow 0$; // records success of pad transmission
**Round** $2r - 1$**, for** $r \geq 1$ **:**
$k \leftarrow \#\mathtt{elimChannels}$;
**if** $k > t_s - t_a$ **then** // prevents sender from eliminating too many channels
  **return** $b$;
**end if**
$\bar{b} \leftarrow \Pi_{\mathsf{sRMT}}(\mathtt{elimChannels})$; // tell $\mathbf{R}$ what channels to consider
**if** $\bar{b} = 0$ **then**
  **return** $\bar{b}$;
**end if**
$\mathbf{x} \leftarrow_{\$} \mathcal{C}^{(k)}$;
$c_i \leftarrow x_i$; // send $i$-th symbol of $\mathbf{x}$ along $c_i$
**Round** $2r$**:**
$\mathbf{y}' \leftarrow \Pi_{\mathsf{sRMT}}()^a$;
**if** $\mathbf{y}' = \mathtt{ok}$ **then**

15: $e \leftarrow m + \mathbf{h}^{(k)}\mathbf{x}^T$; // one-time-pad encryption
16: $b \leftarrow \Pi_{\mathsf{sRMT}}(e)$;
17: **return** $b$;
18: **end if**
19: **if** $\mathbf{y}' = \perp$ **then**
20: **return** $b$;
21: **end if**
22: $p \leftarrow$ smallest index such that $\mathbf{y}' \neq \mathbf{x}$; // find one corrupted channel
23: $\texttt{elimChannels} \leftarrow \texttt{elimChannels} \cup \{p\}$;

**Code for R()**

24: $v \leftarrow 0$;
25: $b' \leftarrow 0$; // true only successfully communicating ok to the sender// true only after successful decoding

**Round** $2r-1$**, for** $r \geq 1$**:**

25: $\texttt{elimChannels}' \leftarrow \Pi_{\mathsf{sRMT}}()$;
26: **if** $\texttt{elimChannels}' \neq \perp$ **then**
27: $k' \leftarrow \#\texttt{elimChannels}'$;
28: **for** $i \notin \texttt{elimChannels}'$ **do**
29: $y_i \leftarrow c_i$; // only read values on good channels
30: **end for**
31: **if** $y_j \neq \perp$ for all $j \notin \texttt{elimChanels}'$ **then**
32: $(v, \mathbf{x}') \leftarrow \mathsf{decode}_{\mathcal{C}_0}(\mathbf{y})$;
33: **end if**
34: **end if**

**Round** $2r$:

35: **if** $b' = 1$ **then**
36: $e' \leftarrow \Pi_{\mathsf{sRMT}}()$;
37: **if** $e' \neq \perp$ **then**
38: $m' \leftarrow e' - \mathbf{h}^{(k')}\mathbf{x}'$; // one-time-pad decryption
39: **return** $m'$;
40: **end if**
41: **if** $e' = \perp$ **then**
42: **return** $e'$
43: **end if**
44: **end if**
45: **if** $v$=1 **then**
46: $b' \leftarrow \Pi_{\mathsf{sRMT}}(\texttt{ok})$;
47: **end if**
48: **if** $v = 0$ **then**
49: $b' \leftarrow \Pi_{\mathsf{sRMT}}(\mathbf{y})$; // information to identify corrupted channels
50: **end if**

[a] By $\Pi_{\mathsf{sRMT}}()$ we denote running the protocol as the receiver.

**Lemma 5.** *Assume* $2t_a + t_s < \ell$, $t_a \leq t_s$, *and* $t_s < \ell/2$. *Then, if the channels* $(c_1, \ldots, c_\ell)$ *are synchronous and at most* $t_s$ *are under control of the adversary, protocol* $\Pi^{t_a,t_s}_{\mathsf{sSMT}}$ *achieves* $t_s$*-security.*

**Lemma 6.** *Assume* $2t_a + t_s < \ell$, $t_a \leq t_s$, *and* $t_s < \ell/2$. *If the channels* $(c_1, \ldots, c_\ell)$ *are synchronous and at most* $t_s$ *are under control of the adversary, protocol* $\Pi_{\mathsf{sSMT}}^{t_a,t_s}$ *achieves* $t_a$*-weak correctness,* $t_a$*-perfect privacy, and* $t_a$*-termination.*

### 3.4 Asynchronous SMT

We need a protocol that can be used as the asynchronous protocol $\Pi_{\mathsf{aSMT}}$ in the compiler $\Pi_{\mathsf{hSMT}}$ of Sect. 3.1. Since we do not require any ad-hoc properties, we can employ any protocol from the literature in a black-box fashion, but we briefly describe a protocol here, and defer a more formal description of the protocol and a proof of its security to the full version of this paper [7]. The idea is simple: the sender secret shares their input with a $(\ell, t_a)$-threshold secret sharing scheme sending each share along a distinct channel. The receiver waits until they have received $2t_a + 1$ consistent shares, and then reconstructs the secret. The idea might seem overwhelmingly simple, but the lower number of corrupted channels substantially simplifies matters.

## 4 Impossibility Result

We justify the trade-off assumptions made in the SMT constructions from previous sections, and show that the trade-off $2t_a + t_s < \ell$, together with the trivial constraints $t_a \leq t_s$ and $t_s < \ell/2$, is necessary to achieve perfectly secure message transmission in our hybrid model. The following Lemma is inspired by proofs in [2,4,5], from which we also borrow some notation. Due to space constraints, please refer to the full version of this paper for the proof [7].

**Lemma 7.** *Let* $t_a \leq t_s$. *There exists no SMT protocol that is both* $t_s$*-perfectly secure if the channels are synchronous and* $t_a$*-perfectly secure if the channels are asynchronous, for* $t_s + 2t_a \geq \ell$.

## 5 Round-Efficient Synchronous SMT with Sub-optimal Trade-Off

Assuming the trade-off $t_a + 2t_s < \ell$, we show a protocol $\Pi_{\mathsf{sSMT}}$ with the properties required for the compiler presented in Sect. 3.1 and that runs in 3 rounds when the network is synchronous. This (almost) matches the optimal round complexity of purely synchronous protocols (2 rounds). Our construction adapts known ideas (cf. [17,19]) to the context of security with fallback. Due to space constraints, we assume that the reader is familiar with the basic theory of error correcting codes. More details can be found in the full version of this paper [7]. Below, we report some facts that will be needed in our constructions: once again, we refer the reader to [7] for proofs.

**Lemma 8 ([19], Lemma 2).** *Let $\mathcal{C}$ be an $(\ell, k, d)$-linear code over $\mathbb{F}_q$. Let $\mathbf{H}$ be the parity-check matrix of $\mathcal{C}$. Let $E$ be a linear subspace of $\mathbb{F}_q^n$ such that $w(e) < d$ for all $e \in E$. Then*

$$\begin{aligned} \sigma|_E : E &\to \mathbb{F}_q^{\ell-k} \\ e &\mapsto \mathbf{H}e^T \end{aligned}$$

*is injective.*

**Definition 2.** *Let $\mathcal{Y} \subseteq \mathbb{F}_q^n$. A pseudo-basis of $\mathcal{Y}$ is a subset $\mathcal{W} \subseteq \mathcal{Y}$ such that $\sigma(\mathcal{W})$ is a basis of the linear subspace $\langle \sigma(\mathcal{Y}) \rangle$ of $\mathbb{F}_q^{n-k}$.*

Let $U$ denote a uniformly distributed random variable over $\mathbb{F}_q$, and let $\mathbf{X} = (X_1, \ldots, X_\ell)$ denote a uniformly distributed random variable over $\mathcal{C}$.

**Lemma 9.** *There exists an $(\ell, t_s + 1, \ell - t_s)$-linear code $\mathcal{C}$ and a vector $\mathbf{h} \in \mathbb{F}_q^n$ such that, for all $I \subseteq \{1, \ldots, \ell\}$ with $|I| \leq t_s$, the joint distributions $((X_i)_{i \in I}, U)$ and $\left((X_i)_{i \in I}, \mathbf{h}\mathbf{X}^T\right)$ are equal.*

The intuition is the following: the receiver $\mathbf{R}$ picks $\ell$ random field elements, encodes them using an $(\ell, t_s, t_s + t_a + 1)$ MDS code, and then sends the $i$-th coordinate of the each code-word to the sender via channel $c_i$. The sender receives these code-words with errors introduced by the adversary. Notice that, if the network is asynchronous, the adversary can modify up to $t_a$ symbols of a code-word *and* erase up to $t_s$ symbols. However, we can still ensure that the $t_s + t_a$ errors occur at the same coordinates for all words: if the coordinates at which erasures happen exceed $t_s$, then the sender knows that the channels are asynchronous.

Once the error versions of the code-words have been received, the sender $\mathbf{S}$ computes a pseudo-basis, and communicates it to $\mathbf{R}$ via $\mathsf{RMT}$ together with the syndromes of the errors introduced on code-words that are *not* in the pseudo-basis. With this information receiver $\mathbf{R}$ can now compute all the errors introduced by the adversary on *all* the code-words sent to $\mathbf{S}$. The code-words in the pseudo-basis have been revealed to the adversary, but the remaining words can now be used as shared secret randomness between $\mathbf{S}$ and $\mathbf{R}$ to one-time-pad encrypt messages and communicate them via $\mathsf{RMT}$, as their syndromes leak no information about them.

In general, error-correcting codes need not give any privacy guarantees. For our purposes, however, the knowledge that the adversary gains by seeing up to $t_s$ coordinates of a code-word must not completely determine the code-word (the remaining entropy can be extracted to use as an encryption pad). Considering appropriate codes solves this issue: it is well-known that certain classes of codes are equivalent to threshold-secret sharing schemes. Lastly, in order for $\mathbf{R}$ to correctly compute the errors introduced by the adversary, the minimum distance of the code used must be greater than $t_s + t_a$. Lemma 9 shows how to construct a code with all the required properties, under the assumption that $t_a + 2t_s < \ell$.

Let $\mathsf{PseudoBasis}_{\mathcal{C}}\left(\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(q)}\right)$ be an algorithm that, given $q$ vectors as input with $q \geq \ell - \dim(\mathcal{C})$, efficiently computes a pseudo-basis for these vectors.

Let $\mathsf{ComputeErrors}_{\mathcal{C}}\left(\mathcal{W}, \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(q)}, \sigma, p\right)$ be an algorithm that, given a pseudo-basis of some corrupted versions of $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(q)}$, computes the error introduced on $\mathbf{x}^{(p)}$ from the syndrome $\sigma = \sigma(\mathbf{e}^{(p)})$. Such an algorithm can be found in the full version of this paper [7]. Let $\mathcal{C}$ and $\mathbf{h}$ be as in Lemma 9. Lemmas 10 and 11 are proven in [7].

**Protocol** $\Pi^{t_s,t_a}_{\mathsf{sSMT}}(\mathcal{C}, \mathbf{h})$

**Code for $\mathbf{S}(m)$:**

$b \leftarrow 0$;

**Round 1:**

$\mathtt{erasureCounter} \leftarrow 0$;
**for** $1 \leq i \leq \ell$ **do**
  $\left(y_i^{(1)}, \ldots, y_i^{(t_s+1)}\right) \leftarrow c_i$;
  **if** $y_i^{(j)}$ missing for some $j$ **then**
    $\mathtt{erasureCounter} \leftarrow \mathtt{erasureCounter} + 1$;
  **end if**
**end for**
**if** $\mathtt{erasureCounter} \geq t_s + 1$ **then**
  **return** b;
**end if**
**for** $1 \leq j \leq t_s + 1$ **do**
  $\mathbf{y}^{(j)} \leftarrow \left(y_1^{(j)}, \ldots, y_\ell^{(j)}\right)$;
**end for**
$\mathcal{W} \leftarrow \mathsf{PseudoBasis}_{\mathcal{C}}\left(\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(q)}\right)$;
$\mathbf{y}^{(p)} \leftarrow \left\{\mathbf{y}^{(j)}\right\}_{j=1}^{t_s+1} \setminus \mathcal{W}$; // find vector *not* in the pseudo-basis
$\sigma \leftarrow \mathbf{H}(\mathbf{y}^{(p)})^T$; // the syndrome of $\mathbf{y}^{(p)}$
$pad \leftarrow \mathbf{h}(\mathbf{y}^{(p)})^T$; // the pad to use for encryption

**Round 2, 3:**

$b \leftarrow \Pi_{\mathsf{sRMT}}(\mathcal{W}, s, m + pad)$;
**return** $b$;

**Code for $\mathbf{R}(r_2)$:**

$v \leftarrow \perp$;

**Round 1:**

$\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(t_s+1)} \leftarrow_{\$} \mathcal{C}$;
**for** $1 \leq i \leq \ell$ **do**
  $c_i \leftarrow \left(x_i^{(1)}, \ldots, x_i^{(t_s+1)}\right)$;
**end for**

**Round 2,3:**

$(\mathcal{W}', \sigma', m') \leftarrow \Pi_{\mathsf{sRMT}}()$;
**if** $(\mathcal{W}', \sigma', m') \neq \perp$ **then**
  $p' \leftarrow$ index not in $\mathcal{W}'$;

22: $\mathbf{e}'^{(p')} \leftarrow \mathsf{ComputeErrors}_{\mathcal{C}}\left(\mathcal{W}', \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(t_s+1)}, \sigma', p'\right)$;
23: $\mathbf{y}'^{(p')} = \mathbf{x}^{(p')} + \mathbf{e}'^{(p')}$;
24: $pad' \leftarrow \mathbf{h}(\mathbf{y}'^{(p')})^T$;
25: $v \leftarrow m' - pad'$;
26: **return** $v$;
27: **else**
28: **return** $v$;
29: **end if**

**Lemma 10.** *Assume $t_a + 2t_s < \ell$ and $t_a \leq t_s$. If $(c_1, ..., c_\ell)$ are synchronous channels and at most $t_s$ channels are under control of the adversary, then protocol $\Pi^{t_s,t_a}_{\mathsf{sSMT}}(\mathcal{C}, \mathbf{h})$ achieves $t_s$-security.*

**Lemma 11.** *Assume $t_a + 2t_s < \ell$ and $t_a \leq t_s$. If $(c_1, ..., c_\ell)$ are asynchronous channels and at most $t_a$ channels are under control of the adversary, then protocol $\Pi^{t_s,t_a}_{\mathsf{sSMT}}(\mathcal{C}, \mathbf{h})$ achieves $t_a$-weak correctness and $t_a$-perfect privacy.*

## 6 Conclusions

### 6.1 Putting Things Together

We have investigated the feasibility and optimality of perfectly secure message transmission protocols that achieve security in both synchronous and asynchronous networks. The following corollaries summarize the main results.

**Corollary 1.** *There exists a perfectly secure $\mathsf{SMT}$ protocol that is $t_s$-secure when run over a synchronous network, and $t_a$-secure when run over an asynchronous network if and only if $2t_a + t_s < \ell$.*

**Corollary 2.** *There exists a perfectly secure $\mathsf{SMT}$ protocol that is $t_s$-secure when run over a synchronous network, $t_a$-secure when run over an asynchronous network, and runs in 3 rounds when the network is synchronous, if $t_a + 2t_s < \ell$.*

Using Theorem 6.1 from [1], combined with our $\mathsf{SMT}$ protocol from Corollary 1, we obtain an $n$-party perfectly-secure MPC protocol over networks with $\ell$-connectivity, for any $t_a \leq t_s$ satisfying $3t_s + t_a < n$ and $2t_a + t_s < \ell$.

**Corollary 3 ([1], Theorem 6.1; restated for incomplete networks).** *Let $n$ be the number of parties and $\ell$ be the connectivity of the network. Let $t_a \leq t_s$, such that $3t_s + t_a < n$ and $2t_a + t_s < \ell$. Moreover, let $f : \mathbb{F}^n \rightarrow \mathbb{F}$ be a function represented by an arithmetic circuit over a field $\mathbb{F}$. Then, there is an $n$-party MPC protocol evaluating $f$ over any network with $\ell$ connectivity, such that:*

- *Correctness: (a) When the network is synchronous and there are up to $t_s$ corruptions, all honest parties correctly evaluate the function (with all honest inputs taken into account), and (b) when the network is asynchronous and there are up to $t_a$ corruptions, all honest parties correctly evaluate the function (with $n - t_s$ inputs taken into account).*

– *Privacy: The view of the adversary is independent of the inputs of the honest parties.*

**Acknowledgments.** The authors would like to thank Martin Hirt for some very insightful discussions related to the material in this work.

## References

1. Appan, A., Chandramouli, A., Choudhury, A.: Perfectly- secure synchronous MPC with asynchronous fallback guarantees. In: ACM Symposium on Principles of Distributed Computing (2022)
2. Blum, E., Katz, J., Loss, J.: Synchronous consensus with optimal asynchronous fallback guarantees. In: Hofheinz, D., Rosen, A. (eds.) TCC 2019. LNCS, vol. 11891, pp. 131–150. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36030-6_6
3. Blum, E., Katz, J., Loss, J.: Tardigrade: an atomic broadcast protocol for arbitrary network conditions. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021. LNCS, vol. 13091, pp. 547–572. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-92075-3_19
4. Blum, E., Liu-Zhang, C.-D., Loss, J.: Always have a backup plan: fully secure synchronous MPC with asynchronous fallback. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020. LNCS, vol. 12171, pp. 707–731. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-56880-1_25
5. Choudhury, A., et al.: Secure message transmission in asynchronous networks. J. Parallel Distrib. Comput. **71**, 1067–1074 (2011)
6. Deligios, G., Hirt, M., Liu-Zhang, C.-D.: Round-efficient Byzantine agreement and multi-party computation with asynchronous fallback. In: Nissim, K., Waters, B. (eds.) TCC 2021. LNCS, vol. 13042, pp. 623–653. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-90459-3_21
7. Deligios, G., Liu-Zhang, C.-D.: Synchronous perfectly secure message transmission with optimal asynchronous fallback guarantees. Cryptology ePrint Archive, Paper 2022/1397. https://eprint.iacr.org/2022/1397.2022
8. Desmedt, Y., Wang, Y.: Perfectly secure message transmission revisited. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 502–517. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46035-7_33
9. Dolev, D., et al.: Perfectly secure message transmission. J. ACM **40**(1), 17–47 (1993)
10. Garay, J., Givens, C., Ostrovsky, R.: Secure message transmission by public discussion: a brief survey. In: Chee, Y.M., et al. (eds.) IWCC 2011. LNCS, vol. 6639, pp. 126–141. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20901-7_8
11. Ghinea, D., Liu-Zhang, C.-D., Wattenhofer, R.: Optimal synchronous approximate agreement with asynchronous fallback. In: ACM Symposium on Principles of Distributed Computing. Springer (2022)
12. Kishore, R., Inumella, A., Srinathan, K.: Perfectly secure message transmission over partially synchronous networks. In: ICDCN 2019, Bangalore, India, pp. 302–306. Association for Computing Machinery (2019). isbn: 9781450360944. https://doi.org/10.1145/3288599.3288612

13. Kurosawa, K., Suzuki, K.: Almost secure (1-round, n-channel) message transmission scheme. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **92**(1), 105–112 (2009)
14. Kurosawa, K., Suzuki, K.: Truly efficient 2-round perfectly secure message transmission scheme. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 324–340. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_19
15. Menger, K.: Zur allgemeinen kurventheorie. Fund. Math. **10**, 96–1159 (1927)
16. Momose, A., Ren, L.: Multi-threshold Byzantine fault tolerance. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security, CCS 2021, Virtual Event, Republic of Korea, pp. 1686–1699. Association for Computing Machinery (2021). isbn: 9781450384544. https://doi.org/10.1145/3460120.3484554
17. Resch, N., Yuan, C.: Two-round perfectly secure message transmission with optimal transmission rate. Cryptology ePrint Archive, Report 2021/158 (2021)
18. Md Sayeed, H., Abu-Amara, H.: Efficient perfectly secure message transmission in synchronous networks. Inf. Comput. **126**(1), 53–61 (1996)
19. Spini, G., Zémor, G.: Perfectly secure message transmission in two rounds. In: Hirt, M., Smith, A. (eds.) TCC 2016. LNCS, vol. 9985, pp. 286–304. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53641-4_12
20. Srinathan, K., Narayanan, A., Rangan, C.P.: Optimal perfectly secure message transmission. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 545–561. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28628-8_33