Universally Composable Almost-Everywhere Secure Computation

Nishanth Chandran[1], Pouyan Forghani[2], Juan Garay[2], Rafail Ostrovsky[3], Rutvik Patel[2], and Vassilis Zikas[4]

[1] Microsoft Research
nichandr@microsoft.com
[2] Texas A&M University
{pouyan.forghani,garay,rsp7}@tamu.edu
[3] UCLA
rafail@cs.ucla.edu
[4] Purdue University
vzikas@cs.purdue.edu

Abstract. Most existing work on secure multi-party computation (MPC) ignores a key idiosyncrasy of modern communication networks, that there are a limited number of communication paths between any two nodes, many of which might even be corrupted. The problem becomes particularly acute in the information-theoretic setting, where the lack of trusted setups (and the cryptographic primitives they enable) makes communication over sparse networks more challenging. The work by Garay and Ostrovsky [EUROCRYPT'08] on *almost-everywhere MPC* (AE-MPC), introduced "best-possible security" properties for MPC over such incomplete networks, where necessarily some of the honest parties may be excluded from the computation.

In this work, we provide a universally composable definition of *almost-everywhere security*, which allows us to automatically and accurately capture the guarantees of AE-MPC (as well as AE-communication, the analogous "best-possible security" version of secure communication) in the Universal Composability (UC) framework of Canetti. Our results offer the first simulation-based treatment of this important but under-investigated problem, along with the first simulation-based proof of AE-MPC. To achieve that goal, we state and prove a general composition theorem, which makes precise the level or "quality" of AE-security that is obtained when a protocol's hybrids are replaced with almost-everywhere components.

1 Introduction

Secure multi-party computation (MPC) allows n parties communicating over a network to compute a function on their private inputs so that an adversary corrupting some of the parties can neither disrupt the computation (correctness) nor learn more than (what can be inferred from) the output of the function being computed (privacy).

Despite great progress on the problem since it was first introduced and proven feasible [Yao82,GMW87,BGW88,CCD88] involving hundreds, if not thousands, of published results in cryptography and security, and, more recently, even implemented systems, the overwhelming majority of the solutions assume a *complete* communication network of either authenticated (aka reliable) or secure (both authenticated and private) point-to-point channels. In fact with only a few exceptions, discussed below, this is the case for both practical and theoretical works on MPC, and in particular for works on composable security of MPC—indeed, the latter almost exclusively assume a network that cannot be disconnected by the adversary. This creates a disconnect (pun intended) between the vast MPC literature and modern *ad-hoc* networks, such as the Internet, where the communication might be occurring over an incomplete communication graph with the nodes being routing nodes, that might themselves be corrupted, and/or might even be part of the participating MPC nodes themselves.

At first approximation, there are two situations that might present themselves in such an incomplete network: Either the adversary is able to disconnect the communication graph—by corrupting nodes whose edges are in cuts of the graph—or not. In the former case, it is known that if the parties do not share an authentication-enabling setup, such as a PKI, then the best that can be achieved is the so-called *secure computation without authentication* [BCL+11]: The adversary is able to break down the player set into connected components, so that parties in different connected components compute different instances of the function with inputs from the component—and all other inputs chosen by the adversary, and potentially different for each component. Even this weak form of security is only achievable for computationally bounded adversaries; if one is after information-theoretic (aka unconditional) security, where the adversary is unbounded, then the above guarantee is too much to ask for.

Notwithstanding, even in the latter case, where the adversary cannot disconnect the network, the situation is trickier than one might expect. Indeed, if a PKI-like setup is not assumed[1] then it is known that secure communication between any two parties requires the existence of $O(n)$ paths among them (known to or discoverable by the receiver), the majority of which must remain uncorrupted. This is the well-known *secure message transmission* (SMT) problem [DDWY90]. The result holds even for the *reliable message transmission* (RMT) problem, in which only correctness is required.

The above leads to the following natural question: What is the "best-possible" MPC security we can obtain in such a situation where SMT cannot be in general guaranteed? Towards answering this question, Garay and Ostrovsky [GO08] introduced the properties of so-called *almost-everywhere MPC* (AE-MPC), which extended the concept of AE reliable communication previously studied by Dwork, Peleg, Pippenger, and Upfal [DPPU86]. In a nutshell, this paradigm, which we will refer to as *almost-everywhere security* (AE-security for short) recognizes that when even all-to-all SMT is not possible, then, inevitably, there will be honest (uncorrupted) parties for which we are unable to offer the security guarantees that honest parties enjoy in MPC (i.e., privacy, correctness, etc). The core mission of such protocols is then to minimize the number of such left-out (aka *doomed*) parties in an AE-secure construction, while tolerating the maximum number of corruptions.

However, despite a number of elegant combinatorial arguments to achieve the above goal, the security definition used by these constructions has not caught up with the state of the art in MPC security. In particular, to the best of our knowledge, there exists no simulation-based treatment of AE-security. This means that one cannot directly compose the elegant constructions of AE-secure primitives into a higher level protocol. For example, one would hope to be able to prove that running a standard MPC protocol over an AE-SMT network yields an AE-MPC protocol which does not leave more doomed parties than the underlying AE-SMT construction. Given the state of the art, such a modular statement would be impossible, and one would need to prove AE-MPC security from scratch. Instead, a simulation-based treatment in one of the composable security frameworks would inherit a modular composition theorem making such statements tractable and simpler.

This work's main goal is to derive such a treatment in the Universal Composability (UC) framework of Canetti [Can01]. A major challenge, which we tackle, is to obtain a generic definition of AE-security which can be applied to any type of functionality and captures both AE-communication and AE-computation, two primitives whose treatment has been very different. In fact, we achieve this goal by introducing a generic, composition-preserving transformation from a secure variant of a functionality to its AE-secure counterpart. We show that the derived AE-secure functionalities for secure communication (AE-RMT and AE-SMT) and for secure MPC (AE-MPC): (1) preserve all the desired properties of the previous definitions, and (2) are securely realized by (straightforward UC adaptations of) the corresponding AE-secure protocols. The fact that our treatment preserves composability of the (AE-)security statements allows us to derive, as a simple corollary, the first simulation-based proof of AE-MPC. Next, before providing more details on our results, we provide some necessary literature background that should help the reader appreciate the relevance of our contributions and the challenges associated with them.

In passing, we note that although we adopt the language of UC in our treatment, our definitional framework is generic and can be applied to any of the main-stream composable security frameworks for cryptographic protocols [BPW03,CDPW07,MR11,HS15,CKKR19,BCH+20].

1.1 Related Work

The origins of the "almost-everywhere" (AE) notion can be traced back to the work of Dwork *et al.* [DPPU86], who considered the task of Byzantine agreement [PSL80,LSP82] over sparse communication networks. In such

[1] A PKI setup makes the problem trivial in this case as a complete graph can be trivially built by gossip (i.e., flooding) of signed messages.

networks, correctness cannot be guaranteed for all honest parties, since for example the adversary can cut a node off from the rest of the network by corrupting all of its neighbors. Thus, some honest parties must be given up, and correctness is guaranteed only almost everywhere, i.e., only for the remaining honest parties. The AE notion can be applied to other distributing computing tasks as well: Given a set of parties P of size n and an adversary who corrupts $T \subseteq P$, the parties in some set $D \subseteq P - T$ (D for "doomed") are considered abandoned and the correctness conditions of the task are only guaranteed for the parties in $W = P - T - D$, which are called "privileged." Note that both D and W are functions of T as well as of the underlying protocol and graph. The number of doomed parties thus becomes another parameter to the problem, and the goal is to construct a low-degree network (ideally of constant degree) admitting a protocol that tolerates a large number t of corruptions (ideally, a constant fraction) while dooming as few nodes as possible (ideally $O(t)$ for constant-degree networks).

Returning to the problem of Byzantine agreement, Dolev [Dol81] showed that it requires connectivity at least $2t + 1$ to solve, which implies that every node in the network must have degree $\Omega(t)$. Given this high connectivity requirement, Dwork *et al.* [DPPU86] proposed the notion of AE agreement, in which the agreement and validity properties are guaranteed only for the privileged parties. They showed how to simulate, over an incomplete network, an agreement protocol designed for a complete network by replacing the point-to-point communication with a transmission scheme that works over multiple paths between any two nodes. Thus, they reduced the problem of AE agreement to the problem of AE reliable message transmission (RMT), which guarantees that any two privileged nodes can communicate perfectly reliably.

Dwork *et al.* gave a number of constructions achieving AE-RMT with various combinations of parameters; the two most important are a constant-degree graph admitting an AE-RMT scheme tolerating $t = O(n/\log n)$ corruptions while dooming $O(t)$ nodes, and a graph of degree n^ϵ (for any $0 < \epsilon < 1$) admitting an AE-RMT scheme tolerating $t = O(n)$ corruptions while dooming $O(t)$ nodes. Several follow-up works have obtained improved parameters for AE-RMT (and thus also for AE agreement). Upfal [Upf92] gave a transmission scheme tolerating $t = O(n)$ corruptions and only dooming $O(t)$ nodes in a network of constant degree, which is the optimal set of parameters, but at the expense of an exponential-time protocol. Chandran *et al.* [CGO10] proposed a scheme tolerating $t = O(n)$ corruptions and dooming $O(t/\log n)$ nodes in a network of polylogarithmic degree. Most recently, Jayanti *et al.* [JRV20] used the probabilistic method to show the existence of a logarithmic-degree graph admitting an AE-RMT scheme with the same parameters, thereby strictly improving the [CGO10] result.

Due to the results in [Dol81,DDWY90], standard MPC (guaranteeing correctness and privacy for all honest parties) is possible only in networks with connectivity at least $2t + 1$. To circumvent this high-connectivity requirement and still obtain a meaningful notion of (property-based) MPC over sparse networks, Garay and Ostrovsky [GO08] introduced the notion of AE-MPC, which guarantees correctness and privacy only for the privileged parties.[2] "Regular" information-theoretic MPC (i.e., MPC over a complete network) requires $t < n/3$ [BGW88,CCD88]. In the AE setting, the effect of dooming nodes is equivalent to letting the adversary corrupt some additional t' nodes (which are doomed) by requesting the corruption of t nodes (which are actually corrupted). As shown by Garay and Ostrovsky, AE-MPC in the information-theoretic setting can be achieved when $t + t' < n/3$. Their approach resembles that of Dwork *et al.* [DPPU86] for simulating a protocol meant for a complete network, but there is an additional challenge given the privacy requirement of MPC. To replace point-to-point secure channels, they introduced a new model for the existing (perfectly) SMT problem termed *secure message transmission by public discussion* (SMT-PD), which we now turn to.

The original SMT problem [DDWY90] considers two honest parties, a sender S and a receiver R, connected by n disjoint "wires" and sharing no information. The task is for S to send a message to R in the presence of a computationally unbounded adversary $\mathcal{A}$ who can adaptively corrupt up to t of the wires. SMT requires that the message be conveyed perfectly reliably to R, and also that no information about the message leaks to $\mathcal{A}$. We refer to the simpler problem in which there is no secrecy guarantee as *reliable message transmission* (RMT), and it should be clear that this is consistent with our usage of the term AE-RMT above. While RMT can be achieved for $t < n/2$ by simply sending the message over all wires, Dolev

[2] Technically, they considered the related task of *secure function evaluation* (SFE). We do the same, although for consistency we still refer to the functionality that we realize as AE-MPC.

et al. [DDWY90] showed that SMT is also possible if and only if $t < n/2$. We give a more detailed history of the SMT literature in Appendix A.

Returning to the work by Garay and Ostrovsky [GO08], the SMT-PD model overcomes the necessity of $2t + 1$ wires in SMT by in addition allowing access to an authentic and reliable public channel. Given such a channel (which can be constructed using, e.g., a broadcast protocol), they gave a protocol that is secure as long as at least one of the wires remains honest, at the cost of a small error. To use their SMT-PD protocol over sparse networks (in effect achieving AE-SMT), the wires are replaced by multiple paths between a pair of nodes and the public channel is replaced by AE broadcast. Garay and Ostrovsky provided a way to construct graphs that admit SMT-PD from any of the networks in the AE agreement literature, with asymptotically preserved parameters. Finally, they showed how to "compile" a standard information-theoretic MPC protocol into an AE-MPC protocol over any such graph so that the protocol gives up (i.e., considers as doomed) the same number of parties as the underlying (AE-secure) communication network.

To reiterate, all the above constructions are shown secure in a property-based manner. In Appendix A, we review other related notions from the literature, which do not quite consider AE-security.

1.2 Overview of Our Results

In this work we put forth the first composable (simulation-based) definition and treatment of AE-security. In particular, we devise a definition in Canetti's UC framework [Can01] and prove that the (UC adaptation of) existing AE-secure communication/computation protocols achieve this definition. We emphasize that all of our constructions tolerate adaptive corruptions.

There are several challenges associated with such a task. First, as should be evident from the above discussion, the related literature—from RMT/SMT, to Byzantine agreement, to MPC, and even their AE counterparts—treats the underlying network in different ways: e.g., in MPC, the network is typically a complete graph of point-to-point channels (one per pair), whereas the literature on (AE-)RMT assumes multiple paths (wires or indirect paths) between two parties (Sender and Receiver). Thus, in order to derive a formulation that is general enough to capture the security of the above constructions, one first needs to develop a unified approach to them. Towards this goal, we adopt the graph model as a basis for all these protocols, and express the wires for the AE-secure communication literature as a simple graph which for each wire includes a path going through a unique "wire-party." This allows us to model corrupted wires as standard (party) corruptions in UC.

The second, and more thorny challenge is regarding the (simulation of) doomed parties. Recall that those are parties that due to their poor connectivity (which might be the result of the sparsity of the graph and the corruption choices of the adversary) cannot enjoy the security guarantees that the protocol is designed to offer to honest parties (e.g., correctness and privacy for an MPC protocol). A strawman approach would be to capture those parties plainly as corrupted. This, however, is problematic in several ways: First, corrupted parties lose their security guarantees as soon as they become corrupted, unlike doomed parties who might, at the adversary's discretion, still be allowed some level of security. In particular, the real-world adversary might allow those parties to receive their outputs, which would mean that in the ideal world, the simulator would also need to allow them to produce an output on their output tape, which is not allowed by the UC corruption mechanism.

An attempt to fix the above issue would be to define weaker corruption types corresponding to the flexible guarantees offered to the doomed parties. This, however, is also problematic, as corruptions in UC are by default known to (and declared by) the adversary/environment, whereas the actual identities of doomed parties are not, and depend on the behavior of the adversary (not just the identities of malicious parties). In particular, an adversary following for example a random strategy might not even be aware who is becoming doomed by this strategy.

A third attempt would be to completely change the corruption mechanism of UC so that certain corruptions are not to be declared by the environment. But this would immediately invalidate the composition theorem, which defeats the purpose of using UC in the first place.

It might seem like we are in a deadlock, but the second attempt above is the one that breaks through. In particular, we observe that although the adversary might not include in its view the identities of the doomed parties, still its behavior defines these identities and the corresponding guarantees they receive. This is similar to how inputs of corrupted parties are treated in standard UC security: It is the job of the simulator to extract them from the adversary and hand them over to the functionality.

Using the above idea, instead of modifying the foundations of UC, we define a class of functionalities which take from their adversary (simulator) as an input requests to turn parties as doomed, and upon such requests, allows the simulator to use these parties as if they were corrupted, but without declaring them as corrupted to the framework and without grounding their input/output tapes (e.g., the simulator might still instruct this new functionality to deliver output for doomed parties). In fact, this is done in a way which uses the underlying (non-AE) functionality as a black-box; in other words, by means of an *AE-security wrapper*.

In more detail, in order to properly use the functionality, our AE wrapper builds the entire infrastructure of UC around it (including a fake corruption directory), and whenever a doom request comes in, the wrapper pretends towards its wrapped functionality to be an adversary that corrupts this party. This way, the party remains honest as far as the UC experiment is concerned, but the wrapper has now the ability to give full control over this party to the actual simulator it interacts with.

The final piece of the puzzle is capturing the different assumptions on the ratios of corrupted vs doomed parties while making a composable statement. Here we use an idea inspired by [BMTZ17]: We parameterize the wrapper by the set of all allowable corruption/doom patterns, and make sure that any corruption outside this allowable set is ignored. As an example, if we want to prove security of AE-MPC with $t < \alpha n$ corruptions and $d < \beta n$ doomed parties, we can do it by parameterizing the wrapper with the pair (α, β) and ignoring requests of simulators which do not respect the above requirements.

In fact, to allow for the tightest possible results that accurately translate non-threshold corruption/doom patterns—these are the types of results we get by using structural properties of the underlying graph—we draw inspiration from the mixed general adversary literature [HM97,BTFH$^+$08]. Concretely, we parameterize the wrapper with a corruption/doom structure ("doom structure" for short) which consists of all allowed pairs (C, D) where parties in D can be doomed simultaneously to parties in C being corrupted. We note that, as is common in the general adversary literature, such a structure might be exponentially large. Although this is not an issue in our definition, we note that all of our concrete instantiations consider structures that have a polynomial (in n) representation.

We then apply our definitional framework to capture known AE-secure constructions, as well as (simulation-based) AE-MPC. Next, we describe our results in greater detail.

Almost-everywhere RMT and SMT. We start in Section 3 by modelling the tasks of RMT and SMT (with a dedicated sender and receiver connected by a number of corruptible wires). As part of this, we show how primitives like secure message transmission [DDWY90], which have classically only been considered for an honest majority of wires, can be captured so that their security is defined independently of the number of corrupted wires. This seemingly simple task already has complications when ported to a composable framework like UC: the wires cannot be viewed as reliable/secure message transmission functionalities, since UC functionalities are by default incorruptible. We cast the problem so that we can apply a unified treatment: We model each wire with a dummy party called a "wire-party" that is connected to the sender and receiver and relays messages between them. Wire corruptions can now be modelled by corrupting wire-parties.

In Section 3.1, we confirm that classical RMT/SMT protocols [DDWY90] are UC-secure (in the ordinary, non-AE sense) in our model against corrupted minorities of wire-parties. To handle corrupted majorities (and more generally to capture AE-security), in Section 3.2 we introduce an *AE wrapper functionality* (Figure 8) that is parameterized by a doom structure as defined above. The wrapper accepts requests to doom parties from the simulator according to the doom structure and the current set of corruptions, and it pretends to the underlying functionality that those parties are actually corrupted. We are then able to state the security of RMT/SMT protocols independently of the number of corrupted wires, by using the AE wrapper parameterized by a simple doom structure like the one that allows dooming the sender or receiver when a majority of the wire-parties are corrupted. We finish up in Section 3.3 with a universally composable treatment of the SMT-PD problem [GO08]. In addition to simple wires, SMT-PD requires access to a public channel, which we model with access to an authenticated channel functionality (the same functionality we used to capture RMT security). Looking ahead, we need SMT-PD when we want to elevate RMT to SMT over some classes of sparse graphs (like those in [Upf92,CGO10,JRV20]) that do not rely on paths with an honest majority to obtain reliable message transmission. However, for other sparse graphs, like those in [DPPU86], the technique from [DDWY90] suffices to achieve SMT.

Almost-everywhere *remote* RMT and SMT. In Section 4, we move on to the more complicated case in which an incomplete graph connects several parties and all-to-all communication is desired among them. Interestingly, we show that the same wrapper from Section 3.2, which allowed for the simulation-based treatment of tasks like RMT and SMT (with dedicated sender and receiver) even against corrupted majorities of wires, can also be used to model AE-security of the all-to-all versions of those tasks over an incomplete graph. In particular, in Section 4.1 we use the same ideal functionalities and wrapper (of course, with more complex doom structures) from Section 3 to provide the first universally composable treatment of (AE) reliable communication over the specially constructed sparse graphs in [DPPU86,Upf92,CGO10,JRV20], which we refer to as AE *remote* RMT. Using the same wrapper, we also extend our treatment to AE *remote* SMT for all of these graphs. First, we show that a perfect SMT protocol from [DDWY90] can be adapted to realize perfectly secure AE-SMT over a class of sparse graphs constructed in [DPPU86]. In general, the same approach cannot be directly extended to achieve privacy for other graphs. To overcome this, we adapt an SMT-PD protocol from [GO08] to realize AE-SMT over the graphs in [Upf92,CGO10,JRV20], at the cost of obtaining only statistical UC security. Somewhat surprisingly, for each class of graphs considered in Section 4, both AE-RMT and AE-SMT are achieved under the same doom structure.

Almost-everywhere secure computation. Lastly, we study the composability of AE-security guarantees, with the ultimate goal of realizing AE-MPC. In Section 5.1, we state and prove a general composition theorem, which makes precise the level or "quality" of AE-security (as captured in a doom structure) that is obtained when a protocol's hybrids are replaced with almost-everywhere counterparts, even against general (i.e., not necessarily threshold) adversaries (Theorem 14). We emphasize that this *AE compiler* need not replace all of the hybrids with AE-wrapped versions using the *same* doom structure; thus, we are able to explain, for example, what happens when a protocol uses subprotocols to emulate secure channels and broadcast over a sparse network, but those subprotocols provide different levels of AE-security.

Our composition theorem applies even to protocols that already carry some level of AE-security, and therefore the compiled protocol can easily be composed with higher-level protocols. The crux of the security proof is that the simulator for the compiled protocol can make use of an existing simulator for the original protocol, by pretending that doomed parties are fully corrupted (in reality the situation is more complex, because the given simulator may itself request to doom parties according to the AE-security of the original protocol). As a simple corollary, we show that a protocol achieving standard (non-AE) security using a single hybrid can be compiled into an AE-secure protocol while preserving the doom structure associated with the AE version of that hybrid, at the cost of tolerating a lower corruption threshold (Corollary 6).

In Section 5.2, we apply this corollary to obtain the first simulation-based proof of AE-MPC, over any of the classes of sparse graphs considered in the AE agreement literature—[DPPU86,Upf92,CGO10,JRV20]. In more detail, we simply overlay an AE-secure communication protocol (designed for a sparse network) with a standard information-theoretically secure MPC protocol (designed for fully connected networks), to obtain an AE-secure MPC protocol that only relies on secure channels between parties actually connected in the sparse network and yet achieves the same level of AE-security as the underlying AE-secure communication protocol (Corollaries 7 and 8). Depending on which class of sparse graphs is used, our results from Section 4.2 on realizing AE-SMT over those graphs convey either perfect or statistical UC security.

Next, we review some UC basics, and present some building blocks and their property-based definitions to contrast with our simulation-based treatment in later sections. For the sake of readability, some of the functionalities, protocols, and proofs are presented in the appendix.

2 Preliminaries

2.1 UC Basics

Our results are in the UC framework [Can01] and we briefly summarize it here. Protocol machines, ideal functionalities, the adversary, and the environment are all modeled as interactive Turing machine (ITM) instances, or ITIs. An execution of protocol π consists of a series of activations of ITIs, starting with the environment $\mathcal{Z}$ who provides inputs to and collects outputs from the parties and the adversary $\mathcal{A}$; parties can also give input to and collect output from sub-parties, and $\mathcal{A}$ can communicate with parties via messages. Corruption of parties is modeled by a special `corrupt` message sent from $\mathcal{A}$ to the party; upon receipt of this message, the party sends its entire local state to $\mathcal{A}$, and in all future activations follows the instructions of $\mathcal{A}$. Note that a party p_i can only be corrupted once $\mathcal{A}$ receives a special ($\texttt{corrupt}\ p_i$) input from $\mathcal{Z}$. The (binary) output of the environment $\mathcal{Z}$ at the end of an execution of π with adversary $\mathcal{A}$ is denoted by the random variable $\text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(k,z)$, where $k \in \mathbb{N}$ is the security parameter and $z \in \{0,1\}^*$ is the input to $\mathcal{Z}$. The ensemble $\{\text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}\}_{k\in\mathbb{N},z\in\{0,1\}^*}$ is denoted by $\text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$. The ideal-world process for functionality $\mathcal{F}$ is simply defined as an execution of the ideal protocol $\text{IDEAL}_{\mathcal{F}}$, in which the so-called "dummy" parties just forward inputs from $\mathcal{Z}$ to $\mathcal{F}$ and forward outputs from $\mathcal{F}$ to $\mathcal{Z}$ (in particular, the dummy parties do not communicate with the adversary, but rather the adversary is expected to send messages directly to $\mathcal{F}$, for example corruption messages). The corresponding ensemble is denoted by $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}}$, as the adversary in the ideal world is actually a simulator $\mathcal{S}$.

We are interested in unconditional security. Thus, we say that a protocol π *UC-realizes* an ideal functionality $\mathcal{F}$ if for any computationally unbounded adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}$ (which is polynomial in the complexity of $\mathcal{A}$) such that for any computationally unbounded environment $\mathcal{Z}$, we have $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}} \equiv \text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$. We sometimes consider statistical security, which requires only that the two ensembles be indistinguishable, not identical. For a $(\mathcal{G}_1,\ldots,\mathcal{G}_n)$-hybrid protocol π (which makes subroutine calls to the ideal protocols for the ideal functionalities $\mathcal{G}_1,\ldots,\mathcal{G}_n$), we say that π UC-realizes $\mathcal{F}$ in the $(\mathcal{G}_1,\ldots,\mathcal{G}_n)$-hybrid model. It turns out that (regular) UC-realization is equivalent to UC-realization with respect to a very specific adversary, namely the "dummy" adversary $\mathcal{D}$: this adversary simply follows the instructions of $\mathcal{Z}$ on which messages to send, and moreover reports all received messages to $\mathcal{Z}$. We sometimes use this alternate definition of security, as it is simpler to work with and involves one less quantifier.

Synchrony. We will assume synchronous computation, i.e., our protocols proceed in rounds, where in each round: the uncorrupted parties generate their messages for the current round, as described in the protocol; then the messages addressed to the corrupted parties become known to the adversary; then the adversary generates the messages to be sent by the corrupted parties in this round; and finally, each uncorrupted party receives all the messages sent in this round. Although our treatment is in the (G)UC setting, to avoid over-complicating the exposition, we will use the standard round-based language of, e.g., [Can00,Nie03] to specify our protocols. Notwithstanding, such specifications can be directly translated to the synchronous UC model of Katz *et al.* [KMTZ13] by assuming a clock functionality and bounded (zero) delay channels. (See [KMTZ13] for details.)

2.2 Building Blocks

Here we present building blocks that we will be using in our constructions and their property-based definitions. However, we are not completely formal with the definitions since their main purpose is to contrast with our UC formulations of them.

Recall that the SMT problem involves a sender S connected to a receiver R over n disjoint wires. A solution to SMT is formally defined as follows:

Definition 1 (SMT). *A protocol Π achieves* SMT *if it allows S to send a message $m \in \mathcal{M}$ to R such that the following properties hold for any adversary $\mathcal{A}$ corrupting up to t of the wires:*
- ***Reliability:*** *R correctly outputs $m' = m$.*
- ***Secrecy:*** *$\mathcal{A}$ learns no information about m.*

We can define RMT by simply omitting the secrecy condition, and AE-RMT and AE-SMT are defined by only requiring the reliability and/or secrecy properties to hold for privileged S and R (e.g., according to an RMT or SMT protocol over a sparse network).

For simplicity, we will use the 3-phase protocol in Fig. 1 which is basically the FastSMT protocol from [DDWY90] that tolerates the optimal number (a minority) of "wire" corruptions. The n wires are denoted by $\vec{\gamma} = (\gamma_1,\ldots,\gamma_n)$ and let $\tau = \lceil \frac{n}{2} \rceil - 1$. The protocol works for messages in the field $\mathcal{M} = \mathbb{Z}_q$, and it assumes access to an authenticated channel between the sender and receiver (which can be implemented by simply duplicating the message to be sent over the n wires and having the receiver take majority). The high-level idea of the protocol is that the sender chooses $n\tau+1$ secret pads and secret-shares them among the n wires and associates each share with some checking pieces. The receiver uses the checking pieces to verify the correctness of each pad. If there is a correct pad, the receiver sends its index back to the sender using the authenticated channel so the sender can use the corresponding pad to encrypt the message and send it to the receiver via the authenticated channel. The receiver knows the pad and can decrypt the message successfully. Otherwise, if there is no correct pad, the receiver sends some error detection information to the sender over the authenticated channel. The sender uses the information to detect the faults and send them to the receiver associated with an encryption of the message. Now the receiver fixes the faults, computes the correct pad, and decrypts the message. All the arithmetic in the protocol is modular.

Protocol $\Pi_{\mathrm{DDWY}}(\vec{\gamma}, \tau, m)$

1. **(Phase 1)** The sender S sends $n\tau+1$ strong pads SP_1,SP_2, ..., $SP_{n\tau+1}$. To send each strong pad, S chooses a random polynomial $f(x) \in \mathbb{Z}_q(x)$ of degree τ and sets $pad = f(0)$. Then for each $i \in [n]$ S chooses an additional random polynomial $h_i(x) \in \mathbb{Z}_q$ of degree τ such that $h_i(0) = f(i)$. Finally, for each $i \in [n]$, S sends $h_i(\cdot)$ with a vector of checking pieces $C_i = (c_{1i}, c_{2i}, \ldots, c_{ni})$ to R using wire γ_i where for all $i, j \in [n]$, $c_{ji} = h_j(i)$.
2. **(Phase 2)** For each $k \in [n]$, let T_k be received in the attempted transmission of SP_k and g_i, D_i be possibly corrupted information received as h_i, C_i. If for any T_a all the checking pieces c_{ji} and all polynomials $h_i(\cdot)$ are consistent then R interpolates the pad_a from T_a and sends "a, OK" to S over the authenticated channel. Otherwise, R finds a l such that

$$\{\text{conflicts of } T_l\} \subset \cup_{m \neq l}\{\text{conflicts of } T_m\},$$

   where any unordered pair (i, j) is called a conflict of T_k if $d_{ji} \neq g_j(i)$. Then R sends l and all T_m, $m \neq l$ back to S using authenticated channel.
3. **(Phase 3)**
   - If "a, OK" received over the authenticated channel in phase 2, then S sends $z = m + pad_a$ to R using the authenticated channel. Otherwise, S preforms error detection on all T_j's received from R and sends detected faults and $z = m + pad_i$ to R using authenticated channel.
   - If R previously sent "a, OK" to S in phase 1, then s/he computes $m = z - pad_a$. Otherwise, R corrects the faults in T_i, obtains pad_i and computes $m = z - pad_i$.

Fig. 1. Perfectly secure message transmission protocol over wires

We will sometimes need an SMT-PD protocol, and for that we use the protocol in Fig. 2 from [GO08], which tolerates $n-1$ wire corruptions, assuming access to a public channel and allowing a small probability of error. Intuitively, the sender sends n random bit strings each over a wire to the receiver. In the next step, the sender uses the public channel to send the values corresponding to some random positions in each bit string to the receiver. Now the receiver can use the revealed bits to detect the tampered wires with some probability. Then the receiver informs the sender about the faulty wires over the public channel. Now the sender uses the remaining private bits of the (potentially) non-faulty wires to generate a key and encrypt the message before sending it via the public channel. The receiver can also compute the same key and decrypt the ciphertext. In the protocol, E and D are respectively the encoding and decoding algorithms for an error-correcting code.

Finally, we present the security definition for (property-based) AE-MPC that was given in [GO08]. Recall that W is the set of privileged nodes, as a function of the set of corruptions.

Definition 2 (AE-MPC). *An n-player two-phase protocol Π achieves* AE-MPC *if for any initial value x_i for party P_i for each $i \in [n]$, any probabilistic polynomial-time computable function f, and any adversary $\mathcal{A}$ corrupting a set T of parties there exists a subset W of honest parties such that the following two properties hold at the end of the respective phases:*

Commitment phase: *During this phase, all players commit to their inputs.*

- BINDING: *For all P_i there is a uniquely defined value x_i^*; if $P_i \in W$, then $x_i^* = x_i$.*
- PRIVACY: *For all $P_i \in W$, x_i^* is information-theoretically hidden.*

Computation phase:

- CORRECTNESS: *All $P_i \in W$ output $f(x_1^*, \ldots, x_n^*)$.*
- PRIVACY: *For all $P_i \in W$, no information about x_i^* beyond what can be inferred from the output of the corrupted parties leaks to $\mathcal{A}$ by this phase.*

Protocol $\Pi_{\text{PUB-SMT}}(\vec{\gamma}, Pub, m, l)$

1. The sender S sends n uniformly random bit strings $R_1, R_2, \ldots, R_n$ of length $15l$ to the receiver R through wires $\gamma_1, \gamma_2, \ldots, \gamma_n$, respectively. Let $R'_1, R'_2, \ldots, R'_n$ be the strings received by R. R rejects all wires where $|R'_i| \neq 15l$.
2. For $i \in [n]$, S generates R^*_i by replacing $12l$ randomly chosen positions of R_i with "$*$." Then S sends $R^*_1, R^*_2, \ldots, R^*_n$ to R over Pub.
3. For any $i \in [n]$, if R^*_i and R'_i differ in any "opened" bits, R marks γ_i as "faulty." Then R sends an n-bit string to S over Pub that identifies faulty wires. Let $\vec{\overline{\gamma}} = \{\overline{\gamma_1}, \overline{\gamma_2}, \ldots, \overline{\gamma_s}\}$, $s \leq n$ denote the set of non-faulty wires, and $\overline{R_i}$, $|\overline{R_i}| = 12l$, $1 \leq i \leq s$, denote the corresponding string of unopened bits; let $\overline{R'_i}$ be the corresponding string is R's possession.
4. For $1 \leq i \leq s$, S chooses m_i such that $m = m_1 \oplus m_2 \oplus \cdots \oplus m_s$, and sends $S_i = E(m_i) \oplus \overline{R_i}$, $1 \leq i \leq s$, over Pub. R computes $m'_i = D(S_i \oplus \overline{R'_i})$ for all $1 \leq i \leq s$. Then R outputs $m' = m'_1 \oplus m'_2 \oplus \cdots \oplus m'_s$.

Fig. 2. Secure message transmission by public discussion protocol over wires

3 Almost-Everywhere RMT and SMT

In this section, we use the UC framework to capture classical RMT and SMT protocols, which work in a model where the sender S and receiver R are connected by n disjoint *wires*, as in the abstract formulation in [DDWY90]. Although this is a simple model, here we give a novel treatment of these tasks that also serves as a warm-up to our later results, which look at these tasks over sparse graphs. Since the classical protocols may not provide security when enough of the wires are corrupted, we also introduce an AE *wrapper* that allows parties interacting with the underlying functionality to be marked as "doomed" in such cases. In Section 4, where we consider *remote* RMT and SMT, we will realize the same functionalities for RMT and SMT defined in this section, just in a wrapped form albeit with different parameters.

We begin by modeling the disjoint wires from the classical setting as virtual wires that are represented by UC parties, which we call *wire-parties* and denote by $W_1, \ldots, W_n$. The idea is that a wire-party can securely forward a message from S to R or vice versa as long as it is not corrupted, just as a wire in the classical model can securely transmit a message between S and R as long as it is free of corruptions. Since the basic communication model in UC is completely unprotected, we assume access to the ideal secure channel functionality $\mathcal{F}_{\text{SC}}^{S,R,\vec{W}}$ in Fig. 3 , which provides secure communication between an honest sender or receiver and an honest wire-party over a single round. Looking ahead, this functionality is very similar to the functionality that we will use to capture secure channels between every pair of nodes connected by an edge in a sparse graph. In $\mathcal{F}_{\text{SC}}^{S,R,\vec{W}}$ (and all of our functionalities), $l(\cdot)$ refers to the length of its input, and Infl is short for "influence" (see, e.g., [GKZ10]).

Functionality $\mathcal{F}_{\text{SC}}^{S,R,\vec{W}}$

The secure channel functionality $\mathcal{F}_{\text{SC}}$ is parameterized by the identities of the sender S, the receiver R, and the n wire-parties $\vec{W} = (W_1, \ldots, W_n)$, and it proceeds as follows. At the first activation, verify that $sid = (P_i, P_j, sid')$, where one of P_i and P_j is either S or R, and the other is some wire-party W_i; else halt. Initialize variable m to a default value $\perp$.

– Upon receiving input (Send, sid, v) from P_i in round ρ, record $m \leftarrow v$. If P_i or P_j are marked as corrupted, then send $(\text{SendLeak}, sid, m)$ to the adversary; otherwise send $(\text{SendLeak}, sid, l(m))$.
– Upon receiving $(\text{InflSend}, sid, m')$ from the adversary: If P_i or P_j is corrupted, and (Sent, sid, m) has not yet been sent to P_j, then update $m \leftarrow m'$; otherwise, ignore the command.
– Upon receiving (Fetch, sid) from P_j in round $\rho + 1$, output (Sent, sid, m) to P_j if it has not yet been sent.
– Upon receiving $(\text{Corrupt}, sid, P)$ from the adversary for $P \in \{P_i, P_j\}$, mark P as corrupted and send $(\text{SendLeak}, sid, m)$ to the adversary.

Fig. 3. Secure channel functionality for the wire-party model

For convenience, we use $\mathcal{F}_{\text{SC}}^{S,R,\vec{W}}$ to realize the wire channel functionality $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ in Fig. 4 , which abstracts the process of sending a message to a wire-party, who then forwards it to S or R. The functionality actually allows sending a potentially different message through each wire-party in parallel, and it provides security for a given message as long as S, R, and the wire-party in question are all honest. In addition to simplifying our RMT and SMT protocols, this functionality also has a very intuitive interpretation: it models the sending of messages in a single "phase," in the terminology of the PSMT literature. Note that since we are considering virtual wires that consist of just one intermediate node, the functionality requires two rounds to generate output.

Functionality $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$

The wire channel functionality $\mathcal{F}_{\text{WC}}$ is parameterized by the identities of the sender S, the receiver R, and the n wire-parties $\vec{W} = (W_1, \dots, W_n)$, and it proceeds as follows. At the first activation, verify that $sid = (P_s, P_r, sid')$, where either $P_s = S$ and $P_r = R$, or $P_s = R$ and $P_r = S$; else halt. Initialize variables $m_1, \dots, m_n$ to a default value $\perp$.

- Upon receiving input $(\textsc{Send}, sid, W_i, v_i)$ from P_s in round ρ (which is the same for all W_i), record $m_i \leftarrow v_i$. If any $P \in \{P_s, P_r, W_i\}$ is marked as corrupted, then send $(\textsc{SendLeak}, sid, W_i, m_i)$ to the adversary; otherwise send $(\textsc{SendLeak}, sid, W_i, l(m_i))$.
- Upon receiving $(\textsc{InflSend}, sid, W_i, m'_i)$ from the adversary: If any $P \in \{P_s, P_r, W_i\}$ is corrupted, and $(\textsc{Sent}, sid, W_i, m_i)$ has not yet been sent to P_r, then update $m_i \leftarrow m'_i$; otherwise, ignore the command.
- Upon receiving $(\textsc{Fetch}, sid, W_i)$ from P_r in round ρ': If P_r is corrupted, then send $(\textsc{FetchLeak}, sid, W_i)$ to the adversary; otherwise, if $\rho' = \rho + 2$, then output $(\textsc{Sent}, sid, W_i, m_i)$ to P_r if it has not yet been sent.
- Upon receiving $(\textsc{Output}, sid, W_i)$ from the adversary: If P_r is corrupted, then output $(\textsc{Sent}, sid, W_i, m_i)$ to P_r if it has not yet been sent; otherwise, ignore the command.
- Upon receiving $(\textsc{Corrupt}, sid, P)$ from the adversary for $P \in \{P_s, P_r, W_1, \dots, W_n\}$, mark P as corrupted. If P is some wire-party W_i, then send $(\textsc{SendLeak}, sid, m_i)$ to the adversary; otherwise, send $(\textsc{SendLeak}, sid, m_1, \dots, m_n)$. If $P = P_r$, then additionally leak any previous fetch requests made by P_r.

Fig. 4. Wire communication functionality

We can use the simple protocol $\Pi_{\text{WC}}(S, R, \vec{W})$ in Fig. 17 (Appendix B) to realize $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$:

Theorem 1. *Protocol $\Pi_{\text{WC}}(S, R, \vec{W})$ UC-realizes $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$, in the $\mathcal{F}_{\text{SC}}^{S,R,\vec{W}}$-hybrid model.*

3.1 Universally Composable RMT and SMT

We model the task of RMT in UC with the authenticated channel functionality $\mathcal{F}_{\text{AUTH}}^{\mathcal{P},\mathsf{rnd}}$ in Fig. 5 , which is essentially Canetti's $\mathcal{F}_{\text{AUTH}}$ [Can05] with synchrony (the rnd parameter). There is also a parameter $\mathcal{P}$ representing the set of possible senders and receivers (the functionality itself is single-use). This parameter allows the functionality to verify that the actual sender and receiver can be identified as specific nodes in the network topology over which it is being realized, which is necessary because the realizing protocol will need to perform the same verification.

To realize $\mathcal{F}_{\text{AUTH}}$ in the wire-party model under the assumption that only a minority of the wire-parties get corrupted, we can simply duplicate the message through all wire-parties and have the receiver (which may actually be S) take majority. We give a formal description of protocol $\Pi_{\text{AUTH}}(S, R, \vec{W})$ in Fig. 18 (Appendix B)[3].

Theorem 2. *Protocol $\Pi_{\text{AUTH}}(S, R, \vec{W})$ UC-realizes $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}}$ for $\mathsf{rnd} = 2$, in the $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$-hybrid model against an adversary corrupting up to a minority of the wire-parties.*

Next, we consider SMT in UC. We model the task with the secure channel functionality $\mathcal{F}_{\text{SMT}}^{\mathcal{P},\mathsf{rnd}}$ in Fig. 6 , which is essentially Canetti's $\mathcal{F}_{\text{SMT}}$ [Can05] with synchrony. To realize $\mathcal{F}_{\text{SMT}}$ in the wire-party model under

[3] All of our protocols for RMT only require reliable edges. However, to reduce the number of aiding functionalities (for simplicity) we present all of our RMT protocols in the secure channel hybrid model because eventually we need secure channels for SMT and MPC.

Functionality $\mathcal{F}_{\text{AUTH}}^{\mathcal{P},\mathsf{rnd}}$

The authenticated channel functionality $\mathcal{F}_{\text{AUTH}}$ is parameterized by a set $\mathcal{P}$ of possible senders and receivers as well as an integer rnd indicating the number of rounds that will be used to realize it, and it proceeds as follows. At the first activation, verify that $sid = (S, R, sid')$, where $S, R \in \mathcal{P}$; else halt. Initialize variable m to a default value $\perp$.

- Upon receiving input (SEND, sid, v) from S in round ρ, record $m \leftarrow v$ and send $(\text{SENDLEAK}, sid, m)$ to the adversary.
- Upon receiving $(\text{INFLSEND}, sid, m')$ from the adversary: If S or R is marked as corrupted, and (SENT, sid, m) has not yet been sent to R, then update $m \leftarrow m'$; otherwise, ignore the command.
- Upon receiving input (FETCH, sid) from R in round ρ': If R is corrupted, then send $(\text{FETCHLEAK}, sid)$ to the adversary; otherwise, if $\rho' = \rho + \mathsf{rnd}$, then output (SENT, sid, m) to R if it has not yet been sent.
- Upon receiving (OUTPUT, sid) from the adversary: If R is corrupted, then output (SENT, sid, m) to R if it has not yet been sent; otherwise, ignore the command.
- Upon receiving $(\text{CORRUPT}, sid, P)$ from the adversary for $P \in \{S, R\}$, mark P as corrupted. If $P = R$, then leak any previous fetch requests made by R to the adversary.

Fig. 5. Authenticated communication functionality

Functionality $\mathcal{F}_{\text{SMT}}^{\mathcal{P},\mathsf{rnd}}$

The secure channel functionality $\mathcal{F}_{\text{SMT}}$ is parameterized by a set $\mathcal{P}$ of possible senders and receivers as well as an integer rnd indicating the number of rounds that will be used to realize it, and it proceeds as follows. At the first activation, verify that $sid = (S, R, sid')$, where $S, R \in \mathcal{P}$; else halt. Initialize variable m to a default value $\perp$.

- Upon receiving input (SEND, sid, v) from S in round ρ, record $m \leftarrow v$. If S or R is marked as corrupted, then send $(\text{SENDLEAK}, sid, m)$ to the adversary; otherwise, send $(\text{SENDLEAK}, sid, l(m))$.
- Upon receiving $(\text{INFLSEND}, sid, m')$ from the adversary: If S or R is corrupted, and (SENT, sid, m) has not yet been sent to R, then update $m \leftarrow m'$; otherwise, ignore the command.
- Upon receiving input (FETCH, sid) from R in round ρ': If R is corrupted, then send $(\text{FETCHLEAK}, sid)$ to the adversary; otherwise, if $\rho' = \rho + \mathsf{rnd}$, then output (SENT, sid, m) to R if it has not yet been sent.
- Upon receiving (OUTPUT, sid) from the adversary: If R is corrupted, then output (SENT, sid, m) to R if it has not yet been sent; otherwise, ignore the command.
- Upon receiving $(\text{CORRUPT}, sid, P)$ from the adversary for $P \in \{S, R\}$, mark P as corrupted and send $(\text{SENDLEAK}, sid, m)$ to the adversary. If $P = R$, then additionally leak any previous fetch requests made by R.

Fig. 6. Secure message transmission functionality

the assumption that only a minority of the wire-parties get corrupted, we can run protocol $\Pi_{\text{SMT}}(S, R, \vec{W})$ (Fig. 7), which is essentially the FastSMT protocol from [DDWY90] in our model.

Protocol $\Pi_{\text{SMT}}(S, R, \vec{W})$

Upon receiving input $(\textsc{Send}, sid, v)$ from $\mathcal{Z}$ in round ρ, where $sid = (P_s, P_r, sid')$ for either $P_s = S$ and $P_r = R$ or $P_s = R$ and $P_r = S$, party P_s executes protocol $\Pi_{\text{DDWY}}(\vec{\gamma}, \tau, v)$ with party P_r, where the wires $\gamma_1, \ldots, \gamma_n$ in $\vec{\gamma}$ are taken to be the virtual wires corresponding to the wire-parties $W_1, \ldots, W_n$:

1. In the first phase, P_s uses a single instance of $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ with $sid_1 = (sid, 1)$ to send all the messages instead of using wires in $\vec{\gamma}$. Next, in the second and third phase, P_r and P_s substitute the authenticated channel with separate instances of $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}}$ with $\mathsf{rnd} = 2$ and $sid_2 = (P_r, P_s, sid', 2)$ and $sid_3 = (P_s, P_r, sid', 3)$, respectively. To receive output from the aiding functionalities, P_s and P_r have to send $\textsc{Fetch}$ messages to the functionalities using the correct session IDs as generated above. Note that P_s and P_r execute the protocol in rounds, with two rounds per flow of communication as both $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ and $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}}$ with $\mathsf{rnd} = 2$ are two-round functionalities.
2. Upon receiving input $(\textsc{Fetch}, sid)$ from $\mathcal{Z}$ in round $\rho' = \rho + 6$, P_r outputs $(\textsc{Sent}, sid, m')$ to $\mathcal{Z}$ if it receives m' as the output of this protocol.

Fig. 7. Secure message transmission protocol

Theorem 3. *Protocol* $\Pi_{\text{SMT}}(S, R, \vec{W})$ *UC-realizes* $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ *for* $\mathsf{rnd} = 6$, *in the* $(\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}'}, \mathcal{F}_{\text{WC}}^{S,R,\vec{W}})$*-hybrid model where* $\mathsf{rnd}' = 2$, *against an adversary corrupting a minority of the wire-parties.*

Proof. Let $\mathcal{A}$ be an adversary in the real world. We construct a simulator $\mathcal{S}$ in the ideal world, such that no environment can distinguish whether it is interacting with $\Pi_{\text{SMT}}(S, R, \vec{W})$ and $\mathcal{A}$, or with $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ and $\mathcal{S}$. The simulator internally runs a copy of $\mathcal{A}$, and plays the roles of $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}}$, $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$, and the parties in a simulated execution of the protocol. All inputs from $\mathcal{Z}$ are forwarded to $\mathcal{A}$, and all outputs from $\mathcal{A}$ are forwarded to $\mathcal{Z}$. Moreover, whenever $\mathcal{A}$ corrupts a party in the simulation, $\mathcal{S}$ corrupts the same party in the ideal world by interacting with $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ (except if the party is a wire-party), and if the corruption was direct (i.e., not via one of the aiding functionalities), then $\mathcal{S}$ sends $\mathcal{A}$ the party's state and follows $\mathcal{A}$'s instructions thereafter for that party.

The simulated execution starts upon $\mathcal{S}$ receiving $(\textsc{SendLeak}, sid, \hat{m})$ from $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ in round ρ for $sid = (P_s, P_r, sid')$, where $\hat{m} \in \{m, l(m)\}$ and m is the message to be sent. Now, $\mathcal{S}$ executes the first and second phase of the DDWY protocol honestly, by simulating sending random strong pads (shares $h_i(\cdot)$ and checking pieces $\vec{C}_i = (c_{1i}, \ldots, c_{ni})$) from P_s to P_r through the n wire-parties (i.e., by simulating leakage from $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ to $\mathcal{A}$, and responding to corruption and influence requests directed from $\mathcal{A}$ to that functionality) and by simulating sending the response from P_r to P_s over the authenticated channel (i.e., by appropriately playing the role of $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}}$ for $\mathcal{A}$). For the third phase of the DDWY protocol (i.e., once P_s receives P_r's response), $\mathcal{S}$ simulates as per the DDWY protocol except for choosing z when P_s and P_r are both honest in which $\mathcal{S}$ simulates sending a random value z from P_s to P_r over $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}}$ instead of $z = m \oplus Pad$. It should be noted that when P_s or P_r is corrupted by $\mathcal{A}$, $\mathcal{S}$ learns m from $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ and thus can send $z = m \oplus Pad$ just like the protocol. Note that the simulated P_s may need to abort, and that if the simulated P_r aborts by outputting $\perp$, then $\mathcal{S}$ can simply send an $\textsc{InflSend}$ message to $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$, since this can only happen if $\mathcal{A}$ corrupts P_s or P_r.

Next, we describe how $\mathcal{S}$ simulates P_r's response to a $\textsc{Fetch}$ input from $\mathcal{Z}$ in the real world. If P_r is corrupted by $\mathcal{A}$, then $\mathcal{S}$ can wait to receive $(\textsc{FetchLeak}, sid)$ from $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$, upon which it possibly leaks the fetch to $\mathcal{A}$ and then sends $\textsc{InflSend}$ and $\textsc{Output}$ messages to $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ as appropriate (this case involves $\mathcal{S}$ behaving as the simulator for the AUTH protocol does, except that here we are in the third phase of the protocol, and $\textsc{Fetch}$ inputs that come too early are ignored). If P_s is corrupted by $\mathcal{A}$, then $\mathcal{S}$ needs to constantly influence $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ during the second phase of the protocol, so that the dummy P_r fetches the most up-to-date value when instructed by $\mathcal{Z}$. This case is also handled in a similar fashion to the corresponding case in the proof of the AUTH protocol; however, it is worth noting that even if P_s behaves honestly in the third phase of the protocol, previous misbehavior in the first phase may cause $\mathcal{S}$ to have to immediately influence $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$. If neither P_s nor P_r is corrupted, then $\mathcal{S}$ can simply let the dummy P_r fetch from $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ when instructed by $\mathcal{Z}$. The case that needs more contemplation happens when both P_s and P_r are honest in the beginning of the third phase (at the time $\mathcal{S}$ is deciding about the value of z) and then at least one of them gets corrupted before the protocol ends (before the output is fetched). It is important because in that case, $\mathcal{A}$ receives enough leakage from $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ to interpolate the pad and compute the value of the message from z. Since z is chosen randomly by $\mathcal{S}$, the message learnt by $\mathcal{A}$ deviates from what is sent by P_s which causes $\mathcal{Z}$ distinguish the two worlds. In such a situation, also $\mathcal{S}$ learns the actual value of m from $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ hence it can cheat by calculating a fake pad' satisfying $z = m \oplus pad'$ and then simulate leaking from $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ such that it results in pad'. This way $\mathcal{A}$ learns the message m correctly.

It is easy to see that this simulation is perfect. In particular, when $\mathcal{A}$ does not corrupt P_s or P_r, for each strong pad at most τ shares and their associated checking pieces are revealed to $\mathcal{A}$ in the real world because of our assumption that only a minority of wire-parties are corrupted. Assume that I is the set of indices for corrupted wire-parties, so for each strong pad $\mathcal{A}$ learns $h_j(\cdot), (c_{1j}, c_{2j}, \ldots, c_{nj})$ for all $j \in I$ where $c_{ij} = h_i(j)$ for all $i, j \in [n]$. Since all $h_i(\cdot)$ are random polynomials of degree τ, $\Pr[h_i(0) = a \mid \{c_{ij}\}_{j\in I}] = \Pr[h_i(0) = a]$ and since $h_i(\cdot)$'s are chosen independently $\Pr[h_i(0) = a \mid \{c_{kj}\}_{k\in[n],j\in I}] = \Pr[h_i(0) = a]$. Therefore, by corrupting all the wires with indices in $I_{|I|\leq\tau}$, no information about $h_i(0)$ for $i \notin I$ leaks to $\mathcal{A}$. Moreover, we know that $h_i(0) = f(i)$ and since $f(\cdot)$ is also a random polynomial of degree τ we have $\Pr[f(0) = a \mid \{h_i(0)\}_{i\in I}] = \Pr[f(0) = a]$ ($f(0)$ is the value of the pad). The last probability implies that whichever strong pad is chosen by the protocol, it looks uniformly random to $\mathcal{A}$ and alternatively $\mathcal{Z}$. It means that regardless of which distribution m is chosen from, $z = m \oplus pad$ looks uniformly random to $\mathcal{A}$ and $\mathcal{Z}$ if no more that τ wire-parties are corrupted and P_s and P_r are both honest. Therefore, choosing a random value z by $\mathcal{S}$ looks perfectly indistinguishable from the real protocol execution to $\mathcal{Z}$. At the same time, $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}}$ provides genuine authentication of messages intended to be sent on the authenticated channel in the DDWY protocol, and hence in the real world P_r outputs the sender's input. $\square$

3.2 Corrupted Majorities of Wire-Parties

In the wire-party model, $\mathcal{F}_{\text{AUTH}}$ and $\mathcal{F}_{\text{SMT}}$ can only be realized when the adversary is restricted to corrupting only a minority of wire-parties. When corrupted majorities are allowed, the sender and receiver may essentially become doomed. To allow the simulator to handle such cases, we introduce an *AE wrapper functionality* (Fig. 8) that allows parties to be marked as doomed according to the current set of corruptions. The wrapper accepts "doom" requests according to an adversary structure, and it processes them by simply having the underlying functionality treat doomed parties as fully corrupted. Recall that an adversary structure is a set of c-vectors of subsets of a participant set $\mathcal{P}$, where each component of a vector represents corruptions of a certain type. We consider adversary structures that consist of *doubles* of subsets, corresponding to a corrupted set and a doomed set, respectively, although the two may intersect[4]. We call such structures *doom structures.*

In the model with sender S and receiver R connected by wire-parties $W_1, \ldots, W_n$, we can realize wrapped $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}}$ and $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ with doom structure $\mathscr{D}_{\text{PSMT}}$, defined as follows using participant set $\mathcal{P} = \{S, R, W_1, \ldots, W_n\}$:

- $(T_i, D_i) \in \mathscr{D}_{\text{PSMT}}$ if and only if either $\left|T_i \setminus \{S,R\}\right| < \frac{n}{2}$ and $D_i = \emptyset$ or $\left|T_i \setminus \{S,R\}\right| \geq \frac{n}{2}$ and $D_i \subseteq \{S,R\}$

Theorem 4. *Protocol $\mathit{\Pi}_{\text{AUTH}}(S,R,\vec{W})$ UC-realizes $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{PSMT}}}(\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}})$ for $\mathsf{rnd} = 2$, in the $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$-hybrid model, even against corrupted majorities of wire-parties.*

Proof. Let $\mathcal{A}$ be an adversary in the real world. We construct a simulator $\mathcal{S}$ in the ideal world, such that no environment can distinguish whether it is interacting with $\mathit{\Pi}_{\text{AUTH}}(S,R,\vec{W})$ and $\mathcal{A}$, or with $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{PSMT}}}(\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}})$

[4] This is a technicality, which simplifies some of our definitions and results. For example, the definition of AE-monotonicity (Section 5.1) would not be quite as short and intuitive otherwise. We could define things differently, but that would require small changes elsewhere.

Wrapper Functionality $\mathcal{W}_{\text{AE}}^{\mathscr{D}}(\mathcal{F})$

The wrapper functionality $\mathcal{W}_{\text{AE}}$ is parameterized by a doom structure $\mathscr{D} = \{(T_1, D_1), \ldots, (T_m, D_m)\}$, where each class $(T_i, D_i) \in 2^{\mathcal{P}} \times 2^{\mathcal{P}}$. The underlying functionality is $\mathcal{F}$. Let T be the set of currently corrupted parties and let D be the set of currently doomed parties, both initialized to $\emptyset$.

- Upon receiving $(\textsc{Corrupt}, sid, P_i)$ from the adversary for some $P_i \in \mathcal{P}$: If $(T \cup \{P_i\}, D) \in \mathscr{D}$, then update $T \leftarrow T \cup \{P_i\}$, relay the message to $\mathcal{F}$, and relay $\mathcal{F}$'s response to the adversary.
- Upon receiving $(\textsc{Doom}, sid, P_i)$ from the adversary for some $P_i \in \mathcal{P}$: If $(T, D \cup \{P_i\}) \in \mathscr{D}$, then update $D \leftarrow D \cup \{P_i\}$, send $(\textsc{Corrupt}, sid, P_i)$ to $\mathcal{F}$, and relay $\mathcal{F}$'s response to the adversary.
- Any other request from any party or the adversary is simply relayed to $\mathcal{F}$ without any further action and the output is relayed to the destination specified by $\mathcal{F}$.

Fig. 8. AE wrapper functionality

and $\mathcal{S}$. The simulator $\mathcal{S}$ is very similar to the simulator that was constructed in the proof of Theorem 2. However, $\mathcal{S}$ now interacts with a wrapped functionality, and corruption messages for wire-parties are indeed sent because they can now be processed by the wrapper. The other difference is that the case in which P_s and P_r are not corrupted by $\mathcal{A}$ becomes more complicated. If $\mathcal{A}$ corrupts only a minority of the wire-parties, then $\mathcal{S}$ can simply let the dummy P_r fetch its output as before, albeit from the wrapper. Otherwise, as soon as enough wire-parties are corrupted, $\mathcal{S}$ sends a $\textsc{Doom}$ message for P_s to the wrapper, which will be accepted by definition of $\mathscr{D}_{\text{PSMT}}$. Now, $\mathcal{S}$ can influence the wrapper every time the value that the real-world P_r would have output changes (note that these influence messages will in fact be accepted, because the wrapper will have sent a corruption message for P_s to the underlying $\textsc{auth}$ functionality). Once again, the simulation is perfect. □

Since we can only realize $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{PSMT}}}(\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}})$ against an unrestricted adversary, we modify Π_{SMT} to the protocol in Fig. 9 that works in $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{PSMT}}}(\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}})$-hybrid model.

Protocol $\Pi'_{\text{SMT}}(S, R, \vec{W})$

This protocol is defined as follows:

1. Replace invocations to $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}}$ in protocol $\Pi_{\text{SMT}}(S, R, \vec{W})$ with invocations to $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{PSMT}}}(\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}})$.

Fig. 9. Secure message transmission protocol in the wrapped $\mathcal{F}_{\text{AUTH}}$ hybrid model

Theorem 5. *Protocol $\Pi'_{\text{SMT}}(S, R, \vec{W})$ UC-realizes $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{PSMT}}}(\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}})$ for $\mathsf{rnd} = 6$, in the $(\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{PSMT}}}(\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}}), \mathcal{F}_{\text{WC}}^{S,R,\vec{W}})$-hybrid model, even against corrupted majorities of wire-parties.*

Proof. Let $\mathcal{A}$ be an adversary in the real world. We construct a simulator $\mathcal{S}$ in the ideal world, such that no environment can distinguish whether it is interacting with $\Pi'_{\text{SMT}}(S, R, \vec{W})$ and $\mathcal{A}$, or with $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{PSMT}}}(\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}})$ and $\mathcal{S}$. The simulator $\mathcal{S}$ is very similar to the simulator that was constructed in the proof of Theorem 3. However, $\mathcal{S}$ now interacts with a wrapped functionality, and corruption messages for wire-parties are indeed sent because they can now be processed by the wrapper. Another difference is that the case in which P_s and P_r are not corrupted by $\mathcal{A}$ becomes more complicated. If $\mathcal{A}$ corrupts only a minority of the wire-parties, then $\mathcal{S}$ can simply use a random value of z in the third phase of the protocol, and let the dummy P_r fetch its output as before, albeit from the wrapper.

Otherwise, as soon as enough wire-parties are corrupted, $\mathcal{S}$ sends a $\textsc{Doom}$ message for P_s to the wrapper, which will be accepted by definition of $\mathscr{D}_{\text{PSMT}}$, and obtains m as leakage because the wrapper will send a corruption message for P_s to the underlying $\textsc{smt}$ functionality. Now, $\mathcal{S}$ can use $z = m \oplus pad$ in the third phase, and can influence the wrapper every time the value that the real-world P_r would have output changes (note that these influence messages will in fact be accepted). An additional issue that comes up in the case that P_s and P_r remain honest is that $\mathcal{A}$ might exceed a minority of wire-party corruptions only after $\mathcal{S}$ has already chosen a random z. However, $\mathcal{S}$ can handle this by cheating and computing a fake pad consistent

Protocol $\Pi_{\text{SMT-PD}}(S, R, \vec{W}, l)$

Upon receiving input $(\textsc{Send}, sid, v)$ from $\mathcal{Z}$ in round ρ, where $sid = (P_s, P_r, sid')$ for either $P_s = S$ and $P_r = R$ or $P_s = R$ and $P_r = S$, party P_s executes protocol $\Pi_{\text{PUB-SMT}}(\vec{\gamma}, Pub, v, l)$ with party P_r, where the wires $\gamma_1, \ldots, \gamma_n$ in $\vec{\gamma}$ are taken to be the virtual wires corresponding to the wire-parties $W_1, \ldots, W_n$:

1. In the first phase, P_s uses a single instance of $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ with $sid_1 = (sid, 1)$ to send all the random bit strings instead of using wires in $\vec{\gamma}$. Next in the second, third, and fourth phases, P_s and P_r substitute the public channel Pub with separate instances of $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}'}$ using $sid_2 = (P_s, P_r, sid', 2)$, $sid_3 = (P_r, P_s, sid', 3)$, and $sid_4 = (P_s, P_r, sid', 4)$, respectively. To receive output from the aiding functionalities, P_s and P_r have to send $\textsc{Fetch}$ messages to the functionalities using the correct session IDs as generated above. Note that P_s and P_r execute the protocol in rounds, with two and rnd' rounds per invocation of $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ and $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}'}$, respectively.
2. Upon receiving input $(\textsc{Fetch}, sid)$ from $\mathcal{Z}$ in round $\rho' = \rho + 2 + 3 \cdot \mathsf{rnd}'$, P_r outputs $(\textsc{Sent}, sid, m')$ to $\mathcal{Z}$ if it receives m' as the output of this protocol.
3. There are also several situations in which P_s or P_r has to abort. If P_r receives an invalid message or no message at all from $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}'}$ then P_r aborts by outputting $\perp$.

Fig. 10. Secure message transmission by public discussion protocol

with m, like the simulator in the proof of Theorem 3 does when P_s or P_r becomes corrupted only after $\mathcal{S}$ chooses a z. Finally, $\mathcal{S}$ may need to simulate sender or receiver aborts when $\mathcal{A}$ corrupts a majority of wire-parties but not P_s or P_r; this too can be done since influencing the wrapper will have an effect. Once again, the simulation is perfect. □

Next, we turn to SMT-PD (Section 2.2), which offers an alternative way to achieve SMT against a corrupted majority of wires, in the presence of a public channel.

3.3 Universally Composable SMT-PD

To capture SMT-PD in UC, we use our wire-party model from above, with the public channel modeled by assuming access to $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}'}$, for some rnd'. The protocol is outlined in Fig. 10.

Theorem 6. *Protocol $\Pi_{\text{SMT-PD}}(S, R, \vec{W}, l)$ statistically UC-realizes $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ for $\mathsf{rnd} = 2 + 3 \cdot \mathsf{rnd}'$ in the $(\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}, \mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}'})$-hybrid model, against an adversary corrupting all but one of the wire-parties.*

Proof. Let $\mathcal{A}$ be an adversary in the real world. We construct a simulator $\mathcal{S}$ in the ideal world, such that no unbounded environment $\mathcal{Z}$ can distinguish whether it is interacting with $\Pi_{\text{SMT-PD}}(S, R, \vec{W}, l)$ and $\mathcal{A}$ in the $(\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}, \mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}'})$-hybrid world, or with $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ and $\mathcal{S}$ in the ideal world. The simulator internally runs a copy of $\mathcal{A}$, and plays the roles of $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$, $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}'}$, and the parties in a simulated execution of the protocol. All inputs from $\mathcal{Z}$ are forwarded to $\mathcal{A}$, and all outputs from $\mathcal{A}$ are forwarded to $\mathcal{Z}$. Moreover, whenever $\mathcal{A}$ corrupts a party in the simulation, $\mathcal{S}$ corrupts the same party in the ideal world by interacting with $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ (except if the party is a wire-party), and if the corruption was direct (i.e., not via either of the aiding functionalities), then $\mathcal{S}$ sends $\mathcal{A}$ the party's state and thereafter follows $\mathcal{A}$'s instructions for that party.

The simulated execution starts upon $\mathcal{S}$ receiving $(\textsc{SendLeak}, sid, \hat{m})$ from $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ in round ρ for $sid = (P_s, P_r, sid')$, where $\hat{m} \in \{m, l(m)\}$ and m is the message to be sent. Now, $\mathcal{S}$ simulates the first three phases of the SMT-PD protocol honestly, by simulating sending random bitstrings from P_s to P_r through the n wire-parties (i.e., by simulating leakage from $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ to $\mathcal{A}$, and responding to corruption and influence requests directed from $\mathcal{A}$ to that functionality) and by simulating sending a message from P_s to P_r or vice versa over the public channel (i.e., by appropriately playing the role of $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}'}$ for $\mathcal{A}$). In the fourth phase, $\mathcal{S}$ chooses random m_i's to be encoded (rather than m_i's such that $m = m_1 \oplus \cdots \oplus m_s$) if P_s and P_r are still honest; if P_s or P_r is corrupted by $\mathcal{A}$, then $\mathcal{S}$ learns m from $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ and thus does not need to cheat.

Next, we describe how $\mathcal{S}$ simulates P_r's response to a $\textsc{Fetch}$ input from $\mathcal{Z}$ in the real world. If P_r is corrupted by $\mathcal{A}$, then $\mathcal{S}$ can wait to receive $(\textsc{FetchLeak}, sid)$ from $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$, upon which it possibly leaks the fetch to $\mathcal{A}$ and then sends InflSend and Output messages to $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ as appropriate. Otherwise, if P_s is corrupted by $\mathcal{A}$, then $\mathcal{S}$ needs to constantly influence $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ so that the dummy P_r fetches the most up-to-date value when instructed by $\mathcal{Z}$. Finally, if neither P_s nor P_r is corrupted, then $\mathcal{S}$ can simply let the dummy P_r fetch from $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ when instructed by $\mathcal{Z}$. In this case, the real-world P_r outputs m except with the error probability.

An important issue is that when P_s or P_r is corrupted only after $\mathcal{S}$ has already decided on the random m_i's to be encoded in the fourth phase, $\mathcal{A}$ may be able to recover some m' from its view of the bitstrings sent in the first phase, but m' may not equal m and this could allow $\mathcal{Z}$ to distinguish between the real and ideal worlds. However, $\mathcal{S}$ can handle this case by faking what was sent in the first phase. In particular, at least one bitstring (corresponding to an uncorrupted wire-party) sent in the first phase is not visible to $\mathcal{A}$, so $\mathcal{S}$ can redefine it to be consistent with m (which $\mathcal{S}$ learns from leakage from $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$).

We now claim that the simulation is valid. Although it is not perfect as there is an error probability, $\mathcal{Z}$ still cannot distinguish between the hybrid and ideal worlds. In particular, when P_s and P_r are not corrupted by $\mathcal{A}$, the assumption that $\mathcal{A}$ only corrupts all but one of the wire-parties implies that the random bitstring sent on at least one of the wires in the first phase of the protocol will mask the value of m from $\mathcal{A}$. □

4 Almost-Everywhere *Remote* RMT and SMT

In this section, we consider *remote*—i.e. over a sparse graph G_n—RMT and SMT. As in Section 3, we model the network topology using the parameterized secure channel functionality $\mathcal{F}_{\text{SC}}^{G_n}$ in Fig. 11 , that provides secure channels between parties that are connected in G_n.

Functionality $\mathcal{F}_{\text{SC}}^{G_n}$

The secure channel functionality $\mathcal{F}_{\text{SC}}$ is parameterized by a graph $G_n = (V, E)$ of party identities and communication edges, and it proceeds as follows. At the first activation, verify that $sid = (P_i, P_j, sid')$, where $(P_i, P_j) \in E$; else halt. Initialize variable m to a default value $\perp$.

- Upon receiving input (Send, sid, v) from P_i in round ρ, record $m \leftarrow v$. If P_i or P_j is marked as corrupted, then send $(\text{SendLeak}, sid, m)$ to the adversary; otherwise send $(\text{SendLeak}, sid, l(m))$.
- Upon receiving $(\text{InflSend}, sid, m')$ from the adversary: If P_i or P_j is corrupted, and (Sent, sid, m) has not yet been sent to P_j, then update $m \leftarrow m'$; otherwise, ignore the command.
- Upon receiving (Fetch, sid) from P_j in round $\rho + 1$, output (Sent, sid, m) to P_j if it has not yet been sent.
- Upon receiving $(\text{Corrupt}, sid, P)$ from the adversary for $P \in \{P_i, P_j\}$, mark P as corrupted and send $(\text{SendLeak}, sid, m)$ to the adversary.

Fig. 11. Secure channel functionality for incomplete graph G_n

For convenience, instead of working directly in the $\mathcal{F}_{\text{SC}}^{G_n}$-hybrid model, we use $\mathcal{F}_{\text{SC}}^{G_n}$ to realize the remote secure channel functionality $\mathcal{F}_{\text{R-SC}}^{G_n}$ in Fig. 12 , the counterpart to $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ from Section 3. This functionality provides secure communication over a single path, as long as no node on the path is corrupted.

Using protocol $\Pi_{\text{R-SC}}$ in Fig. 19 (Appendix B) we realize $\mathcal{F}_{\text{R-SC}}^{G_n}$ by simply forwarding the message along the path, which leads to the following statement (proof omitted).

Theorem 7. *Protocol $\Pi_{\text{R-SC}}(G_n)$ UC-realizes $\mathcal{F}_{\text{R-SC}}^{G_n}$ in the $\mathcal{F}_{\text{SC}}^{G_n}$-hybrid model.*

4.1 AE Remote RMT

Graphs of constant degree. We start off by considering the graphs considered in [DPPU86].

DPPU. We describe a transmission scheme due to Dwork *et al.* [DPPU86] which guarantees *reliable* communication (i.e., no privacy) for large sets of privileged nodes in various classes of graphs (see below). At a high level, the scheme associates with every node in the graph a *fan-in* set and a *fan-out* set of a fixed (but not necessarily constant) size. In addition, (not necessarily vertex-disjoint) paths from a node to its sets are

Functionality $\mathcal{F}_{\text{R-SC}}^{G_n}$

The remote secure channel functionality $\mathcal{F}_{\text{R-SC}}$ is parameterized by a graph $G_n = (V, E)$ of party identities and communication edges, and it proceeds as follows. At the first activation, verify that $sid = (S, P_1, \ldots, P_{k-1}, R, sid')$, where $\gamma := (S, P_1, \ldots, P_{k-1}, R)$ is a path in G_n; else halt. Initialize variable m to a default value $\perp$.

- Upon receiving input (Send, sid, v) from S in round ρ, record $m \leftarrow v$. If any $P \in \gamma$ is marked as corrupted, then send $(\text{SendLeak}, sid, m)$ to the adversary; otherwise send $(\text{SendLeak}, sid, l(m))$.
- Upon receiving $(\text{InflSend}, sid, m')$ from the adversary: If any $P \in \gamma$ is corrupted, and (Sent, sid, m) has not yet been sent to R, then update $m \leftarrow m'$; otherwise, ignore the command.
- Upon receiving (Fetch, sid) from R in round ρ': If R is corrupted, then send $(\text{FetchLeak}, sid)$ to the adversary; otherwise, if $\rho' = \rho + k$, then output (Sent, sid, m) to R if it has not yet been sent.
- Upon receiving (Output, sid) from the adversary: If R is corrupted, then output (Sent, sid, m) to R if it has not yet been sent; otherwise, ignore the command.
- Upon receiving $(\text{Corrupt}, sid, P)$ from the adversary for $P \in \gamma$, mark P as corrupted and send $(\text{SendLeak}, sid, m)$ to the adversary; if $P = R$, then additionally leak any previous fetch requests made by R.

Fig. 12. Remote secure channel functionality for single path communication

specified, as well as (vertex-disjoint) paths for all ordered pairs of one node's fan-out set to any other node's fan-in set. When node u wants to send a message to node v, they run the following three-phase protocol, Π_{DPPU}: first u sends the message to all members of its fan-out set; each member then sends the message to its connected (via a path) pair in v's fan-in set; and finally each member in v's fan-in set forwards the message to v, who accepts the value resulting from the majority of received values.

In more detail, let T denote the set of adversarial nodes, and t its maximal size, and for every node $u \in V$, let $\Gamma_{\text{in}}(u)$ and $\Gamma_{\text{out}}(u)$ denote u's fan-in and fan-out set, respectively, such that $|\Gamma_{\text{in}}(u)| = |\Gamma_{\text{out}}(u)| = s > 4t$. Given a set of adversarial nodes T, it is shown in [DPPU86] that if less than a $\frac{1}{8}$ fraction of the paths from a node u to $\Gamma_{\text{out}}(u)$ are corrupted (the path includes the end point in $\Gamma_{\text{out}}(u)$), less than a $\frac{1}{4}$ fraction of the paths from $\Gamma_{\text{out}}(u)$ to $\Gamma_{\text{in}}(v)$ are corrupted, and less than a $\frac{1}{8}$ fraction from $\Gamma_{\text{out}}(v)$ to node v are corrupted, then u and v can communicate reliably. Further, it is shown in [DPPU86] how to construct such a transmission scheme for several classes of graphs. Using the terminology from Section 1, the set of privileged nodes $W(T)$ in this case are the nodes u such that less than a $\frac{1}{8}$ fraction of the paths from u to both $\Gamma_{\text{in}}(u)$ and $\Gamma_{\text{out}}(u)$ are corrupted.

Now let x be the maximal size of $D(T)$ for all T of size at most t. Dwork *et al.* constructed different graphs on which the above three-phase transmission scheme achieves reliable communication between any two privileged nodes. In this setting, one would like to obtain as low a value of x as possible, while tolerating a large value of t (a constant fraction of n is the best possible). While [DPPU86] construct several classes of graphs in the context of Byzantine agreement, we present the parameters for only the graphs that use the above described three-phase transmission scheme (the other graphs use an appended or a different transmission scheme). The parameters achieved are as follows:

- For the butterfly network on n nodes: $t = O(\frac{n}{\log n})$ and $x = O(t \log t)$;
- for almost every r-regular graph ($r \geq 5$): $t = O(n^{1-\epsilon})$ and $x = O(t^{1+\delta} \log t)$, for some $0 < \delta < \epsilon < 1$.

The protocol for remote reliable message transmission, $\Pi_{\text{R-AUTH}}^{\text{DPPU}}$, based on the DPPU (three-phase) transmission scheme in the $\mathcal{F}_{\text{R-SC}}^{G_{\text{DPPU}}}$-hybrid model is outlined in Fig. 20 (Appendix B).

Let $G_{\text{DPPU}} = (V_{\text{DPPU}}, E_{\text{DPPU}})$ be a graph with $|V_{\text{DPPU}}| = n$ that allows a DPPU transmission scheme with fan-in and fan-out sets of size s. To realize wrapped $\mathcal{F}_{\text{AUTH}}^{V_{\text{DPPU}},\mathsf{rnd}}$ over G_{DPPU} with $\Pi_{\text{R-AUTH}}^{\text{DPPU}}$, we define the *doom structure* $\mathscr{D}_{\text{DPPU}}$ as follows:

- First let $D_{\text{DPPU}}(T_i)$ be a subset of participants P such that $P \in T_i$ or at least $\frac{1}{8}$ of the paths from P to $\Gamma_{\text{out}}(P)$ or at least $\frac{1}{8}$ of the paths from $\Gamma_{\text{in}}(P)$ to P are corrupted. Basically, $D_{\text{DPPU}}(T_i)$ is the set of all possible doomed participants associated with the set of corruptions T_i based on the DPPU transmission scheme. $(T_i, D_i) \in \mathscr{D}_{\text{DPPU}}$ if and only if $T_i \subset \mathcal{P}$, $|T_i| < s/4$, and $D_i \subset D_{\text{DPPU}}(T_i)$.

Now we can use $\Pi_{\text{R-AUTH}}^{\text{DPPU}}$ to realize $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{DPPU}}}(\mathcal{F}_{\text{AUTH}}^{V_{\text{DPPU}},\mathsf{rnd}})$:

Theorem 8. *Protocol $\Pi^{\text{DPPU}}_{\text{R-AUTH}}$ UC-realizes $\mathcal{W}^{\mathscr{D}_{\text{DPPU}}}_{\text{AE}}(\mathcal{F}^{V_{\text{DPPU}},\mathsf{rnd}}_{\text{AUTH}})$ for some $\mathsf{rnd} \in O(\log n)$, where $n = |V_{\text{DPPU}}|$, in the $\mathcal{F}^{G_{\text{DPPU}}}_{\text{R-SC}}$-hybrid model against an adversary corrupting less than $s/4$ nodes where s is the number of specified paths between pairs of nodes by the* DPPU *transmission scheme.*

We can also formulate the above result in threshold (as opposed to doom-structure) terms (cf. [DPPU86]):

Corollary 1. *Over a butterfly network of $n = m2^m$ nodes and in the presence of an adversary corrupting up to $t < 2^m/4$ nodes, $\Pi^{\text{DPPU}}_{\text{R-AUTH}}$ guarantees (perfect) reliable message transmission among all but at most $32t \log 16t$ nodes.*

Upfal. Building on [DPPU86], Upfal [Upf92] proposed an alternative transmission scheme for constant-degree graphs, which works over *any* graph; however, his optimal result is achieved only on constant-degree expander graphs with specific parameters. The main limitation of the scheme is that it is computationally inefficient (exponential). Upfal's protocol, Π_{UPFAL}, for reliable message transmission over any graph works as follows. To transmit a message m from a sender S to a receiver R, S sends m to R through all the simple paths connecting them. As the message travels along the paths to R, each node on the path appends the ID of the previous node to the message. This way each message received from a corrupted path will contain at least one ID of a corrupted node. (See [Upf92] for details.)

Protocol $\Pi^{\text{UPFAL}}_{\text{R-AUTH}}$ for remote RMT based on UPFAL transmission scheme in the $\mathcal{F}^{G_{\text{UPFAL}}}_{\text{SC}}$-hybrid model is described in Fig. 21 (Appendix B).

Let $G^{\text{UPFAL}}_n = (V_{\text{UPFAL}}, E_{\text{UPFAL}})$ be a d-regular expander graph with $|V_{\text{UPFAL}}| = n$. To realize wrapped $\mathcal{F}^{V_{\text{UPFAL}},\mathsf{rnd}}_{\text{AUTH}}$, we define the *doom structure* $\mathscr{D}_{\text{UPFAL}}$ as follows:

- Let $D(T_i)$ be the set defined by the following iterative process: Starting with the set $S = T_i$, repeatedly add all participants $Q \notin S$ such that at least $\frac{1}{5}$ of Q's neighbors (according to G^{UPFAL}_n) are in the set S. $(T_i, D_i) \in \mathscr{D}_{\text{UPFAL}}$ if and only if $T_i \subset \mathcal{P}$, $|T_i| < t < 1/72n$ and $D_i \subset D(T_i)$.

We can now use $\Pi^{\text{UPFAL}}_{\text{R-AUTH}}$ to realize $\mathcal{W}^{\mathscr{D}_{\text{UPFAL}}}_{\text{AE}}(\mathcal{F}^{V_{\text{UPFAL}},\mathsf{rnd}}_{\text{AUTH}})$:

Theorem 9. *Protocol $\Pi^{\text{UPFAL}}_{\text{R-AUTH}}$ UC-realizes $\mathcal{W}^{\mathscr{D}_{\text{UPFAL}}}_{\text{AE}}(\mathcal{F}^{V_{\text{UPFAL}},\mathsf{rnd}}_{\text{AUTH}})$ for some $\mathsf{rnd} \in O(\log n)$, where n is the number of nodes, in the $\mathcal{F}^{G^{\text{UPFAL}}_n}_{\text{SC}}$-hybrid model against an adversary corrupting less than $1/72n$ nodes.*

Note that the above simulator needs to run the potentially exponential-time process that R does at the end of the protocol to determine the output for the case that at least one of S and R is doomed. However, that seems reasonable since the protocol itself runs in exponential time.

As before, in threshold terms (cf. [Upf92]), we obtain:

Corollary 2. *Over any d-regular graph G with $\lambda(G) \leq 2\sqrt{d-1}$ and in the presence of an adversary corrupting up to $t < 1/72n$ nodes, $\Pi^{\text{UPFAL}}_{\text{R-AUTH}}$ guarantees (perfect) reliable message transmission among all but at most $6t$ nodes.*

We note that explicit constructions of d-regular graphs exist, with $\lambda(G) \leq 2\sqrt{d-1}$, for any $d = p+1$, p a prime [LPS86].

Graphs of poly-logarithmic degree. Chandran *et al.* [CGO10] proposed a randomly constructed graph, G^{CGO}_n, of poly-logarithmic degree and an almost-everywhere reliable message transmission scheme over it. A very high level idea of their construction is to transmit a message via multiple paths and also perform some sort of error correction along the way. To construct their graph, G^{CGO}_n, over n vertices, they first form $n \log^k n$ overlapping committees of size $O(\log \log n)$ via walks on expander graphs. Then they make each committee a clique by putting all the edges inside each committee. They also connect committees by "super-edges" using DPPU butterfly network. A super-edge connection between two committees means that every two corresponding nodes in those committees are connected (i.e. ith node of one committee is connected to ith node of the other one). In the end, each node is assigned an connected to a poly-logarithmic number of committees chosen by an expander graph. Committees connected to a node are called the node's *helper* committees. Chandran *et al.* proved that each node in the above graph has a poly-logarithmic degree.

The proposed protocol Π_{CGO} for reliable message transmission over G^{CGO}_n is as follows. To transmit a message m from a sender S to a receiver R, S sends m to all its helper committees. S's helper committees are nodes of a DPPU graph so they can send the message they have received to R's helper committees using the DPPU transmission scheme. Then all the R's helper committees forward the message to R. In the end, R takes a simple majority and outputs the value as the value received. We should mention that to send a message from one committee to another using a super-edge, each node sends the message to its corresponding node in the other committee and then all the nodes in the destination committee run a *differential agreement* protocol [FG03] over the values they have received.[5] (See [CGO10] for details.) Protocol $\Pi^{\text{CGO}}_{\text{R-AUTH}}$ for remote reliable message transmission based on the CGO transmission scheme in the $\mathcal{F}_{\text{SC}}^{G_n^{\text{CGO}}}$-hybrid model is outlined in Fig. 22 (Appendix B).

Let $G_n^{\text{CGO}} = (V_{\text{CGO}}, E_{\text{CGO}})$ be a graph with $|V_{\text{CGO}}| = n$ constructed as above. To realize wrapped $\mathcal{F}_{\text{AUTH}}^{V_{\text{CGO}},\mathsf{rnd}}$ over G_n^{CGO} with $\Pi^{\text{CGO}}_{\text{R-AUTH}}$, we define the *doom structure* $\mathscr{D}_{\text{CGO}}$ as follows:

- Let $D_{\text{CGO}}(T_i)$ be the set of all participants P such that $P \in T_i$ or at most $\frac{5}{6}$th fraction of P's helper committees are privileged. A committee is honest if at most $\frac{1}{4}$th fraction of its members are corrupted. Committees are categorized as privileged and unprivileged based on the $D_{\text{DPPU}}(\cdot)$ function defined over sets of committees (considering committees as super-nodes). $(T_i, D_i) \in \mathscr{D}_{\text{CGO}}$ if and only if $T_i \in \{S \subset V_{\text{CGO}} | \text{at most } \frac{n \log^k n}{4 \log(n \log^k n)} \text{ number of committees are not honest}\}$ (i.e, DPPU works at the committee level) and $D_i \subset D_{\text{CGO}}(T_i)$.

Chandran *et al.* proved that there exists constants $\alpha_{\text{CGO}}, \beta_{\text{CGO}}$ such that for any adversary corrupting a set T of size less than $\alpha_{\text{CGO}} n$, at most $\frac{n \log^k n}{4 \log(n \log^k n)}$ committees are not honest and $|D_{\text{CGO}}(T)| < \beta_{\text{CGO}} \frac{|T|}{\log n}$. For those constants we have the following statement.

Theorem 10. *Protocol $\Pi^{\text{CGO}}_{\text{R-AUTH}}$ UC-realizes $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{CGO}}}(\mathcal{F}_{\text{AUTH}}^{V_{\text{CGO}},\mathsf{rnd}})$ for some $\mathsf{rnd} \in O(\log n \log\log n)$, where n is the number of nodes, in the $\mathcal{F}_{\text{SC}}^{G_n^{\text{CGO}}}$-hybrid model against an adversary corrupting less than $\alpha_{\text{CGO}} n$ nodes.*

As before, we can also formulate the above result in terms of thresholds over the number of corrupted and doomed nodes, as stated in [CGO10] in the property-based setting:

Corollary 3. *There exists constants $\alpha_{\text{CGO}}, \beta_{\text{CGO}}$ such that over $G_n^{\text{CGO}} = (V_{\text{CGO}}, E_{\text{CGO}})$ with $|V_{\text{CGO}}| = n$ and in the presence of an adversary corrupting up to $t < \alpha_{\text{CGO}} n$ nodes, $\Pi^{\text{CGO}}_{\text{R-AUTH}}$ guarantees perfect reliable message transmission among all but $\beta_{\text{CGO}} \frac{t}{\log n}$ nodes.*

Graphs of logarithmic degree. An optimal transmission scheme over logarithmic-degree graphs has recently been proposed by Jayanti *et al.* in [JRV20]. Their graph construction is also randomized. Their graphs consist of $z = k \log n$ layers where k is a constant and n is the number of nodes. All layers are constructed using the same method but over a randomly permuted set of nodes. To form a layer, they arbitrarily partition the nodes into n/s committees of size $s = c \log\log n$ where c is a constant. Within each committee, they instantiate an UPFAL expander graph and then connect committees with DPPU butterfly graph using "super-edges." A super-edge is a perfect matching between set of nodes of two different committees. Let $\mathcal{G}_{\text{JRV}}$ be the family of graphs constructed as above.

The protocol they proposed, Π_{JRV}, for reliable message transmission over graphs in $\mathcal{G}_{\text{JRV}}$ goes as follows: To transmit a message from node S to node R, S sends the message through each layer separately and then R takes a simple majority over all the values received from all the layers. In each layer, if S and R are in the same committee they send the message by simply invoking UPFAL within the committee. Otherwise (i.e., S and R are located in different committees), S sends the message to all the nodes in its committee using UPFAL. Then S's committee sends the message to R's committee using DPPU over super-edges. Every node in R's committee (except R) sends the message to R using UPFAL. Finally, R takes the majority over all the incoming messages and considers it as the message received through that specific layer. We should note that to send a message over a super-edge, each node sends the message to its matched node and then every node in the destination committee sends the message to all the other nodes using UPFAL and finally each node locally takes the majority of received messages. (See [JRV20] for details.) Protocol $\Pi^{\text{JRV}}_{\text{R-AUTH}}$ based on the JRV transmission scheme in the $\mathcal{F}_{\text{SC}}^{G_n^{\text{JRV}}}$-hybrid model is outlined in Fig. 23 (Appendix B).

Let $G_n^{\text{JRV}} = (V_{\text{JRV}}, E_{\text{JRV}}) \in \mathcal{G}_{\text{JRV}}$ be a graph with $|V_{\text{JRV}}| = n$. In each layer of G_n^{JRV}, if a committee contains more than $\frac{1}{72}s$ corruptions (i.e., UPFAL does not work), they call it *bad* and if the total number of bad

[5] At a high level, differential agreement is a kind of agreement that guarantees that if many (not all) of the honest parties begin with the same value, all of the honest parties will output that value.

committees in the layer exceeds $\frac{n/s}{4\log(n/s)}$ (i.e., DPPU does not work) they call the layer bad. To realize wrapped $\mathcal{F}_{\text{AUTH}}^{V_{\text{JRV}},\mathsf{rnd}}$ over G_n^{JRV} with $\Pi_{\text{R-AUTH}}^{\text{JRV}}$, we define the *doom structure* $\mathscr{D}_{\text{JRV}}$ as follows:

- Let $D_{\text{JRV}}(T_i)$ be the set of all participants P such that $P \in T_i$ or P is doomed in more than $\frac{1}{10}z$ layers among all the good layers. A node is considered doomed in a layer if it is located in a doomed committee (wrt $D_{\text{DPPU}}(\cdot)$) or is doomed itself within its committee (wrt $D_{\text{UPFAL}}(\cdot)$). $(T_i, D_i) \in \mathscr{D}_{\text{JRV}}$ if and only if $T_i \in \{S \subset V_{\text{JRV}} | \text{at most } \frac{1}{5}\text{th of the layers are bad}\}$ and $D_i \subset D_{\text{JRV}}(T_i)$.

Jayanti *et al.* proved that there exists a graph $G_n^{\text{JRV}} \in \mathcal{G}_{\text{JRV}}$ and constants $\alpha_{\text{JRV}}, \beta_{\text{JRV}}$ such that for any adversary corrupting the set T of size less than $\alpha_{\text{JRV}} n$ nodes, at most $\frac{1}{5}$th of its layers are bad and $|D_{\text{JRV}}(T)| < \beta_{\text{JRV}} \frac{|T|}{\log n}$. For such a graph and constants we have the following statement.

Theorem 11. *Protocol $\Pi_{\text{R-AUTH}}^{\text{JRV}}$ UC-realizes $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{JRV}}}(\mathcal{F}_{\text{AUTH}}^{V_{\text{JRV}},\mathsf{rnd}})$ for some $\mathsf{rnd} \in O(\log n \cdot \log\log\log n)$, where n is the number of nodes, in the $\mathcal{F}_{\text{SC}}^{G_n^{\text{JRV}}}$-hybrid model against an adversary corrupting less than $\alpha_{\text{JRV}} n$ nodes.*

Again, in threshold terms, we obtain (cf [JRV20]):

Corollary 4. *There exists a graph G_n^{JRV} with n nodes and constants $\alpha_{\text{JRV}}, \beta_{\text{JRV}}$ such that in the presence of an adversary corrupting up to $t < \alpha_{\text{JRV}} n$ nodes, $\Pi_{\text{R-AUTH}}^{\text{JRV}}$ guarantees perfect reliable message transmission among all but $\beta_{\text{JRV}} \frac{t}{\log n}$ nodes.*

4.2 AE Remote SMT

Similarly to Section 3, we can use a perfect SMT protocol [DDWY90] over the set of paths provided by the DPPU transmission scheme to achieve *secure* communication. We stress, however, that the s paths from S to R are not necessarily the same (except reversed) as the s paths from R to S.

The protocol for remote SMT in the $(\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{DPPU}}}(\mathcal{F}_{\text{AUTH}}^{V_{\text{DPPU}},\mathsf{rnd}}), \mathcal{F}_{\text{R-SC}}^{G_{\text{DPPU}}})$-hybrid model is outlined in Fig 13.

Protocol $\Pi_{\text{R-SMT}}^{\text{DPPU}}$

1. Upon receiving input $(\textsc{Send}, sid, v)$ from $\mathcal{Z}$ in round ρ, where $sid = (S, R, sid')$ for $S, R \in V_{\text{DPPU}}$, the sender S executes protocol $\Pi_{\text{DDWY}}(\vec{\gamma}, \tau, v)$ with R, where the wires $\gamma_1, \ldots, \gamma_s$ in $\vec{\gamma}$ for communication from S to R are taken to be the s paths $\lambda_1, \ldots, \lambda_s$ from S to R (as specified by the DPPU transmission scheme), respectively. More precisely, in the first phase S and R use s different instances of $\mathcal{F}_{\text{R-SC}}^{G_{\text{DPPU}}}$ with SID's $sid_i = (\lambda_i, P)$ to send all the messages instead of using the wires in $\vec{\gamma}$. Next, in the second and third phases, the authenticated channel is substituted with separate instances of $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{DPPU}}}(\mathcal{F}_{\text{AUTH}}^{V_{\text{DPPU}},\mathsf{rnd}})$ with SID's $sid_{\text{AUTH}_1} = (R, S, 1, sid')$ and $sid_{\text{AUTH}_2} = (S, R, 2, sid')$, respectively, where rnd is the maximum length of any three-step path specified by the DPPU transmission scheme. To receive output from the aiding functionalities, S and R have to send $\textsc{Fetch}$ messages to the functionalities using the correct session IDs as generated above and in the correct rounds. In particular, the first-phase messages are fetched in rounds $\rho + l_i$ where l_i is the length of the i'th path from S to R, the second-phase message is sent in round $\rho + \ell$ (ℓ is the maximum value of l_i's), and the second-phase and third-phase messages are respectively fetched in rounds $\rho + \mathsf{rnd} + \ell$ and $\rho + 2 \cdot \mathsf{rnd} + \ell$.
2. Upon receiving input $(\textsc{Fetch}, sid)$ from $\mathcal{Z}$ in round $\rho + 2 \cdot \mathsf{rnd} + \ell$, R outputs $(\textsc{Sent}, sid, m')$ to $\mathcal{Z}$ if it receives m' as the output of this protocol.

Fig. 13. Perfect remote SMT protocol over G_{DPPU}

To realize wrapped $\mathcal{F}_{\text{SMT}}^{V_{\text{DPPU}},\mathsf{rnd}'}$ for some rnd' over G_{DPPU} with $\Pi_{\text{R-SMT}}^{\text{DPPU}}$, we use the same *doom structure* $\mathscr{D}_{\text{DPPU}}$ from Section 4.1. This result is formally stated in the following theorem.

Theorem 12. *Protocol $\Pi_{\text{R-SMT}}^{\text{DPPU}}$ UC-realizes $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{DPPU}}}(\mathcal{F}_{\text{SMT}}^{V_{\text{DPPU}},2\cdot\mathsf{rnd}+\ell})$ in the $(\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{DPPU}}}(\mathcal{F}_{\text{AUTH}}^{V_{\text{DPPU}},\mathsf{rnd}}), \mathcal{F}_{\text{R-SC}}^{G_{\text{DPPU}}})$-hybrid model against an adversary corrupting less than $s/4$ nodes where ℓ and s are the maximum length and the number of specified paths between pairs of nodes by DPPU transmission scheme, respectively.*

Proof. Let $\mathcal{A}$ be an adversary in the real world. We construct a simulator $\mathcal{S}$ in the ideal world, such that no environment can distinguish whether it is interacting with $\Pi^{\text{DPPU}}_{\text{R-SMT}}$ and $\mathcal{A}$, or with $\mathcal{W}^{\mathscr{D}_{\text{DPPU}}}_{\text{AE}}(\mathcal{F}^{V_{\text{DPPU}},2\cdot\mathsf{rnd}+\ell}_{\text{SMT}})$ and $\mathcal{S}$. The simulator $\mathcal{S}$ has similar structure as the simulator in the proof of Theorem 5. However, $\mathcal{S}$ needs to simulate different aiding functionalities for which the structure of the internal simulation is the same. Moreover, now $\mathcal{S}$ can doom S or R under a slightly different condition according to $\mathscr{D}_{\text{DPPU}}$. More specifically, the simulator in the proof of Theorem 5 needs to doom at least one of S and R when a majority of wires between them are corrupted (which is allowed by $\mathscr{D}_{\text{PSMT}}$) but $\mathcal{S}$ here can do that if at least $\frac{1}{8}$ of paths to Γ_{out} or from Γ_{in} are corrupted for one of S and R. As it is discussed in [DPPU86], whenever a majority of wires between S and R are corrupted at least $\frac{1}{8}$ of paths to Γ_{out} or from Γ_{in} are corrupted for one of S and R. Therefore, difference in the doom structures also does not affect correctness of the simulation.

$\Pi^{\text{DPPU}}_{\text{R-SMT}}$ is based on Π_{DDWY} which has exactly 3 rounds of communication. In the first round of communication, $\Pi^{\text{DPPU}}_{\text{R-SMT}}$ sends values using $\mathcal{F}^{G_{\text{DPPU}}}_{\text{R-SC}}$ over specified paths of length at most ℓ. Each of the remaining two rounds of communication consists of an invocation of $\mathcal{W}^{\mathscr{D}_{\text{DPPU}}}_{\text{AE}}(\mathcal{F}^{V_{\text{DPPU}},\mathsf{rnd}}_{\text{AUTH}})$ which takes rnd rounds. Therefore, $\Pi^{\text{DPPU}}_{\text{R-SMT}}$ takes $2\cdot\mathsf{rnd}+\ell$ rounds to terminate. □

Corollary 5. *Over a butterfly network of $n = m2^m$ nodes and in the presence of an adversary corrupting up to $t < 2^m/4$ nodes, $\Pi^{\text{DPPU}}_{\text{R-SMT}}$ guarantees (perfect) secure message transmission among all but at most $32t\log 16t$ nodes.*

Although the above technique helped us make the DPPU transmission scheme secure, it cannot in general be extended to other transmission schemes. In the above approach we need a majority of honest paths between any pair of privileged nodes to realize a secure link between them. Many transmission schemes such as UPFAL do not guarantee such a property for privileged nodes. To realize the SMT functionality using other transmission schemes, one approach is to use SMT-PD (Section 3.3) since having access to an authenticated channel, it only requires a single honest path between sender and receiver to establish a secure channel. This approach can be used to make any reliable message transmission scheme secure since these schemes realize authenticated channel and guarantee at least an honest path between any pair of privileged nodes. The only downside of using SMT-PD compared to the technique from [DDWY90] is that it only provides statistical security rather than perfect security.

The protocol for SMT-PD in the $(\mathcal{F}^{G_n}_{\text{R-SC}}, \mathcal{W}^{\mathscr{D}_{\text{SMT-PD}}}_{\text{AE}}(\mathcal{F}^{V,\mathsf{rnd}}_{\text{AUTH}}))$-hybrid model is presented in Fig. 14, using the following notation.

Protocol $\Pi_{\text{SMT-PD}}$

1. Upon receiving input $(\textsc{Send}, sid, v)$ from $\mathcal{Z}$ in round ρ, where $sid = (S, R, sid')$ for $S, R \in V$, the sender S executes protocol $\Pi_{\text{PUB-SMT}}(S, R, v, \mathcal{C})$ with the receiver R, where $\mathcal{C}$ is taken to be the set of specified paths $\gamma_1, \ldots, \gamma_s$ from S to R. More precisely, in the first phase, S uses s different instances of $\mathcal{F}^{G_n}_{\text{R-SC}}$ with SIDs $sid_i = (\gamma_i, sid')$ to send all the random bit strings instead of using channels in $\mathcal{C}$. Next, in the second, third, and fourth phases, S and R substitute the public channel with separate instances of $\mathcal{W}^{\mathscr{D}_{\text{SMT-PD}}}_{\text{AE}}(\mathcal{F}^{V,\mathsf{rnd}}_{\text{AUTH}})$ with SIDs $sid_2 = (S, R, 2, sid')$, $sid_3 = (R, S, 3, sid')$, and $sid_4 = (S, R, 4, sid')$, respectively. To receive output from the aiding functionalities, S and R have to send $\textsc{Fetch}$ messages to the functionalities using the correct session IDs as generated above and in the correct rounds. In particular, the first-phase messages are fetched in rounds $\rho + l_i$ where l_i is the length of γ_i, the second-phase message is sent in round $\rho + \ell$ (ℓ is maximum length of specified paths) and fetched in round $\rho + \ell + \mathsf{rnd}$, the third-phase message is sent in round $\rho + \ell + \mathsf{rnd}$ and fetched in round $\rho + \ell + 2\cdot\mathsf{rnd}$, and the fourth-phase message is sent in round $\rho + \ell + 2\cdot\mathsf{rnd}$ and fetched in round $\rho + \ell + 3\cdot\mathsf{rnd}$.
2. Upon receiving input $(\textsc{Fetch}, sid)$ from $\mathcal{Z}$ in round $\rho + \ell + 3\cdot\mathsf{rnd}$, R outputs $(\textsc{Sent}, sid, m')$ to $\mathcal{Z}$ if it receives m' as the output of this protocol.

Fig. 14. Remote SMT protocol with public discussion

Let $G_n = (V, E)$ be a graph with polynomially many paths of length at most ℓ specified between every pair of nodes. To realize wrapped $\mathcal{F}^{V,\mathsf{rnd}'}_{\text{SMT}}$ over G_n for some rnd' using $\Pi_{\text{SMT-PD}}$, we define the *doom structure* $\mathscr{D}_{\text{SMT-PD}}$ as follows:

- $(T_i, D_i) \in \mathscr{D}_{\text{SMT-PD}}$ if and only if $T_i, D_i \subset \mathcal{P}$ and at least one of the specified paths between any pair of nodes in $\mathcal{P} \setminus D_i$ is completely contained by $\mathcal{P} \setminus T_i$.

Theorem 13. *Define* $t = \max_{(T_i,D_i)\in\mathscr{D}_{\text{SMT-PD}}} |T_i|$, *then protocol* $\Pi_{\text{SMT-PD}}$ *statistically UC-realizes* $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{SMT-PD}}}(\mathcal{F}_{\text{SMT}}^{V,\ell+3\cdot\mathsf{rnd}})$ *in the* $(\mathcal{F}_{\text{R-SC}}^{G_n}, \mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{SMT-PD}}}(\mathcal{F}_{\text{AUTH}}^{V,\mathsf{rnd}}))$*-hybrid model against an adversary corrupting less than* t *nodes, where* ℓ *is the maximum length of specified paths between any pair of nodes.*

Proof. Let $\mathcal{A}$ be an adversary in the real world. We construct a simulator $\mathcal{S}$ in the ideal world, such that no unbounded environment can distinguish whether it is interacting with $\Pi_{\text{SMT-PD}}$ and $\mathcal{A}$ in the $(\mathcal{F}_{\text{R-SC}}^{G_n}, \mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{SMT-PD}}}(\mathcal{F}_{\text{AUTH}}^{V,\mathsf{rnd}}))$-hybrid world, or with $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{SMT-PD}}}(\mathcal{F}_{\text{SMT}}^{V,\ell+3\cdot\mathsf{rnd}})$ and $\mathcal{S}$ in the ideal world. The simulator $\mathcal{S}$ is very similar to the simulator that was constructed in the proof for $\Pi_{\text{SMT-PD}}(S, R, \vec{W}, l)$ from Section 3.3. However, $\mathcal{S}$ now interacts with a wrapped SMT functionality, and corruption messages for all parties are sent. Moreover, $\mathcal{S}$ needs to simulate different aiding functionalities, although the general structure of the internal simulation is the same. The other difference is that the case in which S and R (which were denoted P_s and P_r in Section 3.3) are not corrupted by $\mathcal{A}$ becomes more complicated, because we are now working with access to a *wrapped* AUTH functionality rather than an ideal authenticated channel. If either S or R becomes doomed according to the doom structure $\mathscr{D}_{\text{SMT-PD}}$ before the fourth phase, then $\mathcal{S}$ does not need to cheat in the fourth phase because s/he learns m and can simulate based on that, however, it does need to constantly influence the wrapper. The case that needs more contemplation happens when both S or R are privileged in the beginning of the fourth phase (when $\mathcal{S}$ is deciding about values S_i's) and then at least one of them becomes doomed (before the output is fetched). It is important because in this case $\mathcal{A}$ can learn the message by getting enough control over *wrapped* AUTH functionality to remove all the honest paths and to force the sender to send all the messages through corrupted paths. Since $\mathcal{S}$ has already chosen random S_i's, the message learnt by $\mathcal{A}$ may deviate from what is sent by S which causes $\mathcal{Z}$ distinguish the two worlds. In such a situation $\mathcal{S}$ also learns the correct message from the wrapper hence s/he can cheat by calculating fake pads (R_i''s) resulting in the right message and then simulate leaking fake pads from $\mathcal{F}_{\text{R-SC}}^{G_n}$. This way, $\mathcal{A}$ learns the message m correctly.

$\Pi_{\text{SMT-PD}}$ is based on $\Pi_{\text{PUB-SMT}}$ which has exactly 4 rounds of communication. In the first round of communication, $\Pi_{\text{SMT-PD}}$ sends values using $\mathcal{F}_{\text{R-SC}}^{G_n}$ over specified paths of length at most ℓ. Each of the remaining three rounds of communication consists of an invocation of $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{SMT-PD}}}(\mathcal{F}_{\text{AUTH}}^{V,\mathsf{rnd}})$ which takes rnd rounds. Therefore, $\Pi_{\text{SMT-PD}}$ takes $3 \cdot \mathsf{rnd} + \ell$ rounds to terminate. ☐

According to [DPPU86], all the realizable *doom structures* for AE *remote* RMT satisfy the condition in $\mathscr{D}_{\text{SMT-PD}}$. Basically, existence of at least one completely honest path between any pair of privileged nodes is guaranteed by the fact that corrupted nodes cannot disconnect any pair of privileged nodes. Therefore, $\Pi_{\text{SMT-PD}}$ can be used with any of the reliable transmission schemes in Section 4.1 to statistically realize *wrapped* SMT functionality over their graphs.

5 Almost-Everywhere Secure Computation

In this section, we consider general UC-secure computation in the almost-everywhere setting. We start by proving a composition theorem that shows how to compile a protocol Π realizing some functionality $\mathcal{F}$ with the help of several hybrids into an *almost-everywhere* version of Π, by wrapping each hybrid with a potentially different doom structure $\mathscr{D}_i$. These structures can be arbitrary, subject only to a certain monotonicity property, although they must correspond to the same participant set (indeed, composition would not make much sense otherwise); the compiled protocol is then shown to realize a *wrapped* version of $\mathcal{F}$, using a new doom structure $\mathscr{D}'$. In its full generality, our composition theorem is not restricted to security against only threshold adversaries, and the original protocol Π may itself realize a wrapped functionality associated with some doom structure $\mathscr{D}$. This latter fact, along with the fact that the monotonicity property carries over to the new doom structure $\mathscr{D}'$, make the compiled protocol readily amenable to further composition. We conclude by applying a special case of the composition theorem to obtain AE-MPC over the sparse graphs that were considered in Section 4. Rather than constructing protocols from scratch, we simply apply our generic AE compiler to replace the secure channels that are used in standard MPC protocols with AE-SMT, which we have already shown how to realize over these sparse graphs.

5.1 A General Composition Theorem

Let us first introduce some notation. Viewing a doom structure $\mathscr{D}$ (with participant set $\mathcal{P}$) as a binary relation over $2^{\mathcal{P}}$ and $2^{\mathcal{P}}$, denote by $\mathsf{Dom}(\mathscr{D})$ the set of values that appear as a first component in $\mathscr{D}$ (in other words, the set of all corruption sets allowed by $\mathscr{D}$). Say that $\mathscr{D}$ is *AE-monotone* if whenever $(T_i, D_i) \in \mathscr{D}$ and $T_i \subseteq T_j$ for $T_j \in \mathsf{Dom}(\mathscr{D})$, it holds that $(T_j, D_i) \in \mathscr{D}$. Different from the standard notion of monotonicity in the general adversary literature, AE-monotonicity captures the intuitive property that when additional parties are corrupted, parties that were previously doomed are still doomed (or newly corrupted). In fact, the work of [GO08] included a similar assumption: that the function mapping the set of corrupted parties to the corresponding set of unprivileged parties is monotonically increasing. It seems that AE-monotonicity is important for simulatability, since for example the simulator may want to make a doom request for a newly doomed party only after some additional parties are corrupted in the meantime, and in such a case the doom structure needs to admit that request. Fortunately, all of the doom structures that we consider are AE-monotone.

The AE compiler is shown in Fig. 15. It takes as input a protocol Π realizing some wrapped functionality $\mathcal{W}_{\text{AE}}^{\mathscr{D}}(\mathcal{F})$ in the $(\mathcal{F}_1, \ldots, \mathcal{F}_m)$-hybrid model and turns it into a protocol that works in the $(\mathcal{W}_{\text{AE}}^{\mathscr{D}_1}(\mathcal{F}_1), \ldots, \mathcal{W}_{\text{AE}}^{\mathscr{D}_m}(\mathcal{F}_m))$-hybrid model.

Compiler $\mathcal{C}^{\mathscr{D}_1,\ldots,\mathscr{D}_m}(\Pi)$

Apply the following modifications to protocol Π (which uses $\mathcal{F}_1, \ldots, \mathcal{F}_m$ as hybrids):

1. For each $i \in [m]$, instead of using $\mathcal{F}_i$, parties use $\mathcal{W}_{\text{AE}}^{\mathscr{D}_i}(\mathcal{F}_i)$ (which has the same input/output format to the parties).

Fig. 15. The AE compiler

Of course, the compiled protocol will not in general realize wrapped $\mathcal{F}$ with the same doom structure $\mathscr{D}$. In the following theorem, we construct a new doom structure $\mathscr{D}'$ representing the level of AE-security that is retained. Since we consider general adversaries, the compiled protocol can tolerate a set T' of corruptions only if T' can be tolerated by *all* of the assumed doom structures (i.e., $\mathscr{D}$ as well as $\mathscr{D}_1, \ldots, \mathscr{D}_m$). Furthermore, the set of parties in the compiled protocol that are considered doomed (relative to the corruptions in T') can consist of, roughly speaking, parties that are doomed with respect to *any* of the wrapped hybrids (such parties are collected in $D(T')$ below) or that would have been doomed in the original protocol Π (the parties denoted by A). In fact, since Π may already carry some level of AE-security, as captured by $\mathscr{D}$, we must expand the latter set to include parties that only become doomed when some or all of the parties in the former set are actually corrupted. This is crucial for our simulation strategy to work, and it explains why we require that $T' \cup D(T')$ is also tolerated by $\mathscr{D}$.

Theorem 14. *Let $\mathscr{D}, \mathscr{D}_1, \ldots, \mathscr{D}_m$ be AE-monotone doom structures over the same participant set $\mathcal{P}$. Define $\mathcal{T} := \mathsf{dom}(\mathscr{D})$, and $\mathcal{T}' := (\bigcap_{i=1}^m \mathsf{dom}(\mathscr{D}_i)) \cap \mathcal{T}$. For any $T' \in \mathcal{T}'$, define*

$$D(T') := \bigcup_{i=1}^{m} \left(\bigcup_{(T', D_j) \in \mathscr{D}_i} D_j \right).$$

Suppose that for all $T' \in \mathcal{T}'$, it holds that $T' \cup D(T') \in \mathcal{T}$. If protocol Π UC-realizes $\mathcal{W}_{\text{AE}}^{\mathscr{D}}(\mathcal{F})$ in the $(\mathcal{F}_1, \ldots, \mathcal{F}_m)$-hybrid model against a $\mathcal{T}$-adversary, then $\mathcal{C}^{\mathscr{D}_1,\ldots,\mathscr{D}_m}(\Pi)$ UC-realizes $\mathcal{W}_{\text{AE}}^{\mathscr{D}'}(\mathcal{F})$ in the $(\mathcal{W}_{\text{AE}}^{\mathscr{D}_1}(\mathcal{F}_1), \ldots, \mathcal{W}_{\text{AE}}^{\mathscr{D}_m}(\mathcal{F}_m))$-hybrid model against a $\mathcal{T}'$-adversary, where $\mathscr{D}'$ is defined as follows: For all $T' \in \mathcal{T}'$, we have $(T', D \cup A) \in \mathscr{D}'$ if $D \subseteq D(T')$ and $(T' \cup D(T'), A) \in \mathscr{D}$. Moreover, $\mathscr{D}'$ is AE-monotone.

Proof. We first prove that $\mathscr{D}'$ is AE-monotone. Suppose that $(T_i, D_i) \in \mathscr{D}'$ and $T_i \subseteq T_j$ for $T_j \in \mathcal{T}'$. This means that $D_i = D \cup A$ for some D, A such that $D \subseteq D(T_i)$ and $(T_i \cup D(T_i), A) \in \mathscr{D}$. We want to show that $(T_j, D_i) \in \mathscr{D}'$, and it suffices to show that $D \subseteq D(T_j)$ and $(T_j \cup D(T_j), A) \in \mathscr{D}$. Since $D(T_i) \subseteq D(T_j)$ (using the fact that $\mathscr{D}_1, \ldots, \mathscr{D}_m$ are all AE-monotone), it follows that $D \subseteq D(T_j)$. On the other hand, since

$T_i \cup D(T_i) \subseteq T_j \cup D(T_j)$, it follows that $(T_j \cup D(T_j), A) \in \mathscr{D}$ (using the fact that $\mathscr{D}$ is AE-monotone and that $T_j \cup D(T_j) \in \mathcal{T}$). We now prove the security of the compiled protocol.

Let $\mathcal{S}$ be a simulator (guaranteed to exist by the security of Π) such that no unbounded environment $\mathcal{Z}$ can distinguish whether it is interacting with Π and the dummy adversary $\mathcal{D}$ in the $(\mathcal{F}_1, \ldots, \mathcal{F}_m)$-hybrid world, or with $\mathcal{W}_{\text{AE}}^{\mathscr{D}}(\mathcal{F})$ and $\mathcal{S}$ in the ideal world. We use $\mathcal{S}$ to construct a simulator $\mathcal{S}'$ such that no unbounded environment $\mathcal{Z}'$ can distinguish whether it is interacting with $\mathcal{C}^{\mathscr{D}_1,\ldots,\mathscr{D}_m}(\Pi)$ and $\mathcal{D}$ in the $(\mathcal{W}_{\text{AE}}^{\mathscr{D}_1}(\mathcal{F}_1), \ldots, \mathcal{W}_{\text{AE}}^{\mathscr{D}_m}(\mathcal{F}_m))$-hybrid world, or with $\mathcal{W}_{\text{AE}}^{\mathscr{D}'}(\mathcal{F})$ and $\mathcal{S}'$ in the ideal world.

$\mathcal{S}'$ internally runs $\mathcal{S}$ and plays the role of the environment and $\mathcal{W}_{\text{AE}}^{\mathscr{D}}(\mathcal{F})$ for it. Inputs from $\mathcal{Z}'$ are forwarded to $\mathcal{S}$, with some additional processing. When $\mathcal{Z}'$ sends a corruption request directed to a party (i.e., telling $\mathcal{D}$ to corrupt a party directly), this is forwarded without modification. However, when $\mathcal{Z}'$ sends message delivery requests directed to an instance of $\mathcal{W}_{\text{AE}}^{\mathscr{D}_i}(\mathcal{F}_i)$ for some $i \in [m]$ (e.g., telling $\mathcal{D}$ to send a Corrupt or Influence message to that functionality), $\mathcal{S}'$ sends message delivery requests directed to a corresponding instance of $\mathcal{F}_i$, with the following exception: a request to deliver a Doom message is replaced by a request to deliver a Corrupt message if the doom structure $\mathscr{D}_i$ would accept it and is dropped otherwise.

Similarly, outputs from $\mathcal{S}$ are forwarded to $\mathcal{Z}'$, with some additional processing. Assuming that Π uses instances of $\mathcal{F}_1, \ldots, \mathcal{F}_m$ to handle all inter-party communication, note that these outputs should take the form of reports of incoming messages directed from either a party or an instance of an aiding functionality $\mathcal{F}_i$ to the dummy adversary for Π; thus, the processing done by $\mathcal{S}'$ is that reported messages from an instance of $\mathcal{F}_i$ are replaced by reported messages from an instance of $\mathcal{W}_{\text{AE}}^{\mathscr{D}_i}(\mathcal{F}_i)$. Finally, $\mathcal{S}'$ plays the role of $\mathcal{W}_{\text{AE}}^{\mathscr{D}}(\mathcal{F})$ by simply forwarding messages from $\mathcal{W}_{\text{AE}}^{\mathscr{D}'}(\mathcal{F})$ to $\mathcal{S}$ as if coming from $\mathcal{W}_{\text{AE}}^{\mathscr{D}}(\mathcal{F})$, and forwarding messages directed to $\mathcal{W}_{\text{AE}}^{\mathscr{D}}(\mathcal{F})$ (from $\mathcal{S}$) to $\mathcal{W}_{\text{AE}}^{\mathscr{D}'}(\mathcal{F})$, except that Corrupt messages for doomed parties (i.e., parties that $\mathcal{Z}'$ did not request to corrupt) are replaced by Doom messages. We emphasize in particular that Doom requests from $\mathcal{S}$ are forwarded without modification, which works because of the definition of $\mathscr{D}'$. It remains to reduce to the security of Π.

Assume for the sake of a contradiction that there is an environment $\mathcal{Z}'$ such that $\text{IDEAL}_{\mathcal{W}_{\text{AE}}^{\mathscr{D}'}(\mathcal{F}),\mathcal{S}',\mathcal{Z}'} \not\equiv \text{EXEC}_{\mathcal{C}^{\mathscr{D}_1,\ldots,\mathscr{D}_m}(\Pi),\mathcal{D},\mathcal{Z}'}$. Then, we construct an environment $\mathcal{Z}$ such that $\text{IDEAL}_{\mathcal{W}_{\text{AE}}^{\mathscr{D}}(\mathcal{F}),\mathcal{S},\mathcal{Z}} \not\equiv \text{EXEC}_{\Pi,\mathcal{D},\mathcal{Z}}$. The environment $\mathcal{Z}$ will simulate an interaction between $\mathcal{Z}'$ and $\mathcal{D}$, and output whatever $\mathcal{Z}'$ outputs, as well as do some additional processing that mimics the processing done by $\mathcal{S}'$. First, $\mathcal{Z}'$ is "activated" with $\mathcal{Z}$'s input z. Whenever $\mathcal{Z}'$ instructs its dummy adversary to deliver a message to an instance of an aiding functionality $\mathcal{W}_{\text{AE}}^{\mathscr{D}_i}(\mathcal{F}_i)$, this is translated by $\mathcal{Z}$ into a delivery request for a corresponding instance of $\mathcal{F}_i$ and forwarded to the external adversary (either $\mathcal{S}$ or $\mathcal{D}$), except that a request to deliver a Doom message is converted into a request to deliver a Corrupt message if allowed by $\mathscr{D}_i$ and dropped otherwise. Corruption requests directed to parties are forwarded to the external adversary unmodified.

Next, whenever $\mathcal{Z}$ receives subroutine output from the external adversary, this is forwarded to $\mathcal{Z}'$, except that reported messages from instances of $\mathcal{W}_{\text{AE}}^{\mathscr{D}_i}(\mathcal{F}_i)$ are translated into reported messages from corresponding instances of $\mathcal{F}_i$. Finally, $\mathcal{Z}$ simply relays inputs and outputs between $\mathcal{Z}'$ and parties. We now claim that $\text{IDEAL}_{\mathcal{W}_{\text{AE}}^{\mathscr{D}'}(\mathcal{F}),\mathcal{S}',\mathcal{Z}'} \equiv \text{IDEAL}_{\mathcal{W}_{\text{AE}}^{\mathscr{D}}(\mathcal{F}),\mathcal{S},\mathcal{Z}}$ and $\text{EXEC}_{\mathcal{C}^{\mathscr{D}_1,\ldots,\mathscr{D}_m}(\Pi),\mathcal{D},\mathcal{Z}'} \equiv \text{EXEC}_{\Pi,\mathcal{D},\mathcal{Z}}$. If $\mathcal{Z}$ interacts with $\mathcal{W}_{\text{AE}}^{\mathscr{D}}(\mathcal{F})$ and $\mathcal{S}$, then the view of the simulated $\mathcal{Z}'$ within $\mathcal{Z}$ is identical to the view of $\mathcal{Z}'$ when interacting with $\mathcal{W}_{\text{AE}}^{\mathscr{D}'}(\mathcal{F})$ and $\mathcal{S}'$, and similarly if $\mathcal{Z}$ interacts with Π and $\mathcal{D}$, then the view of the simulated $\mathcal{Z}'$ within $\mathcal{Z}$ is identical to the view of $\mathcal{Z}'$ when interacting with $\mathcal{C}^{\mathscr{D}_1,\ldots,\mathscr{D}_m}(\Pi)$ and $\mathcal{D}$. That concludes the proof. $\square$

In the specific case that Π realizes an *unwrapped* functionality $\mathcal{F}$ (indeed, one can always apply our AE wrapper to $\mathcal{F}$ with a doom structure of the form $\{(T_i, \emptyset)\}_i$, which is trivially AE-monotone, in order to obtain an equivalent functionality) in the $\mathcal{G}$-hybrid model against a threshold adversary, we obtain the following corollary, which requires some additional notation. Say that a doom structure $\mathscr{D}$ (with participant set $\mathcal{P}$) is *t-complete* if $\max_{(T_i,D_i)\in\mathscr{D}} |T_i| = t$, and $T \in \mathsf{Dom}(\mathscr{D})$ for all $T \subseteq \mathcal{P}$ with $|T| \leq t$ (in other words, if all possible sets of corruptions of size at most t are allowed by $\mathscr{D}$). Moreover, say that a doom structure $\mathscr{D}$ is *D-monotone* if whenever $(T_j, D_j) \in \mathscr{D}$ and $D_i \subseteq D_j$, it holds that $(T_j, D_i) \in \mathscr{D}$.

Corollary 6. *Let $\mathscr{D}$ be a t'-complete, D-monotone, and AE-monotone doom structure. Define* $t := \max_{|T'|=t'} \left| \left(\bigcup_{(T',D_i)\in\mathscr{D}} D_i \right) \cup T' \right|$. *If protocol Π UC-realizes $\mathcal{F}$ in the $\mathcal{G}$-hybrid model against a t-adversary, then $\mathcal{C}^{\mathscr{D}}(\Pi)$ UC-realizes $\mathcal{W}_{\text{AE}}^{\mathscr{D}}(\mathcal{F})$ in the $\mathcal{W}_{\text{AE}}^{\mathscr{D}}(\mathcal{G})$-hybrid model against a t'-adversary.*

Observe that t'-completeness enables the simulator to handle a threshold adversary that can corrupt *any* t' parties, and D-monotonicity enables the doom structure $\mathscr{D}$ that is used to wrap $\mathcal{G}$ to be preserved when wrapping $\mathcal{F}$. By construction, all of our doom structures satisfy these two properties. We remark that t has a very natural interpretation, namely the maximum number of parties that can become unprivileged (with respect to $\mathscr{D}$) when t' parties are corrupted.

5.2 AE-MPC

We now present our main result: how to achieve almost-everywhere MPC over several classes of sparse graphs in a composable manner. We assume a protocol that achieves "regular" MPC on a fully connected network of point-to-point secure channels, and show how to transform it into a protocol that achieves AE-MPC (with a lower corruption threshold) over a sparse graph with secure channels only between connected parties, using our AE compiler. To capture the MPC task, we use the functionality $\mathcal{F}_{\text{MPC}}^{f,\mathcal{P},\mathsf{rnd}}$ in Fig. 16, which is essentially Canetti's $\mathcal{F}_{\text{SFE}}$ [Can05] with synchrony.

Functionality $\mathcal{F}_{\text{MPC}}^{f,\mathcal{P},\mathsf{rnd}}$

The MPC functionality $\mathcal{F}_{\text{MPC}}$ is parameterized by a function $f : (\{0,1\}^* \cup \{\perp\})^n \times R \to (\{0,1\}^*)^n$, a participant set $\mathcal{P}$, and an integer rnd indicating the number of rounds that will be used to realize it, and it proceeds as follows. At the first activation, verify that $sid = (\mathcal{V}, sid')$, where $\mathcal{V}$ is an ordered set of n identities from $\mathcal{P}$, denoted $P_1, \ldots, P_n$; else halt. Initialize variables $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ to a default value $\perp$.

- Upon receiving input $(\textsc{InputF}, sid, v_i)$ from some $P_i \in \mathcal{V}$ in round ρ (which is the same for all P_i), set $x_i \leftarrow v_i$. If P_i is marked as corrupted, then send $(\textsc{InputLeakF}, sid, P_i, x_i)$ to the adversary; otherwise send $(\textsc{InputP}, sid, P_i)$.
- Upon receiving $(\textsc{InflInputF}, sid, P_i, x'_i)$ from the adversary for some $P_i \in \mathcal{V}$: If P_i is corrupted, and $(\textsc{OutputF}, sid, y_j)$ has not yet been sent to any $P_j \in \mathcal{V}$, then update $x_i \leftarrow x'_i$; otherwise, ignore the command.
- Upon receiving $(\textsc{InflOutputF}, sid, P_i, y'_i)$ from the adversary for some $P_i \in \mathcal{V}$, store y'_i.
- Upon receiving $(\textsc{Fetch}, sid)$ from some $P_i \in \mathcal{V}$ in round ρ': If P_i is corrupted, then send $(\textsc{FetchLeak}, sid, P_i)$ to the adversary; otherwise, if $\rho' = \rho + \mathsf{rnd}$, do the following:
  - If x_j has been set for all uncorrupted $P_j \in \mathcal{V}$, and no y_j has been set for any uncorrupted $P_j \in \mathcal{V}$, then choose $r \leftarrow R$ and set $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n; r)$.
  - Output $(\textsc{OutputF}, sid, y_i)$ to P_i if it has not yet been sent.
- Upon receiving $(\textsc{Output}, sid, P_i)$ from the adversary for some $P_i \in \mathcal{V}$: If P_i is corrupted, then output $(\textsc{OutputF}, sid, y'_i)$ to P_i if it has not yet been sent.
- Upon receiving $(\textsc{Corrupt}, sid, P_i)$ from the adversary for some $P_i \in \mathcal{V}$, mark P_i as corrupted and send $(\textsc{LeakF}, sid, P_i, x_i, y_i)$ to the adversary. Additionally leak any previous fetch requests made by P_i.

Fig. 16. Standard MPC functionality with synchrony

Although standard information-theoretic MPC protocols tolerating $t < \frac{n}{3}$ corruptions are known [BGW88,CCD88], they assume access to a broadcast channel, noting that broadcast can be achieved when $t < \frac{n}{3}$. However, [HZ10] showed that classical broadcast protocols are not adaptively secure in a simulation-based setting, and gave a VSS-based protocol that does in fact realize adaptively secure broadcast with perfect security for $t < \frac{n}{3}$, assuming only secure channels. Therefore, there exists a protocol that UC-realizes $\mathcal{F}_{\text{MPC}}^{f,\mathcal{P},\mathsf{rnd}}$ for any n-ary function f and some rnd in the $\mathcal{F}_{\text{SMT}}^{\mathcal{P},1}$-hybrid model, against an adversary corrupting less than $\frac{n}{3}$ parties. It is clear that this holds even in the $\mathcal{F}_{\text{SMT}}^{\mathcal{P},\ell}$-hybrid model, for arbitrary ℓ. Now, by invoking Corollary 6 (which of course also offers statistical security) and then applying the (regular) UC composition theorem in tandem with our results in Theorems 12 and 13 showing how to achieve AE-SMT over several classes of sparse graphs with either perfect or statistical security, we obtain the following corollaries showing how to achieve AE-MPC over those classes of graphs, with different combinations of parameters (recall that the maximum number of doomed nodes is encoded into each doom structure).

Corollary 7. *For any n-ary function f, there exists a protocol that UC-realizes $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{DPPU}}}(\mathcal{F}_{\text{MPC}}^{f,V_{\text{DPPU}},\mathsf{rnd}})$ in the $\mathcal{F}_{\text{SC}}^{G_n^{\text{DPPU}}}$-hybrid model against a t-adversary, for some* rnd *and $t \in O(\frac{n}{\log n})$.*

Corollary 8. *Let* $\mathtt{x} \in \{\text{UPFAL}, \text{CGO}, \text{JRV}\}$. *For any* n*-ary function* f*, there exists a protocol that statistically UC-realizes* $\mathcal{W}_{\mathrm{AE}}^{\mathscr{D}_{\mathtt{x}}}(\mathcal{F}_{\mathrm{MPC}}^{f,V_{\mathtt{x}},\mathsf{rnd}})$ *in the* $\mathcal{F}_{\mathrm{SC}}^{G_n^{\mathtt{x}}}$*-hybrid model against a* t*-adversary, for some* rnd *and* $t \in O(n)$.

References

ACd06. Saurabh Agarwal, Ronald Cramer, and Robbert de Haan. Asymptotically optimal two-round perfectly secure message transmission. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 394–408, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Heidelberg, Germany.

AOPC20. Bar Alon, Eran Omri, and Anat Paskin-Cherniavsky. Mpc with friends and foes. In *Annual International Cryptology Conference*, pages 677–706. Springer, 2020.

BCDH18. Elette Boyle, Ran Cohen, Deepesh Data, and Pavel Hubáček. Must the communication graph of mpc protocols be an expander? In *Annual International Cryptology Conference*, pages 243–272. Springer, 2018.

BCG21. Elette Boyle, Ran Cohen, and Aarushi Goel. Breaking the $O(\sqrt{n})$-bit barrier: Byzantine agreement with polylog bits per party. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, PODC'21, page 319–330, New York, NY, USA, 2021. Association for Computing Machinery.

BCH+20. Christian Badertscher, Ran Canetti, Julia Hesse, Björn Tackmann, and Vassilis Zikas. Universal composition with global subroutines: Capturing global setup within plain UC. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part III*, volume 12552 of *LNCS*, pages 1–30, Durham, NC, USA, November 16–19, 2020. Springer, Heidelberg, Germany.

BCL+11. Boaz Barak, Ran Canetti, Yehuda Lindell, Rafael Pass, and Tal Rabin. Secure computation without authentication. *J. Cryptol.*, 24(4):720–760, 2011.

BGW88. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC'88*, pages 1–10, 1988.

BMTZ17. Christian Badertscher, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. Bitcoin as a transaction ledger: A composable treatment. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 324–356, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

BPW03. Michael Backes, Birgit Pfitzmann, and Michael Waidner. A composable cryptographic library with nested operations. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *ACM CCS 2003*, pages 220–230, Washington, DC, USA, October 27–30, 2003. ACM Press.

BTFH+08. Zuzana Beerliová-Trubíniová, Matthias Fitzi, Martin Hirt, Ueli M. Maurer, and Vassilis Zikas. MPC vs. SFE: Perfect security in a unified corruption model. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 231–250, San Francisco, CA, USA, March 19–21, 2008. Springer, Heidelberg, Germany.

Can00. Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, January 2000.

Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145, Las Vegas, NV, USA, October 14–17, 2001. IEEE Computer Society Press.

Can05. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Cryptology ePrint Archive, Report 2000/067*, December 2005. Latest version at `http://eprint.iacr.org/2000/067/`.

CCD88. D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (abstract). In *STOC'88*, pages 11–19, 1988.

CDPW07. Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. Universally composable security with global setup. In Salil P. Vadhan, editor, *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, volume 4392 of *Lecture Notes in Computer Science*, pages 61–85. Springer, 2007.

CGO10. Nishanth Chandran, Juan A. Garay, and Rafail Ostrovsky. Improved fault tolerance and secure computation on sparse networks. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part II*, volume 6199 of *Lecture Notes in Computer Science*, pages 249–260. Springer, 2010.

CKKR19. Jan Camenisch, Stephan Krenn, Ralf Küsters, and Daniel Rausch. iuc: Flexible universal composability made simple. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 191–221. Springer, 2019.

DDWY90. D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly secure message transmission. In *FOCS'90*, pages 36–45, 1990.

Dol81. Danny Dolev. Unanimity in an unknown and unreliable environment. In *22nd FOCS*, pages 159–168, Nashville, TN, USA, October 28–30, 1981. IEEE Computer Society Press.

DPPU86. C. Dwork, D. Peleg, N. Pippenger, and E. Upfal. Fault tolerance in networks of bounded degree (preliminary version). In *STOC'86*, pages 370–379, 1986.

FG03. M. Fitzi and J. Garay. Efficient player-optimal protocols for strong and differential consensus. In *PODC'03*, pages 211–220, 2003.

FHM98. Matthias Fitzi, Martin Hirt, and Ueli M. Maurer. Trading correctness for privacy in unconditional multi-party computation (extended abstract). In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 121–136, Santa Barbara, CA, USA, August 23–27, 1998. Springer, Heidelberg, Germany.

GKZ10. Juan A Garay, Aggelos Kiayias, and Hong-Sheng Zhou. A framework for the sound specification of cryptographic tasks. In *2010 23rd IEEE Computer Security Foundations Symposium*, pages 277–289. IEEE, 2010.

GMW87. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229, New York City, NY, USA, May 25–27, 1987. ACM Press.

GO08. J. Garay and R. Ostrovsky. Almost-everywhere secure computation. In *EUROCRYPT'08*, pages 307–323, 2008.

GP92. Juan A. Garay and Kenneth J. Perry. A continuum of failure models for distributed computing. In *WDAG*, volume 647, pages 153–165, 1992.

Gri12. Jacopo Griggio. *Perfectly secure message transmission protocols with low communication overhead and their generalization*. PhD thesis, Cryptology, 2012.

HM97. Martin Hirt and Ueli M. Maurer. Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract). In James E. Burns and Hagit Attiya, editors, *16th ACM PODC*, pages 25–34, Santa Barbara, CA, USA, August 21–24, 1997. ACM.

HS15. Dennis Hofheinz and Victor Shoup. GNUC: A new universal composability framework. *J. Cryptol.*, 28(3):423–508, 2015.

HZ10. Martin Hirt and Vassilis Zikas. Adaptively secure broadcast. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 466–485, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany.

JRV20. Siddhartha Jayanti, Srinivasan Raghuraman, and Nikhil Vyas. Efficient constructions for almost-everywhere secure computation. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 159–183, Zagreb, Croatia, May 10–14, 2020. Springer, Heidelberg, Germany.

KMTZ13. Jonathan Katz, Ueli Maurer, Björn Tackmann, and Vassilis Zikas. Universally composable synchronous computation. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 477–498, Tokyo, Japan, March 3–6, 2013. Springer, Heidelberg, Germany.

KS08. K. Kurosawa and K. Suzuki. Truly efficient 2-round perfectly secure message transmission scheme. In *EUROCRYPT'08*, pages 324–340, 2008.

KS09. V. King and J. Saia. From almost everywhere to everywhere: Byzantine agreement with $\tilde{\mathrm{O}}(\mathrm{n}^{3/2})$ bits. In *DISC'09*, pages 464–478, 2009.

LPS86. Alexander Lubotzky, Ralph Phillips, and Peter Sarnak. Explicit expanders and the ramanujan conjectures. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 240–246, 1986.

LSP82. L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

MR11. Ueli Maurer and Renato Renner. Abstract cryptography. In Bernard Chazelle, editor, *Innovations in Computer Science - ICS 2011, Tsinghua University, Beijing, China, January 7-9, 2011. Proceedings*, pages 1–21. Tsinghua University Press, 2011.

Nie03. Jesper Buus Nielsen. *On Protocol Security in the Cryptographic Model*. PhD thesis, 2003.

PSL80. M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *JOURNAL OF THE ACM*, 27:228–234, 1980.

SAA96. Hasab Md. Sayeed and Hosame Abu-Amara. Efficient perfectly secure message transmission in synchronous networks. In *Information and Computation*, volume 126, pages 53–61, 1996.

SNR04. K. Srinathan, Arvind Narayanan, and C. Pandu Rangan. Optimal perfectly secure message transmission. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 545–561, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Heidelberg, Germany.

SZ16. Gabriele Spini and Gilles Zémor. Perfectly secure message transmission in two rounds. In *TCC (B1)*, pages 286–304. Springer, 2016.

Upf92. E. Upfal. Tolerating linear number of faults in networks of bounded degree. In *PODC'92*, pages 83–89, 1992.

Yao82. Andrew Chi-Chih Yao. Space-time tradeoff for answering range queries (extended abstract). In *14th ACM STOC*, pages 128–136, San Francisco, CA, USA, May 5–7, 1982. ACM Press.

A Further Related Work

History of SMT. The (perfectly) secure message transmission (SMT) problem was first studied by Dwork *et al.* [DDWY90], and we already introduced it in Section 1.1. Dwork *et al.* showed that 1-way SMT is possible if and only if $t < n/3$, and that 2-way SMT is possible if and only if $t < n/2$. (An SMT protocol is called 1-*way* if information flows only from the sender S to the receiver R, and 2-*way* if S and R are allowed to converse.) They gave both 2-phase and 3-phase protocols for 2-way SMT, where a *phase* is a flow of communication from S to R or vice versa, although their 2-phase solution is not efficient (polynomial-time). With foresight, we will be using their 3-phase protocol in our constructions. A rich line of follow-up works improving the efficiency of 2-phase SMT ensued, including work by Sayeed and Abu-Amara [SAA96], who gave a solution with transmission rate (total number of bits transmitted to the bit-size of the secret) $O(n^3)$, communication complexity $O(n^3 \log n)$, and polynomial computational costs; Srinathan *et al.* [SNR04] demonstrated a lower bound of $O(n)$ on the transmission rate; Agarwal *et al.* [ACd06] constructed a protocol with the optimal transmission rate, but the computational costs are exponential; while Kurosawa and Suzuki [KS08] gave a breakthrough result achieving an optimal transmission rate of $25n + o(n)$ with polynomial computational costs, while maintaining a communication complexity of $O(n^3 \log n)$ (a polynomial-time 2-phase SMT protocol). Griggio [Gri12] was able to reduce the transmission rate to $6n + o(n)$. Finally, Spini and Zémor [SZ16] further reduced the transmission rate to $5n + o(n)$, while also improving the communication complexity to $O(n^2 \log n)$.

Related models. We start with hybrid failure models (e.g., [GP92,FHM98]), which allow the adversary to maliciously corrupt some parties as well as cause another form of failure (e.g., passive or fail-stop corruption) to some other parties. Another relevant failure model is the one explored by Alon *et al.* [AOPC20], which technically considered two independent adversaries, one corrupting maliciously and the other one only passively. Their model is a special case of the adaptive model such that the malicious adversary can only corrupt statically before the protocol starts while the passive corruptions can be done at the end of the execution. In the AE setting, adversarial corruptions also have the effect of indirectly influencing the behavior of some of the honest parties (those who become "doomed"). The difference is that in our model, this other type of failure is defined structurally, based on the graph and the set of corruptions.

Also related is the work by King and Saia [KS09] (and follow-ups) who considered randomized Byzantine agreement over complete networks, but without all-to-all communication in order to improve the communication complexity. Their aim, however, is to still obtain full (not AE) agreement. The same approach is also explored by Boyle *et al.* [BCDH18] to reduce the communication cost of MPC over complete graphs. They investigated special characteristics (expansion) of the communication graph which is dynamically determined as a part of the protocol. In a recent follow-up, Boyle *et al.* [BCG21] defined an "almost-everywhere communication functionality" and used it as a hybrid in their low-communication Byzantine agreement protocols. However, this functionality is used to model a very specific communication tree, in which parties assigned to the root node can use the tree to send messages to all but a small fraction of the honest parties (called "isolated"), while the underlying model is still a complete network of point-to-point authenticated channels.

B Functionalities and Protocols

Protocol $\Pi_{\text{R-SC}}(G_n)$

1. Upon receiving input $(\textsc{Send}, sid, v)$ from $\mathcal{Z}$ in round ρ, where $sid = (S, P_1, \ldots, P_{k-1}, R, sid')$ and $(S, P_1, \ldots, P_{k-1}, R)$ is a path in G_n, the sender S sends $(\textsc{Send}, sid_1, v)$ to an instance of $\mathcal{F}_{\text{SC}}^{G_n}$ with SID $sid_1 = (S, P_1, 1, sid')$.
2. For each $i \in [k-1]$: Upon activation in round $\rho + i$, party P_i sends $(\textsc{Fetch}, sid_i)$ to $\mathcal{F}_{\text{SC}}^{G_n}$. Upon receiving back $(\textsc{Sent}, sid_i, m)$, P_i sends $(\textsc{Send}, sid_{i+1}, m)$ to an instance of $\mathcal{F}_{\text{SC}}^{G_n}$ with SID $sid_{i+1} = (P_i, P_j, i+1, sid')$, where $P_j = P_{i+1}$ if $i < k-1$ and $P_j = R$ if $i = k-1$.
3. Upon receiving input $(\textsc{Fetch}, sid)$ from $\mathcal{Z}$ in round $\rho + k$, the receiver R sends $(\textsc{Fetch}, sid_k)$ to $\mathcal{F}_{\text{SC}}^{G_n}$. Upon receiving back $(\textsc{Sent}, sid_k, m)$, R outputs $(\textsc{Sent}, sid, m)$ to $\mathcal{Z}$.

Fig. 19. Remote secure channel protocol for realizing $\mathcal{F}_{\text{R-SC}}^{G_n}$

Protocol $\Pi_{\text{WC}}(S, R, \vec{W})$

1. Upon receiving input $(\textsc{Send}, sid, W_i, v_i)$ from $\mathcal{Z}$ in round ρ (which is the same for all W_i), where $sid = (P_s, P_r, sid')$ for either $P_s = S$ and $P_r = R$ or $P_s = R$ and $P_r = S$, party P_s sends $(\textsc{Send}, sid_i^1, v_i)$ to an instance of $\mathcal{F}_{\text{SC}}^{S,R,\vec{W}}$ with $sid_i^1 = (P_s, W_i, sid)$.
2. Upon activation in round $\rho + 1$, each wire-party W_i sends $(\textsc{Fetch}, sid_i^1)$ to $\mathcal{F}_{\text{SC}}^{S,R,\vec{W}}$. Upon receiving back $(\textsc{Sent}, sid_i^1, m_i)$, W_i sends $(\textsc{Send}, sid_i^2, m_i)$ to an instance of $\mathcal{F}_{\text{SC}}^{S,R,\vec{W}}$ with $sid_i^2 = (W_i, P_r, sid)$.
3. Upon receiving input $(\textsc{Fetch}, sid, W_i)$ from $\mathcal{Z}$ in round $\rho+2$, party P_r sends $(\textsc{Fetch}, sid_i^2)$ to $\mathcal{F}_{\text{SC}}^{S,R,\vec{W}}$. Upon receiving back $(\textsc{Sent}, sid_i^2, m_i')$, P_r outputs $(\textsc{Sent}, sid, W_i, m_i')$ to $\mathcal{Z}$.

Fig. 17. Wire communication protocol

Protocol $\Pi_{\text{AUTH}}(S, R, \vec{W})$

1. Upon receiving input $(\textsc{Send}, sid, v)$ from $\mathcal{Z}$ in round ρ, where $sid = (P_s, P_r, sid')$ for either $P_s = S$ and $P_r = R$ or $P_s = R$ and $P_r = S$, party P_s sends $(\textsc{Send}, sid_{\text{AUTH}}, W_i, v)$ for each wire-party W_i to a single instance of $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ with SID $sid_{\text{AUTH}} = (sid, \textsc{auth})$.
2. Upon receiving input $(\textsc{Fetch}, sid)$ from $\mathcal{Z}$ in round $\rho + 2$, party P_r sends $(\textsc{Fetch}, sid_{\text{AUTH}}, W_i)$ for each W_i to $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$. Upon receiving back $(\textsc{Sent}, sid_{\text{AUTH}}, W_i, m_i)$ for each W_i, P_r takes a simple majority of the m_i's. More precisely, after receiving at least $\lfloor \frac{n}{2} \rfloor + 1$ copies of some message m' corresponding to different wire-parties, P_r outputs $(\textsc{Sent}, sid, m')$ to $\mathcal{Z}$. (If not enough copies were received, e.g. because P_s was corrupted, then P_r outputs $\perp$.)

Fig. 18. Reliable message transmission protocol in the wire-party model

Protocol $\Pi_{\text{R-AUTH}}^{\text{DPPU}}$

1. Upon receiving input $(\textsc{Send}, sid, v)$ from $\mathcal{Z}$ in round ρ, where $sid = (S, R, sid')$ for $S, R \in V_{\text{DPPU}}$, the sender S sends $(\textsc{Send}, sid_i, v)$ to an instance of $\mathcal{F}_{\text{R-SC}}^{G_{\text{DPPU}}}$ with SID $sid_i = (\gamma_i, sid')$ for each of the paths $\gamma_1, \ldots, \gamma_s$ from S to R as specified by the DPPU transmission scheme.
2. For each $i \in [s]$: Upon activation in round $\rho + l_i$, where l_i is the length of path γ_i, the receiver R sends $(\textsc{Fetch}, sid_i)$ to $\mathcal{F}_{\text{R-SC}}^{G_{\text{DPPU}}}$. Upon receiving back $(\textsc{Sent}, sid_i, m_i)$, R stores m_i as the value received on path γ_i.
3. Upon receiving input $(\textsc{Fetch}, sid)$ from $\mathcal{Z}$ in round $\rho + \mathsf{rnd}$, where rnd is the maximum length of *any* three-step path (i.e., not necessarily one from S to R) specified by the DPPU transmission scheme, R takes a simple majority of the stored m_i's. More precisely, after receiving at least $\lfloor \frac{s}{2} \rfloor + 1$ copies of the same message m', R outputs $(\textsc{Sent}, sid, m')$ to $\mathcal{Z}$. (If not enough copies were received, then R outputs $\perp$.)

Fig. 20. Remote RMT protocol based on DPPU transmission scheme

Protocol $\Pi_{\text{R-AUTH}}^{\text{UPFAL}}$

1. Upon receiving input $(\textsc{Send}, sid, v)$ from $\mathcal{Z}$ in round ρ, where $sid = (S, R, sid')$ for $S, R \in V_{\text{UPFAL}}$, the sender S executes protocol $\Pi_{\text{UPFAL}}(S, R, v)$ with the receiver R, where sending a message from node to node is replaced by separate invocations to $\mathcal{F}_{\text{SC}}^{G_n^{\text{UPFAL}}}$ (note that we do not use $\mathcal{F}_{\text{R-SC}}^{G_n^{\text{UPFAL}}}$ here, because Π_{UPFAL} actually requires appending to the message as it travels along a path to R). To receive output from the instances of $\mathcal{F}_{\text{SC}}^{G_n^{\text{UPFAL}}}$, all nodes involved have to send $\textsc{Fetch}$ messages in the correct rounds.
2. Upon receiving input $(\textsc{Fetch}, sid)$ from $\mathcal{Z}$ in round $\rho + \mathsf{rnd}$, where rnd is the maximum length of any path used in the protocol, R outputs $(\textsc{Sent}, sid, m')$ if it receives m' as the output of this protocol.

Fig. 21. Remote RMT protocol based on UPFAL transmission scheme

Protocol $\Pi^{\text{CGO}}_{\text{R-AUTH}}$

1. Upon receiving input (Send, sid, v) from $\mathcal{Z}$ in round ρ, where $sid = (S, R, sid')$ for $S, R \in V_{\text{CGO}}$, the sender S executes protocol $\Pi_{\text{CGO}}(S, R, v)$ with the receiver R, where sending a message from node to node is replaced by separate invocations to $\mathcal{F}_{\text{SC}}^{G_n^{\text{CGO}}}$. To receive output from the instances of $\mathcal{F}_{\text{SC}}^{G_n^{\text{CGO}}}$, all nodes involved have to send Fetch messages in the correct rounds.
2. Upon receiving input (Fetch, sid) from $\mathcal{Z}$ in round $\rho + \mathsf{rnd}$, where rnd is the maximum number of rounds required by DPPU transmission scheme over committees multiplied by the maximum number of rounds required by differential agreement inside the committees, R outputs (Sent, sid, m') if it receives m' as the output of this protocol.

Fig. 22. Remote RMT protocol based on CGO transmission scheme

Protocol $\Pi^{\text{JRV}}_{\text{R-AUTH}}$

1. Upon receiving input (Send, sid, v) from $\mathcal{Z}$ in round ρ, where $sid = (S, R, sid')$ for $S, R \in V_{\text{JRV}}$, the sender S executes protocol $\Pi_{\text{JRV}}(S, R, v)$ with the receiver R, where sending a message from node to node is replaced by separate invocations to $\mathcal{F}_{\text{SC}}^{G_n^{\text{JRV}}}$. To receive output from the instances of $\mathcal{F}_{\text{SC}}^{G_n^{\text{JRV}}}$, all nodes involved have to send Fetch messages in the correct rounds.
2. Upon receiving input (Fetch, sid) from $\mathcal{Z}$ in round $\rho + \mathsf{rnd}$, where rnd is the maximum number of rounds required by DPPU transmission scheme over committees multiplied by the maximum number of rounds required by UPFAL transmission scheme inside the committees, R outputs (Sent, sid, m') if it receives m' as the output of this protocol.

Fig. 23. Remote RMT protocol based on JRV transmission scheme

C Proofs

Proof of Theorem 1. Let $\mathcal{A}$ be an adversary in the real world. We construct a simulator $\mathcal{S}$ in the ideal world, such that no environment can distinguish whether it is interacting with $\Pi_{\text{WC}}(S, R, \vec{W})$ and $\mathcal{A}$, or with $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ and $\mathcal{S}$. The simulator internally runs a copy of $\mathcal{A}$, and plays the roles of $\mathcal{F}_{\text{SC}}^{S,R,\vec{W}}$ and the parties in a simulated execution of the protocol. All inputs from $\mathcal{Z}$ are forwarded to $\mathcal{A}$, and all outputs from $\mathcal{A}$ are forwarded to $\mathcal{Z}$. Moreover, whenever $\mathcal{A}$ corrupts a party in the simulation, $\mathcal{S}$ corrupts the same party in the ideal world by interacting with $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$, and if the corruption was direct (i.e., not via $\mathcal{F}_{\text{SC}}^{S,R,\vec{W}}$), then $\mathcal{S}$ sends $\mathcal{A}$ the party's state and thereafter follows $\mathcal{A}$'s instructions for that party. The simulated execution starts upon $\mathcal{S}$ receiving $(\text{SendLeak}, sid, W_i, \hat{m_i})$ from $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ in round ρ for $sid = (P_s, P_r, sid')$, where $\hat{m_i} \in \{m_i, l(m_i)\}$ and m_i is the message to be sent through wire-party W_i, and it involves simulating P_s sending m_i to W_i through an instance of $\mathcal{F}_{\text{SC}}^{S,R,\vec{W}}$ (i.e., by simulating leakage from $\mathcal{F}_{\text{SC}}^{S,R,\vec{W}}$ to $\mathcal{A}$, and responding to corruption and influence requests directed from $\mathcal{A}$ to that functionality). Note that while $\mathcal{S}$ does not know m_i when P_s, P_r, and W_i are all honest, this is not a problem because in this case the real-world adversary only obtains $l(m_i)$ from $\mathcal{F}_{\text{SC}}^{S,R,\vec{W}}$. Messages to be sent by P_s through other wire-parties in round ρ are simulated in the same way. Next, in round $\rho + 1$, $\mathcal{S}$ simulates W_i fetching from $\mathcal{F}_{\text{SC}}^{S,R,\vec{W}}$ and then forwarding the obtained value to P_r, by once again playing the role of $\mathcal{F}_{\text{SC}}^{S,R,\vec{W}}$ for $\mathcal{A}$.

Finally, we describe how $\mathcal{S}$ simulates P_r's response to a Fetch input from $\mathcal{Z}$ in round $\rho + 2$. If P_r is corrupted by $\mathcal{A}$, then $\mathcal{S}$ can wait to receive $(\text{FetchLeak}, sid, W_i)$ from $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$, upon which it leaks the fetch to $\mathcal{A}$ if P_r was corrupted directly, and then sends InflSend and Output messages (for wire-party W_i) to $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ as appropriate. Otherwise, if P_s or W_i is corrupted by $\mathcal{A}$, then $\mathcal{S}$ influences $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ (for wire-party W_i) every time the value that would be fetched from $\mathcal{F}_{\text{SC}}^{S,R,\vec{W}}$ by the simulated P_r changes, e.g. due to $\mathcal{A}$'s influencing of $\mathcal{F}_{\text{SC}}^{S,R,\vec{W}}$ (note that this might occur in round ρ). If none of P_s, P_r, and W_i are corrupted by $\mathcal{A}$, then $\mathcal{S}$ can simply let the dummy P_r fetch from $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ when instructed by $\mathcal{Z}$, because in this case the real-world adversary cannot prevent P_r from fetching the actual message to be sent through W_i. It is easy to see that this simulation is perfect. □

Proof of Theorem 2. Let $\mathcal{A}$ be an adversary in the real world. We construct a simulator $\mathcal{S}$ in the ideal world, such that no environment can distinguish whether it is interacting with $\Pi_{\text{AUTH}}(S, R, \vec{W})$ and $\mathcal{A}$, or with $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}}$ and $\mathcal{S}$. The simulator internally runs a copy of $\mathcal{A}$, and plays the roles of $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ and the parties in a simulated execution of the protocol. All inputs from $\mathcal{Z}$ are forwarded to $\mathcal{A}$, and all outputs from $\mathcal{A}$ are forwarded to $\mathcal{Z}$. Moreover, whenever $\mathcal{A}$ corrupts a party in the simulation, $\mathcal{S}$ corrupts the same party in the ideal world by interacting with $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}}$ (except if the party is a wire-party), and if the corruption was direct (i.e., not via $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$), then $\mathcal{S}$ sends $\mathcal{A}$ the party's state and thereafter follows $\mathcal{A}$'s instructions for that party. The simulated execution starts upon $\mathcal{S}$ receiving $(\textsc{SendLeak}, sid, m)$ from $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}}$ in round ρ for $sid = (P_s, P_r, sid')$, and it involves simulating P_s sending m to P_r through the n wire-parties via a single instance of $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ (i.e., by simulating leakage from $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ to $\mathcal{A}$, and responding to corruption and influence requests directed from $\mathcal{A}$ to that functionality).

Next, we describe how $\mathcal{S}$ simulates P_r's response to a $\textsc{Fetch}$ input from $\mathcal{Z}$ in round $\rho + 2$. If P_r is corrupted by $\mathcal{A}$, then $\mathcal{S}$ can wait to receive $(\textsc{FetchLeak}, sid)$ from $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}}$, upon which it does the following. If the corruption was not direct, then $\mathcal{S}$ sends an $\textsc{InflSend}$ message to $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}}$ with the value that the real-world P_r would have output after fetching n values from $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ (in particular, $\mathcal{S}$ takes into account any $\textsc{InflSend}$ messages sent by $\mathcal{A}$ to the simulated instance of $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$), before sending an $\textsc{Output}$ message; if the corruption was in fact direct, then $\mathcal{S}$ simulates P_r reporting to $\mathcal{A}$ that a $\textsc{Fetch}$ input from $\mathcal{Z}$ was received, and then sends appropriate $\textsc{InflSend}$ and $\textsc{Output}$ messages to $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}}$ once $\mathcal{A}$ instructs P_r to output something to $\mathcal{Z}$ (recall that in this case, the simulated P_r follows the instructions of $\mathcal{A}$ each time it is activated). If P_r is not corrupted by $\mathcal{A}$, but P_s is, then $\mathcal{S}$ influences $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}}$ every time the value that the real-world P_r would have output changes (this might happen after $\mathcal{A}$ influences the simulated instance of $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$, or, in the case that P_s was corrupted directly, after $\mathcal{A}$ instructs the simulated P_s to send a different message via the instance of $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$). Finally, if neither P_s nor P_r is corrupted, then $\mathcal{S}$ can simply let the dummy P_r fetch from $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}}$ when instructed by $\mathcal{Z}$, because the assumption that $\mathcal{A}$ corrupts only a minority of the wire-parties implies that the real-world P_r receives enough copies of P_s's input $m = v$. Note that in this case, the dummy P_r immediately outputs the fetched value to $\mathcal{Z}$, which is fine because the real-world P_r cannot be corrupted in the time between receiving a $\textsc{Fetch}$ input from $\mathcal{Z}$ and outputting to $\mathcal{Z}$, since the activations alternate between P_r and the instance of $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$. It is easy to see that this simulation is perfect. □

Proof of Theorem 8 Let $\mathcal{A}$ be an adversary in the real world. We construct a simulator $\mathcal{S}$ in the ideal world, such that no environment can distinguish whether it is interacting with $\Pi_{\text{R-AUTH}}^{\text{DPPU}}$ and $\mathcal{A}$ in the $\mathcal{F}_{\text{R-SC}}^{G_{\text{DPPU}}}$-hybrid world, or with $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{DPPU}}}(\mathcal{F}_{\text{AUTH}}^{V_{\text{DPPU}},\mathsf{rnd}})$ and $\mathcal{S}$ in the ideal world. The simulator internally runs a copy of $\mathcal{A}$, and plays the roles of $\mathcal{F}_{\text{R-SC}}^{G_{\text{DPPU}}}$ and the parties in a simulated execution of the protocol, which starts when $\mathcal{S}$ receives $(\textsc{SendLeak}, sid, m)$ from the wrapper. Whenever $\mathcal{A}$ corrupts a party in the simulated execution, $\mathcal{S}$ corrupts the same party in the ideal world, and as a result $\mathcal{S}$ is able to influence the wrapper with the appropriate value when S or R is corrupted by $\mathcal{A}$. If S and R are not corrupted by $\mathcal{A}$, but at least one of S and R has more than $\frac{1}{8}$th fraction of paths to Γ_{out} or from Γ_{in} corrupted, then $\mathcal{S}$ can still influence the wrapper because in this case $\mathcal{S}$ can doom at least one of S and R according to $\mathscr{D}_{\text{DPPU}}$. The only case in which $\mathcal{S}$ cannot influence is when both S and R are privileged which means they have less than $\frac{1}{8}$ of paths to Γ_{out} and from Γ_{in} corrupted. However, it follows from the results in [DPPU86] that $\mathcal{A}$ also cannot influence the value recovered by R in this case, so $\mathcal{S}$ can simply let the dummy R fetch from the wrapper when instructed by $\mathcal{Z}$.

All the paths specified by DPPU transmission scheme over the butterfly network have length of $O(\log n)$. Since $\Pi_{\text{R-AUTH}}^{\text{DPPU}}$ transmits the message form S to R by sending it through the specified paths, its execution requires only $\mathsf{rnd} \in O(\log n)$ rounds. □

Proof of Theorem 9 Let $\mathcal{A}$ be an adversary in the real world. We construct a simulator $\mathcal{S}$ in the ideal world, such that no environment can distinguish whether it is interacting with $\Pi_{\text{R-AUTH}}^{\text{UPFAL}}$ and $\mathcal{A}$ in the $\mathcal{F}_{\text{SC}}^{G_n^{\text{UPFAL}}}$-hybrid world, or with $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{UPFAL}}}(\mathcal{F}_{\text{AUTH}}^{V_{\text{UPFAL}},\mathsf{rnd}})$ and $\mathcal{S}$ in the ideal world. The simulator internally runs a copy of $\mathcal{A}$, and plays the roles of $\mathcal{F}_{\text{SC}}^{G_n^{\text{UPFAL}}}$ and the parties in a simulated execution of the protocol, which starts when $\mathcal{S}$ receives (SENDLEAK, sid, m) from the wrapper. Whenever $\mathcal{A}$ corrupts a party in the simulated execution, $\mathcal{S}$ corrupts the same party in the ideal world, and as a result $\mathcal{S}$ is able to influence the wrapper with the appropriate value when S or R is corrupted by $\mathcal{A}$. If S and R are not corrupted by $\mathcal{A}$, but at least one of S and R is returned by $\mathscr{D}_{\text{UPFAL}}(T)$, then $\mathcal{S}$ can still influence the wrapper because in this case $\mathcal{S}$ can doom at least one of S and R according to $\mathscr{D}_{\text{UPFAL}}$. The only case in which $\mathcal{S}$ cannot influence is when both S and R are privileged which means that they are not returned by $D_{\text{UPFAL}}(T)$. However, it follows from the results in [Upf92] that $\mathcal{A}$ also cannot influence the value recovered by R in this case, so $\mathcal{S}$ can simply let the dummy R fetch from the wrapper when instructed by $\mathcal{Z}$.

Diameter of an expander graph is of $O(\log n)$. Since we are working on expander graphs and UPFAL transmission scheme uses only simple paths between S and R, all the messages are received by the receiver in $O(\log n)$ rounds. Therefore, $\Pi_{\text{R-AUTH}}^{\text{UPFAL}}$ requires $\mathsf{rnd} \in O(\log n)$ rounds to terminate. □

Proof of Theorem 10 Let $\mathcal{A}$ be an adversary in the real world. We construct a simulator $\mathcal{S}$ in the ideal world, such that no environment can distinguish whether it is interacting with $\Pi_{\text{R-AUTH}}^{\text{CGO}}$ and $\mathcal{A}$, or with $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{CGO}}}(\mathcal{F}_{\text{AUTH}}^{V_{\text{CGO}},\mathsf{rnd}})$ and $\mathcal{S}$. The simulator internally runs a copy of $\mathcal{A}$, and plays the roles of $\mathcal{F}_{\text{SC}}^{G_n^{\text{CGO}}}$ and the parties in a simulated execution of the protocol, which starts when $\mathcal{S}$ receives (SENDLEAK, sid, m) from the wrapper. Whenever $\mathcal{A}$ corrupts a party in the simulated execution, $\mathcal{S}$ corrupts the same party in the ideal world, and as a result $\mathcal{S}$ is able to influence the wrapper with the appropriate value when S or R is corrupted by $\mathcal{A}$. If S and R are not corrupted by $\mathcal{A}$, but at most $\frac{5}{6}$th fraction of helpers of S or R are privileged, then $\mathcal{S}$ can still influence the wrapper because in this case $\mathcal{S}$ can doom at least one of S and R according to doom structure $\mathscr{D}_{\text{CGO}}$. The only case that $\mathcal{S}$ cannot influence is when both S and R are privileged which means more than $\frac{5}{6}$th fraction of their helpers are privileged. As it is shown in [CGO10], $\mathcal{A}$ also cannot influence the communication when both S and R are privileged so $\mathcal{S}$ does not need to influence in the ideal world in that situation and can simply let the dummy R fetch from the wrapper.

Each transmission over super-edges consists of some parallel instances of $\mathcal{F}_{\text{SC}}^{G_n^{\text{CGO}}}$ (between corresponding nodes) followed by an execution of differential agreement inside the destination committee. According to [FG03], deterministic differential agreement requires at most linear number of rounds. Since in CGO transmission scheme committees are of size $O(\log\log n)$, the number of rounds required by each super-edge transmission is $O(\log\log n)$. We know in CGO transmission scheme, there are $n\log^k n$ committees communicating through DPPU transmission scheme over super-edges. We also discussed earlier that DPPU transmission scheme requires logarithmic number of rounds. therefore, the total number of rounds required by $\Pi_{\text{R-AUTH}}^{\text{CGO}}$ is $O\left(\log\left(n\log^k n\right)\log\log n\right) = O(\log n \cdot \log\log n)$. □

Proof of Theorem 11 Let $\mathcal{A}$ be an adversary in the real world. We construct a simulator $\mathcal{S}$ in the ideal world, such that no environment can distinguish whether it is interacting with $\Pi_{\text{R-AUTH}}^{\text{JRV}}$ and $\mathcal{A}$, or with $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{JRV}}}(\mathcal{F}_{\text{AUTH}}^{V_{\text{JRV}},\mathsf{rnd}})$ and $\mathcal{S}$. The simulator internally runs a copy of $\mathcal{A}$, and plays the roles of $\mathcal{F}_{\text{SC}}^{G_n^{\text{JRV}}}$ and the parties in a simulated execution of the protocol, which starts when $\mathcal{S}$ receives (SENDLEAK, sid, m) from the wrapper. Whenever $\mathcal{A}$ corrupts a party in the simulated execution, $\mathcal{S}$ corrupts the same party in the ideal world, and as a result $\mathcal{S}$ is able to influence the wrapper with the appropriate value when S or R is corrupted by $\mathcal{A}$. If S and R are not corrupted by $\mathcal{A}$, but S or R are doomed in at least $\frac{1}{10}z$ number of good layers, then $\mathcal{S}$ can still influence the wrapper because in this case $\mathcal{S}$ can doom at least one of S and R according to doom structure $\mathscr{D}_{\text{JRV}}$. The only case $\mathcal{S}$ cannot influence is when both S and R are privileged which means they are honest and doomed in at most $\frac{1}{10}z$ number of good layers. As it is shown in [JRV20], $\mathcal{A}$ also cannot influence the communication when both S and R are privileged, so $\mathcal{S}$ does not need to influence in the ideal world in that situation and can let the dummy R fetch from the wrapper.

Each transmission over super-edges consists of some parallel instances of $\mathcal{F}_{\text{SC}}^{G_n^{\text{JRV}}}$ (between corresponding nodes) followed by some parallel executions of UPFAL transmission scheme inside the destination committee. As discussed earlier, UPFAL transmission scheme requires logarithmic number of rounds. Since in JRV transmission scheme each committee has size of $s = O(\log\log n)$, each super-edge transmission takes $O(\log\log\log n)$ rounds. We know in JRV transmission scheme, there are n/s committees communicating through DPPU transmission scheme over super-edges. We also discussed earlier that DPPU transmission scheme requires logarithmic number of rounds. Therefore, the total number of rounds required by $\Pi^{\text{JRV}}_{\text{R-AUTH}}$ is $O\left(\log(n/s)\log\log\log n\right) = O(\log n \cdot \log\log\log n)$. $\square$