Synchronous Perfectly Secure Message Transmission with Optimal Asynchronous Fallback Guarantees

Giovanni Deligios[1] and Chen-Da Liu-Zhang*[2]

[1] gdeligios@inf.ethz.ch, ETH Zurich
[2] chen-da.liuzhang@ntt-research.com, NTT Research

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 ℓ disjoint paths, of which at most t paths 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 our results, following the recent work by Appan, Chandramouli, and Choudhury [PODC'22], we obtain an n-party perfectly-secure synchronous multi-party computation protocol with asynchronous fallback over any network with connectivity ℓ, 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 ℓ-connected graph there are at least ℓ disjoint paths among any two nodes [14], one often considers the abstraction in which **S** and **R** are simply connected via ℓ channels (also called wires), representing vertex-disjoint paths in the network graph. Assuming an adversary that can corrupt at most t parties in the network, this translates to at most t of the ℓ 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 **S** and **R** from ℓ channels of which an unknown subset of t is under full control of the adversary.

Protocols for 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 SMT can be achieved in the synchronous model if up to $t_s < \ell/2$ wires are corrupted [9, 8, 12], while perfectly secure 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:

* This work was partially carried out while the author was at CMU. 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.

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).

With the recent work 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 ℓ, 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[3] 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 [18, 16].

1.2 Technical Overview

Feasibility. Our feasibility result has three main ingredients:

- 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 ℓ 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.

[3] 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.

– 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. We report a known construction in our notation and prove its security.

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.

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**.

Some of our constructions heavily exploit the linearity of the underlying secret sharing schemes and the fact that errors are always introduced on the same wires. In this setting, the language of error-correcting codes significantly improves the exposition. For consistency, we adopt this language for all of our constructions. It is well known that threshold secret sharing schemes and maximum distance separable codes are essentially equivalent. Lemma 10, for example, can be understood as constructing a secret sharing scheme from an appropriate code. It should therefore not come as a surprise that at many points we talk about secret sharing schemes or certain error-correcting codes interchangeably. More details can be found in [6].

1.3 Related Work

Synchronous Protocols with Asynchronous Fallback. A recent line of works [2, 4, 3, 7, 15, 1, 11] 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, 17, 19, 13, 10, 18]. 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 [18, 16]. 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.

1.4 Outline of the Paper

In Section 2 we introduce our model and a definition for secure message transmission. Section 3 contains the feasibility result for $2t_a + t_s < n$, and Section 4 includes the corresponding tight impossibility result. In Section 5, we show a protocol with concrete round efficiency (when the network is synchronous), under the trade-off $t_a + 2t_s < n$. Finally, Section 6 concludes putting by things together and posing open problems.

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 ℓ-connected if ℓ 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 ℓ is equal to the number of disjoint paths between any two given vertices [14]. 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 Δ. 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 ℓ 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 Π 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 Π is a protocol for 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**[4] **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 Π achieves t-correctness, t-perfect privacy and t-termination, we say that Π is t-perfectly secure.

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. Section 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 ℓ 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 Section 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.

[4] 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.

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

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

1: $b \leftarrow \mathbf{S}_s(m, r_1)$;
2: **if** $b = 1$ **then**
3: **return** b;
4: **else**
5: $b \leftarrow \mathbf{S}_a(m, r_1)$;
6: **return** b;
7: **end if**

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

1: $v \leftarrow \mathbf{R}_s(r_2)$;
2: **if** $v \neq \perp$ **then**
3: **return** v;
4: **else**
5: $v \leftarrow \mathbf{R}_a(r_2)$;
6: **return** v;
7: **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 Π_{sSMT} achieves t_s-security.*

Proof. Let's first argue about correctness. By t_s correctness of $\Pi_{s\mathsf{SMT}}$, $\mathbf{R}_s$ outputs m, and by t_s-termination $\mathbf{R}_s$ terminates. So $\mathbf{R}_h$ outputs m. By the same reasoning, since $\mathbf{S}_s$ outputs 1, $\mathbf{S}_h$ outputs 1 and terminates (and never runs $\mathbf{S}_a$). Since Π_{sSMT} achieves t_s-privacy, and Protocol $\Pi_{a\mathsf{SMT}}$ is not run, t_s-privacy is preserved. To conclude, t_s termination follows from t_s-termination and t_s-correctness of Π_{sSMT}.

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

Proof. Let's first argue about correctness. By t_a-weak correctness of Π_{sSMT} we only need to distinguish two cases. If $\mathbf{R}_s$ outputs $v = m$ then $\mathbf{R}_h$ outputs $b = m$ (even if $\mathbf{S}_h$ might run $\mathbf{S}_a$). In this case, if $\mathbf{S}_s$ outputs $b = 1$, then $\mathbf{S}_h$ outputs $b = 1$ and terminates. Else, it runs $\mathbf{S}_a(m, r_1)$, and by t_a-correctness of Π_{aSMT} it outputs $b = 1$ and terminates. Else, if $\mathbf{R}_s$ output $v = \perp$, then t_a-weak correctness of Π_{sSMT} guarantees that $\mathbf{S}_s$ outputs $b = 0$. This means that $\mathbf{S}_h$ runs $\mathbf{S}_a(m, r_1)$ and $\mathbf{R}_a$ runs $\mathbf{R}_a(r_2)$. Therefore, by t_a-correctness of Π_{aSMT}, the output of $\mathbf{S}_a$ (and therefore $\mathbf{S}_h$) is $b = 1$, while $\mathbf{R}_a$ (and therefore $\mathbf{R}_h$) outputs $v = m$. From t_a-perfect privacy of Π_{sSMT} and Π_{aSMT} directly follows t_a-perfect privacy of Π_{hSMT}. Arguing about termination is straightforward.

3.2 Synchronous RMT with Asynchronous Detection

Before describing our construction for Π_{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 ℓ 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.

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

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

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 $\mathbf{R}()$:

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 Π_{sRMT} achieves t_s-correctness and t_s-termination.*

Proof. Since the channels are synchronous, in Round 1 the receiver $\mathbf{R}$ receives the value m over at least $\ell - t_s > t_s$ distinct channels (and any $m' \neq m$ over at most t_s). This means that $\mathbf{R}$ sets $v = m$ and sends `ok` over all channels. Therefore, in Round 2, the receiver $\mathbf{R}$ outputs $v = m$, while the sender $\mathbf{S}$ receives the message `ok` over at least $\ell - t_s > t_s$ distinct channels, and outputs $b = 1$. Arguing about termination is straightforward, as the protocol is round-based.

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 Π_{sRMT} achieves t_a-weak correctness and t_a-termination.*

Proof. If the receiver outputs $v \neq \perp$ in Round 2, they have received at least $t_s + 1 \geq t_a + 1 > t_a$ messages v over distinct channels in Round 1. This means that at least 1 of these messages came from an honest channel, which means that $v = m$. On the other hand, if the receiver $\mathbf{R}$ outputs $\perp$, there is no message m that they have received over $t_s + 1$ channels in Round 1, and therefore they do not send the message `ok` over any channel to $\mathbf{S}$ in Round 2. Since the adversary can produce at most $t_a \leq t_s < t_s + 1$ messages `ok`, then the sender $\mathbf{S}$ outputs 0 in Round 2. Arguing about termination is straightforward, as the protocol is round-based.

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 (ℓ, 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 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 Δ. 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 10 guarantees that, for all ℓ 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 succesful, then $b = 1$ and $\mathbf{x} \in \mathcal{C}$, otherwise $b = 0$. To ease the notation, we consider that the sRMT protocol runs in 1 round.

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 $2r$:

$\mathbf{y}' \leftarrow \Pi_{\mathsf{sRMT}}()$;
if $\mathbf{y}' = \mathtt{ok}$ **then**
    $e \leftarrow m + \mathbf{h}^{(k)}\mathbf{x}^T$; // one-time-pad encryption
    $b \leftarrow \Pi_{\mathsf{sRMT}}(e)$;
    return b;
end if
if $\mathbf{y}' = \perp$ **then**
    return b;
end if
$p \leftarrow$ smallest index such that $\mathbf{y}' \neq \mathbf{x}$; // find one corrupted channel
$\mathtt{elimChannels} \leftarrow \mathtt{elimChannels} \cup \{p\}$;

Code for $\mathbf{R}()$

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

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

$\mathtt{elimChannels}' \leftarrow \Pi_{\mathsf{sRMT}}()$;
if $\mathtt{elimChannels}' \neq \perp$ **then**
    $k' \leftarrow \#\mathtt{elimChannels}'$;
    for $i \notin \mathtt{elimChannels}'$ **do**
        $y_i \leftarrow c_i$; // only read values on good channels
    end for
    if $y_j \neq \perp$ for all $j \notin \mathtt{elimChanels}'$ **then**
        $(v, \mathbf{x}') \leftarrow \mathsf{decode}_{c_0}(\mathbf{y})$;
    end if
end if

Round $2r$:

if $b' = 1$ **then**
    $e' \leftarrow \Pi_{\mathsf{sRMT}}()$;
    if $e' \neq \perp$ **then**
        $m' \leftarrow e' - \mathbf{h}^{(k')}\mathbf{x}'$; // one-time-pad decryption
        return m';
    end if

```
41:     if e' = ⊥ then
42:         return e'
43:     end if
44: end if
45: if v=1 then
46:     b' ← Π_sRMT(ok);
47: end if
48: if v = 0 then
49:     b' ← Π_sRMT(y); // information to identify corrupted channels
50: end if
```

Lemma 5. *Assume $2t_a+t_s < \ell$, $t_a \le t_s$, 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 $\Pi_{\mathsf{sSMT}}^{t_a,t_s}$ achieves t_s-security.*

Proof. To begin, we argue about t_s-correctness. In any round, by t_s-correctness of sRMT we have $\texttt{elimChannels}' = \texttt{elimChannels}$, so that $k = k'$. Therefore $\mathbf{S}$ and $\mathbf{R}$ agree on the code being used for transmission of the code-word $\mathbf{x}$ and on the set of channels to consider. In any given round, either decoding of the code-word $\mathbf{x}'$ is successful, or $\mathbf{S}$ detects one corrupted channel and adds it to elimChannels. If the adversary does not introduce any errors in the code-word $\mathbf{x}$, then decoding succeeds. If decoding fails (or does not happen at all because a value was erased), then the adversary has introduced at least one error in the code-word $\mathbf{x}$. Hence in Round $r+1$, the sender $\mathbf{S}$ will find at least one index p such that $\mathbf{y}' \neq \mathbf{x}$, and add p to elimChannels. Decoding always succeeds when $k = k' = \#\texttt{elimChannels} = t_s - t_a$. In this case, $\mathrm{d}_{\min}\left(\mathcal{C}^{(k)}\right) = \ell - t_s$, while the number of corrupted channels is at most $t_s - (t_s - t_a) = t_a$. Since by assumption $t_a \le \frac{n-t_s-1}{2}$, then also $t_a \le \left\lfloor \frac{\mathrm{d}_{\min}\left(\mathcal{C}^{(k)}\right)-1}{2} \right\rfloor$. If decoding succeeds, then $\mathbf{x} = \mathbf{x}'$ and (again thanks to t_s-correctness of sRMT) it follows that $m' = e' - \mathbf{h}^{(k')}\mathbf{x}' = e - \mathbf{h}^{(k)}\mathbf{x} = m + \mathbf{h}^{(k)}\mathbf{x} - \mathbf{h}^{(k)}\mathbf{x} = m$ and $b = 1$. To argue about t_s-perfect privacy, we show that in each round the adversary learns no information about the secret pad $\mathbf{h}^{(k)}\mathbf{x}$. This is guaranteed by Lemma 10 if the number of corrupted channels is strictly smaller than $\dim\left(\mathcal{C}^{(k)}\right) = t_s + 1 - k$. Because the number of corrupted channels equals $t_s - k$ for all k, the claim holds. Termination is straightforward as the protocol is round-based.

Lemma 6. *Assume $2t_a + t_s < \ell$, $t_a \le t_s$, 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.*

Proof. We first argue about t_s-correctness. If $\mathbf{R}$ outputs $\perp$, then t_a-weak correctness of Π_{sRMT} guarantees that $\mathbf{S}$ outputs 0. Assume that $\mathbf{R}$ outputs $m' \neq \perp$. Then, by inspection of the protocol, we know that $v = 1$ (which also implies a symbol was received on all channels not in elimChannels), $m' = e' - \mathbf{h}^{(k')}\mathbf{x}'$, and $\mathbf{x}' = \mathsf{decode}_{\mathcal{C}}^{(k')}(\mathbf{y})$. Notice that t_a-weak correctness of Π_{sRMT} guarantees that $k' = k$ and $e = e'$. Since decoding was successful, we know that $\mathbf{x}' = \mathbf{x}$, as the maximum number of errors introduced by the adversary is t_a, while $\mathrm{d}_{\min}\left(\mathcal{C}^{(k)}\right) = \ell - t_s$, so that the assumption $2t_a + t_s < \ell$ guarantees that $t_a \le \left\lfloor \frac{\mathrm{d}_{\min}\left(\mathcal{C}^{(k)}\right)-1}{2} \right\rfloor$. Therefore, $m' = e' - \mathbf{h}^{(k')}\mathbf{x}' = e - \mathbf{h}^{(k)}\mathbf{x} = m + \mathbf{h}^{(k)}\mathbf{x} - \mathbf{h}^{(k)}\mathbf{x} = m$. To argue about t_a-perfect privacy, we show that in each round the adversary learns no information about the secret pad $\mathbf{h}^{(k)}\mathbf{x}$. This is guaranteed by Lemma 10 if the number of corrupted channels is strictly smaller than $\dim\left(\mathcal{C}^{(k)}\right) = t_s + 1 - k$. Observe that at any point in the execution of the protocol $k \le t_s - t_a$. Since the number of corrupted channels is at most t_a, then for all k we have $\dim\left(\mathcal{C}^{(k)}\right) \ge t_s + 1 - (t_s - t_a) \ge t_a + 1 > t_a$, and the claim holds. Termination is straightforward as the protocol is round-based.

3.4 Asynchronous SMT

We present an SMT protocol that is secure when channels $(c_1, \ldots, c_\ell)$ are asynchronous even if up to t_a of them are under control of the adversary. This protocol can be used as the asynchronous protocol Π_{aSMT} in the compiler Π_{hSMT} of Section 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 describe a protocol for completeness.

The idea is simple: the sender secret shares their input with a (ℓ, 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. We describe the protocol using the language of error correcting codes for consistency with other constructions presented. Let $(\mathcal{C}, \mathbf{h})$ be a code and a vector as in Lemma 10, with $t_s = t_a$. As a result, $\mathcal{C}$ is an $(\ell, t_a + 1, 2t_a + 1)$ MDS code. Let $\mathsf{decode}_{\mathcal{C}}(\mathbf{y})$ denote an (efficient) decoding algorithm for $\mathcal{C}$ returning a couple $(b, \mathbf{x})$. If decoding is successful $\mathbf{x}$ is the decoded code-word and $b = 1$, otherwise $b = 0$.

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

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

4: $\mathbf{x} \leftarrow_{\$} \{\mathbf{y} \in \mathcal{C} \,|\, \mathbf{h}\mathbf{y}^T = m\}$;
5: $c_i \leftarrow x_i$;

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

1: `receivedCounter` $\leftarrow 0$;
2: $\mathbf{y} \leftarrow (\perp, \ldots, \perp)$;

Upon receiving a value on channel c_i do

6: $y_i \leftarrow c_i$
7: `receivedCounter` $\leftarrow$ `receivedCounter` $+ 1$;
8: **if** `receivedCounter` $\geq \ell - t_a$ **then**
9: **for** $1 \leq i \leq \ell$ **do**
10: **if** $y_i = \perp$ **then**
11: $y_i \leftarrow 0$;
12: **end if**
13: **end for**
14: $(b, \mathbf{x}') \leftarrow \mathsf{decode}_{\mathcal{C}}(\mathbf{y})$;
15: **if** $b = 1$ **then**
16: $m' \leftarrow \mathbf{h}\mathbf{x}'^T$;
17: **return** m';
18: **end if**
19: **end if**

Lemma 7. *Assume $3t_a < \ell$. If $(c_1, ..., c_\ell)$ are asynchronous channels and at most t_a channels are under control of the adversary, then protocol $\Pi^{t_a}_{\mathsf{aSMT}}(\mathcal{C}, \mathbf{h})$ achieves t_a- security.*

Proof. Let's first argue about t_a-correctness. Assume the receiver $\mathbf{R}$ outputs m'. Then $m' = \mathbf{h}\mathbf{x}'^T$ so that it suffices to show that $\mathbf{x} = \mathbf{x}'$. Since in this case we know $b = 1$, it means the $\mathsf{decode}_{\mathcal{C}}(\mathbf{y})$ terminated successfully. At most $t_a + (\ell - (\ell - t_a))$ symbols of $\mathbf{y}'$ are different than those of $\mathbf{x}$ (at most t_a ones received from channels under control of the adversary and at most $\ell - (\ell - t_a)$ set to 0 by $\mathbf{R}$) and therefore $\mathbf{x}' = \mathbf{x}$, because $\mathrm{d}_{\min}(\mathcal{C}) = 2t_a + 1$. Lemma 10 directly implies t_a-perfect privacy, since $\mathbf{h}\mathbf{x}^T$ is statistically independent from the at most t_a symbols of $\mathbf{x}$ that the adversary learns. Finally, since at least $\ell - t_a$ values travelling on channels not under control of the adversary are eventually delivered to $\mathbf{R}$, the algorithm $\mathsf{decode}_{\mathcal{C}}(\mathbf{y})$ eventually succeeds, because $\mathbf{y}$ contains at most t_a errors and $t_a \leq \lfloor \frac{\mathrm{d}_{\min}(\mathcal{C})-1}{2} \rfloor$. This guarantees t_a-termination.

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 [5, 2, 4], from which we also borrow some notation.

Lemma 8. *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 $t_s + 2t_a \geq \ell$.*

Proof. Let $2t_a + t_s = \ell$, and assume there is a SMT protocol Π that is t_s-perfectly secure when the channels are synchronous and t_a-perfectly secure when the channels are asynchronous. Let K, A, B denote a partition of the set of ℓ channels such that $|A| = |B| = t_a$ and $|K| = t_s$. Consider the following scenarios.

– **Scenario** 1. All channels are synchronous, the sender **S** has input message $m_1 \in \mathbb{F}$ and randomness r_1. The receiver randomness is r_1'. The adversary corrupts the channels in K and simply deletes all messages on these channels. Let α_1 denote the messages travelling on channels A and β_1 denote the messages travelling on channels B in this execution. Denote by T_1 the time it takes for this protocol to terminate (with respect to the receiver's local clock).

By t_s-perfect privacy of Π and because $t_a \leq t_s$, it follows that, for all $m_2 \neq m_1 \in \mathbb{F}$, there exist sender and receiver randomness r_2 and r_2' such that the messages travelling on channels A are exactly α_1.[5] With this observation in mind, consider the following scenarios.

– **Scenario** 2**.** All channels are synchronous and bidirectional, the sender **S** has input message $m_2 \in \mathbb{F}$ and randomness r_2. The receiver randomness is r_2'. The adversary corrupts the channels in K and simply deletes all messages on these channels. Let $\alpha_2 = \alpha$ denote the messages travelling on the wires A, and β_2 denote the messages travelling on the wires B. Denote by T_2 the time it takes for this protocol to terminate (with respect to the receiver's local clock).
– **Scenario** 3**.** All channels are asynchronous and bidirectional, the sender **S** has input $m_1 \in \mathbb{F}$ and randomness r_1, and the receiver **R** has randomness r_2'. The adversary delays all messages travelling on the channels in K by longer than $T_1 + T_2$. Furthermore the adversary corrupts the channels in B, and sends messages according to β_2 from the sender to the receiver, and according to β_1 from the receiver to the sender. Furthermore, the adversary schedules the delivery of messages according to executions of Scenarios 1 and 2.

In Scenario 1, by t_s-correctness of Π, the receiver outputs m_1. By the same reasoning, in Scenario 2, the receiver outputs m_2. In scenario 3, the view of the receiver is identical to that of Scenario 2. However, by t_a-correctness of Π, the receiver outputs m_1, which is a contradiction, since we assumed $m_1 \neq m_2$.

5 Round-Efficient Synchronous SMT with Sub-Optimal Trade-off

Assuming the trade-off $t_a + 2t_s < \ell$, we show a protocol Π_{sSMT} with the properties required for the compiler presented in Section 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. [18, 16]) to the context of security with fallback. Before giving an overview of the protocol, we establish the main technical tools used in the construction. In Section A we provide basic concepts about error-correcting codes needed for our goals.

Lemma 9 ([18]**, Lemma 2).** *Let $\mathcal{C}$ be an (ℓ, 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.

Proof. Consider $\ker(\sigma|_E)$. If $c \in \mathcal{C}$ then $c \in \ker(\sigma)$. Since $\mathbf{0} \in \mathcal{C}$, then for all $\mathbf{0} \neq e \in E$ we have $e \notin \mathcal{C}$, because the minimum Hamming distance of $\mathcal{C}$ is d and $w(e) = \mathrm{d}(e, \mathbf{0}) < d$. Hence, $\ker(\sigma|_E) = \ker(\sigma) \cap E = \{\mathbf{0}\}$.

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 $\mathbb{F}_q^{n-k}$.*

Remark 1. (***Computing Errors from Syndromes and a Pseudo-Basis***) For some $q > \ell - k$ consider the set $\mathcal{X} = \{x^{(1)}, \ldots, x^{(q)}\} \subseteq \mathcal{C}$, and the set $\mathcal{E} = \{e^{(1)}, \ldots, e^{(q)}\} \subseteq \mathbb{F}_q^n$ with the property that $\#\{j : \exists k.\, e_j^{(k)} \neq 0\} = d$ (i.e. errors all introduced at the same d coordinates). Finally, let $\mathcal{Y} = \{y^{(1)}, \ldots, y^{(q)}\}$,

[5] This reasoning does not go through if we allow for an error probability in the protocols. In fact, this impossibility result does not apply to statistically secure message transmission.

where $y^{(k)} = x^{(k)} + e^{(k)}$. Knowing the code-words $\mathcal{X}$ and a pseudo-basis of $\mathcal{Y}$, one can compute the set $\mathcal{E}$, as follows.

Consider the syndromes $\sigma\left(e^{(k)}\right)$ for all $1 \leq k \leq q$. Since $\sigma\left(e^{(k)}\right) = \sigma\left(y^{(k)}\right) \in \langle\sigma(\mathcal{Y})\rangle$, given a pseudo-basis $\mathcal{W} = \left\{y^{(i)}\right\}_{i\in I}$ of $\mathcal{Y}$, for all $1 \leq k \leq q$ there exist (and can be efficiently computed) coefficients $\left\{\lambda_i^{(k)}\right\}_{i\in I} \subset \mathbb{F}_q$ such that

$$\sigma\left(e^{(k)}\right) = \sum_{i\in I} \lambda_i^{(k)} \sigma\left(y^{(i)}\right).$$

From $\sigma\left(y^{(i)}\right) = \sigma\left(x^{(i)} + e^{(i)}\right) = \sigma\left(x^{(i)}\right) + \sigma\left(e^{(i)}\right) = \sigma\left(e^{(i)}\right)$ it follows that

$$\sigma\left(e^{(k)}\right) = \sum_{i\in I} \lambda_i^{(k)} \sigma\left(e^{(i)}\right) = \sigma\left(\sum_{i\in I} \lambda_i^{(k)} e^{(i)}\right).$$

Due to the assumptions made on the error vectors, by Lemma 9 we know that the syndrome map σ is injective when restricted to the set $\mathcal{E}$, which implies

$$e^{(k)} = \sum_{i\in I} \lambda_i^{(k)} e^{(i)}.$$

Remark 1 should already hint to a possible approach. The intuition is the following: the receiver $\mathbf{R}$ picks ℓ 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 RMT together with the syndromes of the errors introduced on code-words that are *not* in the pseudo-basis. Using *Remark* 1, the 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 RMT

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. Some details are provided in Section A.

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 $t_s + t_a$.

Does there exists a code with all the required properties? The following lemma answers this question in the affirmative, under the assumption that $t_a + 2t_s < \ell$. 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 10. *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.*

Proof. Let $\mathcal{C}'$ be an $(\ell+1, t_s+1, \ell-t_s+1)$ Reed-Solomon code. Such a code exists for all choices of ℓ and $t_s \leq \ell$ (assuming a large enough q). Let $\mathcal{C}$ be the code of length ℓ defined as follows: $\mathbf{x} \in \mathcal{C}$ if and only if there exists $y \in \mathbb{F}_q$ such that $(\mathbf{x}, y) \in \mathcal{C}'$. In other words, $\mathcal{C}$ is composed of all code-words of $\mathcal{C}'$ deprived of the last component: we write $\mathbf{x}' \mapsto \mathbf{x}$ for the projection map. Since this map is linear, the code $\mathcal{C}$ is a linear code. If $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{C}$ then $\mathrm{d}(\mathbf{x}_1, \mathbf{x}_2) \geq \mathrm{d}(\mathbf{x}'_1, \mathbf{x}'_2) - 1 \geq \mathrm{d}_{\min}(\mathcal{C}') - 1 \geq \ell - t_s$, so that the minimum distance of $\mathcal{C}$ is $\ell - t_s$. The dimension of $\mathcal{C}$ remains $t_s + 1$: let $\mathbf{b}'_1, \ldots, \mathbf{b}'_{t_s+1}$ be a basis of $\mathcal{C}'$. Suppose that there exist $\mathbb{F}_q \ni \lambda_i$'s such that $\sum_{i=1}^{t_s+1} \lambda_i \mathbf{b}_i = \mathbf{0}$. Then $\sum_{i=1}^{t_s+1} \lambda_i \mathbf{b}'_i \in \mathcal{C}'$ and $d\left(\sum_{i=1}^{t_s+1} \lambda_i \mathbf{b}_i, \mathbf{0}\right) = 1$, which contradicts our assumptions on t_s. Now to the second part of the lemma. For each $\mathbf{x} \in \mathcal{C}$ there is a (unique, because the minimum distance of $\mathcal{C}'$ is greater than one) $x_{n+1} \in \mathbb{F}_q$ such that $(\mathbf{x}, x_{n+1}) = \mathbf{x}' \in \mathcal{C}'$. Assume $(\mathbf{h}, \alpha)$ to be a vector in $\mathcal{C}'^{\perp}$ such that $\alpha \neq 0$, so that $\mathbf{h}\mathbf{x}^T = (\mathbf{h}, \alpha)\mathbf{x}'^T - \alpha x_{n+1} = -\alpha x_{n+1}$. Furthermore, since

the dimension of $\mathcal{C}'$ is t_s+1, any t_s+1 coordinates completely determine a code-word. Let $I \subset \{1,\ldots,\ell\}$ such that $|I| \leq t_s$. For all $a, \{\beta_i\}_{i\in I} \in \mathbb{F}_q$ we have

$$\begin{aligned} &\Pr\left(\mathbf{h}X^T = a \,\middle|\, (X_i)_{i\in I} = (\beta_i)_{i\in I}\right) = \\ &\Pr\left(X_{n+1} = \frac{a}{-\alpha} \,\middle|\, (X_i)_{i\in I} = (\beta_i)_{i\in I}\right) = \\ &\Pr\left(U = \frac{a}{-\alpha}\right). \end{aligned} \tag{1}$$

To find such a vector $(\mathbf{h}, \alpha)$ consider the parity-check matrix $\mathbf{H}$ of $\mathcal{C}'$. Since $\mathcal{C}'$ is an MDS code, there is at least one row of $\mathbf{H}$ of the form $(\mathbf{h}, \alpha)$ with non-zero α.

Remark 2. Lemma 10 guarantees that for all $a \in \mathbb{F}_q$ and for all I with $|I| \leq t$, the probability $\Pr\left(\mathbf{h}\mathbf{x}^T = a \,\middle|\, (X_i)_{i\in I} = (\beta_i)_{i\in I}\right) = 1/|\mathbb{F}|_q$. Then, for all $\mathbf{e}$ with support I, also has

$$\begin{aligned} &\Pr\left(\mathbf{h}(\mathbf{x}+\mathbf{e})^T = a \,\middle|\, (X_i)_{i\in I} = (\beta_i)_{i\in I}\right) = \\ &\Pr\left(\mathbf{h}\mathbf{x}^T = a - \mathbf{h}\mathbf{e}^T \,\middle|\, (X_i)_{i\in I} = (\beta_i)_{i\in I}\right) = 1/|\mathbb{F}|_q. \end{aligned} \tag{2}$$

Intuitively, this means that good randomness to use for one-time-pad encryption of the message can be extracted even from a code-word containing errors.

We now present the protocol. Let $\mathsf{PseudoBasis}_{\mathcal{C}}\left(\mathbf{y}^{(1)},\ldots,\mathbf{y}^{(q)}\right)$ be an algorithm that, given vectors $\mathbf{y}^{(1)},\ldots,\mathbf{y}^{(q)}$ 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 $\sigma = \sigma(\mathbf{e}^{(p)})$ as described in *Remark* 1. Let $\mathcal{C}$ and $\mathbf{h}$ be as in Lemma 10.

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

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

$b \leftarrow 0$;
Round 1:
`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**
    `erasureCounter` $\leftarrow$ `erasureCounter` $+ 1$;
  end if
end for
if `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:

15: $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(t_s+1)} \leftarrow_{\$} \mathcal{C}$;
16: **for** $1 \leq i \leq \ell$ **do**
17: $\quad c_i \leftarrow \left(x_i^{(1)}, \ldots, x_i^{(t_s+1)}\right)$;
18: **end for**
Round 2,3:
19: $(\mathcal{W}', \sigma', m') \leftarrow \Pi_{\mathsf{sRMT}}()$;
20: **if** $(\mathcal{W}', \sigma', m') \neq \perp$ **then**
21: $\quad p' \leftarrow$ index not in $\mathcal{W}'$;
22: $\quad \mathbf{e}'^{(p')} \leftarrow \mathsf{ComputeErrors}_{\mathcal{C}}\left(\mathcal{W}', \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(t_s+1)}, \sigma', p'\right)$;
23: $\quad \mathbf{y}'^{(p')} = \mathbf{x}^{(p')} + \mathbf{e}'^{(p')}$;
24: $\quad pad' \leftarrow \mathbf{h}(\mathbf{y}'^{(p')})^T$;
25: $\quad v \leftarrow m' - pad'$;
26: $\quad$ **return** v;
27: **else**
28: $\quad$ **return** v;
29: **end if**

Lemma 11. *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.*

Proof. Let's first argue about t_s-correctness. To begin, notice that the adversary can erase values on at most t_s channels, which means that $\texttt{erasureCounter} \leq t_s$ for the entire duration of the protocol. Since the channels c_i's are synchronous and at most t_s are under control of the adversary, by t_s-correctness of protocol $\Pi^{t_s}_{\mathsf{sRMT}}$ the sender $\mathbf{S}$ sets $b = 1$, outputs $b = 1$, and terminates. In Round 1, the adversary can introduce at most t_s errors in each code-word sent by $\mathbf{R}$, and errors are introduced at the same subset of at most t_s coordinates for *all* code-words. If $\mathbf{e}^{(j)}$ denotes the error introduced by the adversary on code-word $\mathbf{x}^{(j)}$, then $w(\mathbf{e}^{(j)}) \leq t_s < \ell - t_s = \mathrm{d}_{\min}(\mathcal{C})$, so that the assumptions of Lemma 9 and *Remark* 1 are satisfied. Again, by t_s-correctness of protocol $\Pi^{t_s}_{\mathsf{sRMT}}$, we know that $(\mathcal{W}', \sigma', m') = (\mathcal{W}, \sigma, m + pad) \neq \perp$; These facts combined guarantee that $p' = p$ and $\mathbf{e}'^{(p')}$ is the actual error introduced by the adversary during transmission of $\mathbf{x}^{(p)}$, so that $\mathbf{y}'^{(p')} = \mathbf{y}^{(p)}$, resulting in $pad' = pad$. In conclusion, the receiver $\mathbf{R}$ outputs $v = m' - pad' = m + pad - pad' = m$ and terminates in Round 2;

Let's now argue about t_s-perfect privacy. Clearly, the messages sent in Round 1 by $\mathbf{R}$ are independent of $\mathbf{S}$'s input m. Let's restrict our attention the the messages sent in Round 2 by $\mathbf{S}$. Here, the same message is sent through each channel (via $\Pi^{t_s}_{\mathsf{sRMT}}$), so that we can simply consider one such message $(\mathcal{W}, \sigma, m + pad)$. Since the $\mathbf{x}^{(i)}$'s are independent, the distributions of the pseudo-basis $\mathcal{W}$ and $\mathbf{x}^{(p)}$ are independent. Furthermore, the adversary could compute σ on its own, since $\sigma = \mathbf{H}(\mathbf{y}^{(p)}) = \mathbf{H}(\mathbf{e}^{(p)})$. To conclude, Lemma 10 and *Remark* 2 guarantees that the distribution of *pad* is uniformly random despite knowing up to t_s components of $\mathbf{x}^{(p)}$. This also means that $m + pad$ is uniformly random.

Arguing about termination is straight-forward, as the protocol is round-based.

Lemma 12. *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.*

Proof. Let's first argue about t_a-weak correctness. If $\texttt{erasureCounter} \geq t_s$, the sender $\mathbf{S}$ does not participate in the Π_{sRMT} subprotocol and the receiver $\mathbf{R}$ outputs $\perp$. Assume that receiver $\mathbf{R}$ outputs $\perp$. Then, by t_a-weak correctness of $\Pi^{t_s}_{\mathsf{sRMT}}$, the sender S outputs 0. On the other hand, if $\mathbf{R}$ does not output $\perp$, and by t_a-weak correctness of $\Pi^{t_s}_{\mathsf{sRMT}}$ this means that $\mathbf{R}$ sets $(\mathcal{W}', \sigma', m') = (\mathcal{W}, \sigma, m + pad)$. Notice that in this case the sender's $\texttt{erasureCounter} \leq t_s$. If $\mathbf{e}^{(j)}$ denotes the error introduced by the adversary on code-word $\mathbf{x}^{(j)}$, then $w(\mathbf{e}^{(j)}) \leq t_s + t_a < \ell - t_s = \mathrm{d}_{\min}(\mathcal{C})$, so that the assumptions of Lemma 9 and *Remark* 1 are satisfied. Then, reasoning as in the proof of Lemma 11 we can conclude that $\mathbf{R}$ outputs m. The t_a-perfect privacy property follows from$t_a \leq t_s$ reasoning as in the proof of Lemma 11. Termination straightforward, as the protocol is round-based.

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* 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* 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 SMT protocol from Corollary 1, we obtain an n-party perfectly-secure MPC protocol over networks with ℓ-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 ℓ 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 $f : \mathbb{F}^n \to \mathbb{F}$ 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 ℓ 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.*

6.2 Open Problems

Our work leaves several interesting research directions. For example, an interesting efficiency problem is whether under the optimal trade-off assumptions there exist protocols for perfectly secure message transmission that match the optimal round complexity of purely synchronous protocols. Another exciting direction is exploring whether our techniques can be employed to achieve security for synchronous protocols with asynchronous fallback in incomplete networks for other complex tasks such as distributed key generation, parallel broadcast, state-machine replication, etc.

6.3 Acknowledgments

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

References

[1] Ananya Appan, Anirudh Chandramouli, and Ashish Choudhury. "Perfectly-Secure Synchronous MPC with Asynchronous Fallback Guarantees". In: *ACM Symposium on Principles of Distributed Computing* (2022).

[2] Erica Blum, Jonathan Katz, and Julian Loss. "Synchronous Consensus with Optimal Asynchronous Fallback Guarantees". In: Nov. 2019, pp. 131–150.

[3] Erica Blum, Jonathan Katz, and Julian Loss. "Tardigrade: An Atomic Broadcast Protocol for Arbitrary Network Conditions". In: *Advances in Cryptology – ASIACRYPT 2021*. Ed. by Mehdi Tibouchi and Huaxiong Wang. Cham: Springer International Publishing, 2021, pp. 547–572. ISBN: 978-3-030-92075-3.

[4] Erica Blum, Chen-Da Liu Zhang, and Julian Loss. "Always Have a Backup Plan: Fully Secure Synchronous MPC with Asynchronous Fallback". In: *Advances in Cryptology – CRYPTO 2020*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12171. LNCS. Springer, Aug. 2020.

[5] Ashish Choudhury et al. “Secure message transmission in asynchronous networks”. In: *J. Parallel Distrib. Comput.* 71 (Aug. 2011), pp. 1067–1074.
[6] Ronald Cramer, Ivan Bjerre Damgård, and Jesper Buus Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015.
[7] Giovanni Deligios, Martin Hirt, and Chen-Da Liu-Zhang. “Round-efficient byzantine agreement and multi-party computation with asynchronous fallback”. In: *Theory of Cryptography Conference*. Springer. 2021, pp. 623–653.
[8] Yvo Desmedt and Yongge Wang. “Perfectly secure message transmission revisited”. In: *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2002, pp. 502–517.
[9] Danny Dolev et al. “Perfectly Secure Message Transmission”. In: *J. ACM* 40.1 (Jan. 1993), pp. 17–47.
[10] Juan Garay, Clint Givens, and Rafail Ostrovsky. “Secure message transmission by public discussion: A brief survey”. In: *International Conference on Coding and Cryptology*. Springer. 2011, pp. 126–141.
[11] Diana Ghinea, Chen-Da Liu-Zhang, and Roger Wattenhofer. “Optimal Synchronous Approximate Agreement with Asynchronous Fallback”. In: *ACM Symposium on Principles of Distributed Computing*. Springer. 2022.
[12] Kaoru Kurosawa and Kazuhiro Suzuki. “Almost secure (1-round, n-channel) message transmission scheme”. In: *IEICE transactions on fundamentals of electronics, communications and computer sciences* 92.1 (2009), pp. 105–112.
[13] Kaoru Kurosawa and Kazuhiro Suzuki. “Truly efficient 2-round perfectly secure message transmission scheme”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2008, pp. 324–340.
[14] Karl Menger. “Zur allgemeinen kurventheorie”. In: *Fund. Math.* 10 (1927), pp. 96–1159.
[15] Atsuki Momose and Ling Ren. “Multi-Threshold Byzantine Fault Tolerance”. In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’21. Virtual Event, Republic of Korea: Association for Computing Machinery, 2021, pp. 1686–1699. ISBN: 9781450384544. DOI: 10.1145/3460120.3484554. URL: https://doi.org/10.1145/3460120.3484554.
[16] Nicolas Resch and Chen Yuan. *Two-Round Perfectly Secure Message Transmission with Optimal Transmission Rate*. Cryptology ePrint Archive, Report 2021/158. 2021.
[17] Hasan Md Sayeed and Hosame Abu-Amara. “Efficient perfectly secure message transmission in synchronous networks”. In: *Information and Computation* 126.1 (1996), pp. 53–61.
[18] Gabriele Spini and Gilles Zémor. “Perfectly Secure Message Transmission in Two Rounds”. In: *Proceedings, Part I, of the 14th International Conference on Theory of Cryptography - Volume 9985*. Berlin, Heidelberg: Springer-Verlag, 2016, pp. 286–304. ISBN: 9783662536407.
[19] K Srinathan, Arvind Narayanan, and C Pandu Rangan. “Optimal perfectly secure message transmission”. In: *Annual International Cryptology Conference*. Springer. 2004, pp. 545–561.

A Error-Correcting Codes

Let $\mathbb{F}_q$ be a finite field. Let $\mathbb{F}_q^\ell$ denote the ℓ-dimensional vector space over $\mathbb{F}_q$.

Definition 3. *An (ℓ, k)-linear code $\mathcal{C}$ over $\mathbb{F}_q$ is a k-dimensional linear subspace of $\mathbb{F}_q^\ell$.*

We denote elements of $\mathcal{C}$ (also called *code-words*) with bold-face lower-case letters ($\mathbf{x}, \mathbf{y}, \mathbf{z}, \dots$), and think of them as vectors of length ℓ with entries in $\mathbb{F}_q$ (by fixing a basis of $\mathbb{F}_q^\ell$, typically the canonical one). A linear code $\mathcal{C}$ is uniquely determined by its *generator matrix* $\mathbf{G}$. This is simply the matrix representing the linear bijection $\mathbb{F}_q^k \to \mathcal{C}$. One can also consider linear maps . Alternatively, a code $\mathcal{C}$ is uniquely determined by its *parity-check matrix* $\mathbf{H}$, that is the matrix representing the unique linear map $\mathbb{F}_q^\ell \to \mathbb{F}_q^{\ell-k}$ whose kernel is exactly $\mathcal{C}$.

Definition 4. *Let $\mathbf{x}, \mathbf{y} \in \mathcal{C}$. The Hamming distance between $\mathbf{x}$ and $\mathbf{y}$, denoted by $\mathrm{d}(\mathbf{x}, \mathbf{y})$, is the number of entries in which the two vectors differ, that is $\mathrm{d}(\mathbf{x}, \mathbf{y}) = \#\{j \in [\ell] : x_j \neq y_j\}$.*

Definition 5. *The Hamming weight of a code-word $\mathbf{x}$, denoted by $\mathrm{w}(\mathbf{x})$, is simply $\mathrm{d}(\mathbf{x}, 0)$.*

Definition 6. *The minimum distance of $\mathcal{C}$, denoted by $\mathrm{d}_{\min}(\mathcal{C})$ is the minimum among all Hamming distances between elements of $\mathcal{C}$, that is $\mathrm{d}_{\min}(\mathcal{C}) = \min_{\mathbf{x},\mathbf{y}\in\mathcal{C}} \mathrm{d}(\mathbf{x}, \mathbf{y})$.*

It is easy to show that $\mathrm{d}_{\min}(\mathcal{C})$ is equal to the minimum Hamming weight among code-words of $\mathcal{C}$.

Lemma 13. (Singleton Bound) *Let $\mathcal{C}$ be an (ℓ, k)-linear code over $\mathbb{F}_q$. Then $\mathrm{d}_{\min}(\mathcal{C}) + k \leq \ell + 1$.*

Definition 7. *Let $\mathcal{C}$ be an (ℓ, k)-linear code over $\mathbb{F}_q$. If $\mathrm{d}_{\min}(\mathcal{C}) + k = \ell + 1$ then we say $\mathcal{C}$ is a maximum distance separable (*MDS*) code.*

Even though it is not needed, we often make the minimum distance of an (ℓ, k)-linear MDS code explicit, and we talk about (ℓ, k, d)-linear MDS codes.

Intuitively, as long as the number of errors introduced in a code-word $\mathbf{x}$ (changes in the coordinates of $\mathbf{x}$ with respect to a fixed basis) is less than $(\mathrm{d}_{\min}(\mathcal{C}) - 1)/2$, then $\mathbf{x}$ can be recovered uniquely. We refer to this process as *decoding*. If the number of errors exceeds this threshold, then, as long as it is less than the minimum distance, the fact that an error has occurred can at least be detected. A class of MDS codes for which efficient decoding algorithms exist are Reed-Solomon codes. Threshold secret sharing schemes and MDS codes are essentially equivalent, so that talking in terms of one or the other is mostly a choice of language. For more details, we refer to [6].