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]Georgia Tech, vzikas@gatech.edu

November 23, 2026

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.

Keywords: Secure multi-party computation, Universal Composability, Almost-Everywhere secure computation, Sparse graphs, Secure message transmission.

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).

*A preliminary version of this work appeared at ITC 2022 [19]. The full version was published in the Journal of Cryptology in 2026 [18].

†Work partially supported by NSF grants CNS-2001082 and CNS-2055694.

‡Supported in part by DARPA under Cooperative Agreement HR0011-20-2-0025, NSF grant CNS-2001096, US-Israel BSF grant 2015782, Cisco Research Award, Google Faculty Award, JP Morgan Faculty Award, IBM Faculty Research Award, Xerox Faculty Research Award, OKAWA Foundation Research Award, B. John Garrick Foundation Award, Teradata Research Award, Lockheed-Martin Research Award and Sunday Group. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, the Department of Defense, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes not withstanding any copyright annotation therein.

§Research supported in part by NSF grants CNS-2055599 and CNS-2001096, BSF grant 2020277, and by Sunday Group.

Despite great progress on the problem since it was first introduced and proven feasible [50, 33, 10, 22] involving hundreds, if not thousands, of publications 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, 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 graph, with some nodes even being routing nodes.

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* [8]: 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 $O(t)$ 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 [25]. The result holds even for the *reliable message transmission* (RMT) problem, in which only correctness is required.

That 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 [31] introduced the properties of *almost-everywhere MPC* (AE-MPC), which extended the concept of AE reliable communication previously studied by Dwork *et al.* [27]. In a nutshell, the paradigm of *almost-everywhere security* (AE-security) recognizes that when even all-to-all SMT is not possible (and only AE-SMT is available), then inevitably there will be uncorrupted parties for which we are unable to offer the security guarantees that honest parties enjoy in MPC (privacy, correctness, etc). The core mission 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 that 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 [15]. A major challenge, which we tackle, is to obtain a generic definition of AE-security that 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 realized by (straightforward UC adaptations of) classical AE-secure protocols. Since our treatment preserves

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

Construction	Degree	Corruption Tolerance	# Parties Doomed	Computation
Dwork *et al.* [27]	$O(1)$	$t = \Omega(n/\log n)$	$O(t)$	polynomial
Dwork *et al.* [27]	$O(n^\epsilon)$	$t = \Omega(n)$	$O(t)$	polynomial
Upfal [49]	$O(1)$	$t = \Omega(n)$	$O(t)$	exponential
Chandran *et al.* [20]	$O(\log^q n)$	$t = \Omega(n)$	$O(t/\log n)$	polynomial
Jayanti *et al.* [37]	$O(\log n)$	$t = \Omega(n)$	$O(t/\log n)$	polylogarithmic

Table 1: Comparison of constructions for AE-RMT and AE-agreement by degree of the graph, corruption tolerance t, number of parties doomed, and computation, all as a function of n (the total number of parties) and/or t. Both ϵ and q are constants.

composability of the (AE-)security statements, we obtain as a simple corollary the first simulation-based proof of AE-MPC.

Finally, 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 [5, 17, 43, 36, 13, 6]. Before providing more details on our results, we first provide some necessary literature background that should help the reader appreciate the relevance of our contributions and the challenges associated with them.

1.1 Related Work

The origins of the "almost-everywhere" (AE) notion can be traced back to the work of Dwork *et al.* [27], who considered the task of Byzantine agreement [45, 41] over sparse communication networks. In such networks, correctness cannot be guaranteed for all honest parties, since for example the adversary can isolate a node by corrupting all 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 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 $W = P - T - D$ (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 [24] 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.* [27] 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 (AE-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 $t = \Omega(n/\log n)$ corruptions while dooming $O(t)$ nodes (suboptimal corruption tolerance), and a graph of degree n^ϵ (for any $0 < \epsilon < 1$) admitting an AE-RMT scheme tolerating $t = \Omega(n)$ corruptions while dooming $O(t)$ nodes (suboptimal degree). Several follow-up works have obtained improved parameters for AE-RMT (and thus also for AE agreement). Upfal [49] gave a transmission scheme tolerating $t = \Omega(n)$ corruptions and dooming $O(t)$ nodes in a network of constant degree, which is the optimal set of parameters, but the protocol runs in exponential time. Chandran *et al.* [20] proposed a scheme tolerating $t = \Omega(n)$ corruptions and dooming $O(t/\log n)$ nodes in a network of poly-logarithmic degree. Most recently, Jayanti *et al.* [37] 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 result from [20]. We summarize the above combinations of parameters in Table 1.

Due to the results in [24, 25], 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 [31] introduced the notion of AE-MPC[2], which guarantees correctness and privacy only for the privileged parties. "Regular" information-theoretic MPC (i.e., MPC over a complete network) requires $t < n/3$ [10, 22]. 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.* [27] for simulating a protocol meant for a complete network, but 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 [25] 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 $\mathcal{A}$. While the simpler task of RMT (with no secrecy requirement) can be achieved for $t < n/2$ by simply duplicating the message over all wires, Dolev *et al.* [25] showed that SMT is also possible if and only if $t < n/2$.[3] We give a more detailed history of the SMT literature in Appendix A.1.

Returning to the work by Garay and Ostrovsky [31], the SMT-PD model overcomes the necessity of $2t + 1$ wires in SMT by additionally 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 resulting protocol dooms the same number of parties as the underlying (AE-secure) communication network.

In a follow-up, Chandran, Garay, and Ostrovsky [21] considered (information-theoretic) AE-MPC with *edge corruptions*. In this model, the adversary is additionally allowed to corrupt edges in the graph, independently of the endpoints of those edges. Chandran *et al.* presented a definitional framework and also demonstrated feasibility via a randomized construction. In our work, we consider only node corruptions.

To reiterate, all the above constructions are shown to be secure in a property-based manner. In Appendix A.2, 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 [15] 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: for example, in MPC the network is typically a complete graph of point-to-point channels (between each pair of parties), whereas the literature on (AE-) RMT assumes multiple paths (wires or indirect paths) between two parties. Thus, to derive a formulation general enough to capture the security of the above constructions, one first needs to develop a unified approach. Towards this goal, we adopt the graph model as a basis for all these protocols, and express the wires in the RMT/SMT literature as a simple graph which for each wire includes a path going through a unique "wire-party." We can then model corrupted wires as standard (party) corruptions in UC.

[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.

[3]It is assumed that communication over the wires is *synchronous*. One could also consider partially synchronous or asynchronous (with eventual message delivery) communication, but the required honest connectivity would suffer. Namely, the lower bound would worsen to $t < n/3$ [23].

The second, and more thorny challenge is regarding the (simulation of) doomed parties. Recall that these 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 for several reasons: For instance, 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, e.g., 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.

Accordingly, instead of modifying the foundations of UC, we define a class of functionalities that take requests from their adversary (simulator) to mark parties as doomed, and allow 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 the functionality to deliver output for doomed parties). In fact, this is done in a black-box manner, by *wrapping* an underlying (non-AE) functionality.

In more detail, 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. Thus, the party remains honest as far as the UC experiment is concerned, but the wrapper now has the ability to give full control over this party to the simulator it interacts with.

The final piece of the puzzle is capturing different ratios of corrupted vs doomed parties while making a composable statement. Here we use an idea inspired by [7]: We parameterize the wrapper by the set of all allowable corruption/doom patterns, and make sure that any request outside this set is ignored. For example, to prove security of AE-MPC with $t < \alpha n$ corruptions and $d < \beta n$ doomed parties, we can parameterize the wrapper with the pair (α, β) and ignore requests of simulators that do not respect the above requirements.

In fact, to allow for the tightest possible results that accurately translate non-threshold corruption/doom patterns (the types of results we get by using structural properties of the underlying graph), we draw inspiration from the mixed general adversary literature [35, 9]: 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. 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 our concrete instantiations consider structures that have a polynomial (in n) representation.

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

Almost-Everywhere RMT and SMT. We start in Section 3 by modeling 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 these primitives, 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 (corruptible) dummy party called a "wire-party," which simply relays messages between the sender and receiver.

In Section 3.1, we confirm that classical RMT/SMT protocols [25] 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 7) 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 AE-security of RMT/SMT protocols independently of the number of corrupted wire-parties, by using 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 [31]. We model the public channel using access to the same functionality that we use to capture RMT security. Looking ahead, we need SMT-PD when we want to elevate AE-RMT to AE-SMT over some classes of sparse graphs (like those in [49, 20, 37]) that do not rely on an honest majority of paths to obtain AE-RMT. However, for other sparse graphs (like those in [27]) the technique from [25] suffices.

Almost-Everywhere *Remote* RMT and SMT. In Section 4, we move on to the more complicated case where an incomplete graph connects several parties and yet all-to-all communication is desired. 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 (with wires replaced by not necessarily disjoint paths in 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 sparse graphs constructed in [27, 49, 20, 37], which we refer to as AE *remote* RMT. In Section 4.2, we extend our treatment to AE *remote* SMT for all of these graphs. First, we show that an SMT protocol from [25] can be adapted to UC-realize perfectly secure AE-SMT over a class of sparse graphs constructed in [27]. 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 [31] to realize AE-SMT over the graphs in [49, 20, 37], 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 using the same doom structure.

Almost-Everywhere Secure Computation. Lastly, in Section 5 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 AE counterparts, even against general (i.e., not necessarily threshold) adversaries (Theorem 26). 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, e.g., what happens when a protocol uses subprotocols to emulate secure channels and broadcast over a sparse network, but those subprotocols provide differing 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-wrapped hybrid, at the cost of tolerating a lower corruption threshold (Corollary 27).

We conclude in Section 5.2, by applying 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 [27, 49, 20, 37]. 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 28 and 29). 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 preliminaries. For the sake of readability, additional related work, and some of the functionalities, protocols, and proofs are presented in the appendices.

2 Preliminaries

For $n \in \mathbb{N}$, we denote by $[n]$ the set $\{1, \ldots, n\}$. Basic graph theory notions are presented in Appendix B.

2.1 UC Basics

Our results are in the UC framework [15] 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 (`corrupt` p_i) input from $\mathcal{Z}$. Denote by $\text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$ the probability distribution ensemble corresponding to the (binary) output of $\mathcal{Z}$ at the end of an execution of π with adversary $\mathcal{A}$. 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 $\mathcal{F}$, including 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 $\text{IDEAL}_{\mathcal{F},\mathcal{S},\mathcal{Z}} \equiv \text{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}$. *Statistical* UC-realization requires only that the two ensembles be indistinguishable, not identical. When π is a $(\mathcal{G}_1, \ldots, \mathcal{G}_n)$-hybrid protocol (i.e., making subroutine calls to $\text{IDEAL}_{\mathcal{G}_1}, \ldots, \text{IDEAL}_{\mathcal{G}_n}$), 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 $\mathcal{D}$, which 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. That is, 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. To avoid over-complicating the exposition, we use the standard round-based language of, e.g., [14, 44] to specify our protocols.

More formally, we assume a global clock functionality $\mathcal{G}_{\text{CLOCK}}$ [38, 6, 7]. Recall that the plain UC framework [15] does not provide any means of enforcing a round structure on protocols, but $\mathcal{G}_{\text{CLOCK}}$ can be used as a hybrid (in both the real and ideal worlds[4]) to guarantee that (1) parties and ideal functionalities are aware of the current round number, (2) the current round does not advance without the approval of all *honest* parties, and (3) the adversary cannot stall the advancement of rounds.

[4] As a global functionality, $\mathcal{G}_{\text{CLOCK}}$ can be called as a subroutine by other functionalities. We refer to [6] for the details of capturing universal composition with global subroutines within plain UC.

For each machine P_i in some participant set $\mathcal{P}$, the functionality $\mathcal{G}_{\text{CLOCK}}$ maintains an indicator bit d_i representing whether P_i is done with its instructions for the current round; this bit is initialized to 0 and set to 1 upon receiving a special ClockUpdate command from P_i. Another variable τ is used to keep track of the global round number (initially 0) and can be queried at any time by any machine using a ClockRead command. In order to increment τ, the functionality must have received the ClockUpdate command from every *honest* $P_i \in \mathcal{P}$ in the current round (so that $d_i = 1$ for all honest $P_i \in \mathcal{P}$), at which point the round advances and all d_i values are reset to 0. Note that, as long as the environment activates honest parties sufficiently many times, the adversary cannot prevent rounds from advancing.

This clock functionality can be used in a straightforward manner to specify synchronous protocols. Namely, a party first carries out its actions for the current round as usual. Then, in the next activation, it sends a ClockUpdate command to the shared instance of $\mathcal{G}_{\text{CLOCK}}$. Following activations are used to repeatedly query $\mathcal{G}_{\text{CLOCK}}$ via the ClockRead command, and as soon as the returned round number has been incremented, the party can be assured that all honest parties are collectively ready to move to the next round. We refer to [7] for a concrete specification of $\mathcal{G}_{\text{CLOCK}}$ but stress that all of our protocols are proven secure in a hybrid model with access to this global functionality.

2.2 Building Blocks

Here we present some building blocks that we will use in our constructions, as well as (somewhat informal) property-based definitions to contrast with our simulation-based treatment in later sections.

Property-based RMT/SMT. 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 (t-)*SMT* if it allows S to send a message $m \in \mathcal{M}$ to R such that the following 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.

The task in SMT-PD is the same, except that some reliability error ϵ is allowed (in exchange for a public channel provided by the model). We can define RMT by simply omitting the secrecy property, and AE-RMT and AE-SMT are defined by only requiring the properties to hold for privileged S and R (e.g., according to some protocol over a sparse graph).

A classical SMT construction. For simplicity, we will use the three-phase SMT protocol $\Pi_{\text{DDWY}}(\vec{\gamma}, m)$ (shown in Figure 20 in Appendix C), which is essentially the "FastSMT" protocol from [25] that tolerates the optimal threshold (a minority) of corrupted wires. The n wires are denoted by $\vec{\gamma} = (\gamma_1, \ldots, \gamma_n)$, and $\tau = \lceil \frac{n}{2} \rceil - 1$ specifies how many corrupted wires can be tolerated. Although the protocol assumes access to an authenticated channel between S and R, this can be implemented by simply sending the message m over all wires and having R take the majority. The high-level description of the protocol is as follows.

In the first phase, S chooses $n\tau + 1$ secret pads and secret-shares them via the n wires, associating each share with some checking pieces; the shares and the corresponding checking information constitute a "strong pad."[5] In the second phase, if some received strong pad is correct (i.e., the shares can be successfully interpolated, regardless of the checking pieces), then R sends its index back to S over the authenticated channel. In this case, in the third phase S uses the corresponding pad to encrypt the message m, and sends it to R over the authenticated channel; R also knows the pad—it is completely determined by at least $\tau + 1$ correct shares—and therefore decrypts m successfully.

Otherwise, if no correct strong pad is received in the first phase, it is possible for R (using the checking information to identify conflicting pairs of wires) to choose all but one of them to return to S over the

[5] In the journal version [26], Dolev *et al.* also present a "SlowSMT" protocol, which uses only simple pads together with a "wire-elimination" strategy, but it requires a linear number of phases in the worst case. Moreover, as an intermediate construction, the authors show how to reduce the phase complexity to 3 without the use of strong pads, by employing a cut-and-choose technique; this version of the protocol is not perfectly secure, as there is an error probability.

authenticated channel, such that the faulty shares of the retained strong pad will be detected by S. (This follows from a simple counting argument, using the fact that there are $n\tau + 1$ strong pads; since at most τ wires can be corrupted, only $n\tau$ distinct pairs of conflicting wires are possible.) In this case, in the third phase S sends the detected faults along with an encryption of m (using the retained pad) to R over the authenticated channel; now R can remove the faulty shares to compute the pad, and once again decrypts m.

Secrecy follows from the fact that the adversary sees at most τ shares of the secret pad, and does not receive enough checking pieces to learn anything about any of the remaining shares. Note that the $n\tau$ strong pads sent back to S over the authenticated channel do not reveal any information about the message either, as each strong pad is generated independently.

A classical SMT-PD construction. We will sometimes need an SMT-PD protocol, and for that we use protocol $\Pi_{\text{PUB-SMT}}(\vec{\gamma}, \text{Pub}, m)$ (shown in Figure 21 in Appendix C), which was given in [31] and tolerates $n-1$ corrupted wires, assuming access to public channel Pub and allowing a small probability of error (determined by a parameter l). The protocol starts with S sending n random bit strings of length $15l$ to R, one over each wire. Next, S reveals $3l$ random positions in each bit string over Pub (recall that the public channel is perfectly reliable). R compares these bits to what was received over the wires, and reports any detected corruptions back to S using Pub. For each (potentially honest) wire that remains, S chooses a random m_i, such that $m = \bigoplus_i m_i$; each m_i is encoded using an error-correcting code (with code-rate $\frac{1}{12}$ and correcting up to a $\frac{1}{4}$ fraction of errors) and then encrypted using the $12l$ unopened bits that were sent over the corresponding wire, before being sent to R over Pub. Finally, R uses the $12l$ bits that were received over each of those wires to decrypt each codeword (possibly with some errors), and with the help of the decoding algorithm, can correct any undetected faults to recover each m_i and ultimately compute m. Secrecy is perfect as long as there is at least one honest wire, because the bits sent on that wire will remain hidden from the adversary and therefore mask the message. However, reliability is contingent on the adversary not being able to flip more than $3l$ (or $\frac{1}{5}$) of the $15l$ bits sent on any wire without being detected when $3l$ random bits are revealed, so there is an error probability bounded by $n\left(\frac{4}{5}\right)^{3l}$. In order to achieve any small error ϵ, it suffices to set $l > \frac{\log(n/\epsilon)}{3\log(5/4)}$.

Property-based AE-MPC. Finally, we present the security definition for (property-based) AE-MPC that was given in [31]. Recall that W is the set of privileged nodes, as a function of the set of corruptions.

Definition 2 (AE-MPC, [31])**.** 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 properties hold at the end of the respective phases.

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

- **Binding:** For each 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:** Each $P_i \in W$ outputs $f(x_1^*, \ldots, x_n^*)$.
- **Privacy:** [Informal] For all $P_i \in W$, no information about x_i^* (beyond what can be inferred from the output) leaks to $\mathcal{A}$.

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 of [25]. 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 with different parameters.

The wire-party model. 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 $W_1, \ldots, W_n$ ($\vec{W}$ for short). 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 $\mathcal{F}_{\text{SC}}^{S,R,\vec{W}}$ shown in Figure 1, which provides secure communication between an honest S or R and an honest wire-party over a single round.[6] Looking ahead, this functionality is very similar to the functionality we use to capture secure channels between every pair of nodes connected by an edge in a sparse graph.

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

Figure 1: The secure channel functionality for the wire-party model.

The functionality $\mathcal{F}_{\text{SC}}^{S,R,\vec{W}}$ is single-use, and the actual sender and receiver (drawn from the set $\{S, R, W_1, \ldots, W_n\}$) for any given instance are respectively encoded as P_i and P_j in the SID. P_i can use the $\textsc{Send}$ command to send a message v to P_j; as usual, only the length $l(v)$ is leaked to the adversary by the functionality (via the $\textsc{SendLeak}$ command), unless P_i or P_j is corrupted. The functionality has a standard corruption interface: namely, the adversary can use the $\textsc{Corrupt}$ command to receive full leakage of the message and also enable adversarial "influence" (cf. [30]) on the message via a future $\textsc{InflSend}$ command. Finally, to retrieve the (possibly modified) message, P_j must use the $\textsc{Fetch}$ command in the next round.

Abstracting communication over wires. For convenience, we use $\mathcal{F}_{\text{SC}}^{S,R,\vec{W}}$ to realize the wire channel functionality $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ shown in Figure 2, which abstracts the process of sending a message to a wire-party, who then forwards it to S or R. We use P_s and P_r, drawn from the set $\{S, R\}$, to induce a direction on this bilateral wire channel. The functionality actually allows P_s to send P_r a potentially different message v_i through each wire-party W_i in parallel (i.e., in the same round), and it provides security for a given message as long as P_s, P_r, and the respective wire-party W_i are all honest. (As usual, if any of these parties are corrupted, then the adversary learns the respective message and can exert influence to modify it.)

[6] Our RMT protocols require only reliable edges. However, we eventually need secure channels to achieve SMT and MPC, so for simplicity, we present everything in the secure channels hybrid model.

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

The functionality is parameterized by the identities of the sender S, the receiver R, and the n wire-parties $\vec{W} = (W_1, \ldots, W_n)$. At the first activation, verify that $sid = (P_s, P_r, sid')$, where $\{P_s, P_r\} = \{S, R\}$. Initialize variables $m_1, \ldots, m_n$ to $\perp$.

- 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 set $m_i \leftarrow m_i'$.
- 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.
- Upon receiving $(\textsc{Corrupt}, sid, P)$ from the adversary for $P \in \{P_s, P_r, W_1, \ldots, W_n\}$, mark P as corrupted. If P is some wire-party W_i, then send $(\textsc{SendLeak}, sid, m_i)$ to the adversary; otherwise (i.e., $P \in \{P_s, P_r\}$), send $(\textsc{SendLeak}, sid, m_1, \ldots, m_n)$. If $P = P_r$, then additionally leak any previous fetch requests made by P_r.

Figure 2: The wire channel functionality.

Note that when a corrupted P_r attempts to fetch the message sent through W_i, the functionality $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ pings the adversary via the FetchLeak command, instead of simply returning the message; moreover, the interface exposes an Output command for the adversary to authorize delivery of the message to P_r. Strictly speaking, these commands are necessary for simulating the output of a corrupted P_r, since parties in the ideal world are incorruptible dummy parties who will immediately forward any messages received from the ideal functionality up to the environment. We deliberately omitted them from the specification of the secure channel functionality $\mathcal{F}_{\text{SC}}^{S,R,\vec{W}}$ for simplicity; as this is an atomic functionality—provided by the model and not meant to be realized—the same issue of simulatability does not come up there.

In addition to simplifying our RMT and SMT protocols, the wire channel functionality also has a very intuitive interpretation: it models the sending of messages in a single "phase" or direction, in the terminology of the PSMT literature (see Appendix A.1 for a discussion of *phases*). Note that since we are considering virtual wires that consist of just one intermediate node, $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ requires two rounds to generate output.

We can use the simple protocol $\Pi_{\text{WC}}(S, R, \vec{W})$ (shown in Figure 22 in Appendix C) to realize $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$. We prove the following proposition in Appendix D.1.

Proposition 3. *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

Modeling RMT. We model the task of RMT in UC with the authenticated communication functionality $\mathcal{F}_{\text{AUTH}}^{\mathcal{P},\mathsf{rnd}}$ shown in Figure 3, which is essentially Canetti's $\mathcal{F}_{\text{AUTH}}$ [16] with explicit 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.

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

The authenticated communication 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.

Figure 3: The authenticated communication functionality.

Achieving RMT in the wire-party model. To realize $\mathcal{F}_{\text{AUTH}}^{\mathcal{P},\mathsf{rnd}}$ in the wire-party model ($\mathcal{P} = \{S, R\}$) assuming only a minority of the wire-parties get corrupted, we can simply duplicate the message through all wire-parties and have the receiver (who may actually be S) take majority; see Figure 4 for a formal specification of protocol $\Pi_{\text{AUTH}}(S, R, \vec{W})$.

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

1. Upon receiving input (SEND, sid, v) from $\mathcal{Z}$ in round ρ, where $sid = (P_s, P_r, sid')$ and $\{P_s, P_r\} = \{S, R\}$, party P_s sends $(\text{SEND}, sid_{\text{AUTH}}, W_i, v)$ for each wire-party W_i to a single instance of $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ with SID $sid_{\text{AUTH}} = (sid, \text{AUTH})$.
2. Upon receiving input (FETCH, sid) from $\mathcal{Z}$ in round $\rho+2$, party P_r sends $(\text{FETCH}, sid_{\text{AUTH}}, W_i)$ for each W_i to $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$. Upon receiving back $(\text{SENT}, sid_{\text{AUTH}}, W_i, m_i)$ 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 (SENT, sid, m') to $\mathcal{Z}$. (If not enough copies were received, e.g. because P_s was corrupted, then P_r outputs $\perp$.)

Figure 4: The RMT protocol in the wire-party model.

Theorem 4. *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.*

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 $\mathcal{Z}$ 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 $(\text{SENDLEAK}, sid, m)$ 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 FETCH input from $\mathcal{Z}$ in round $\rho + 2$. If P_r is corrupted by $\mathcal{A}$, then $\mathcal{S}$ can wait to receive $(\text{FETCHLEAK}, sid)$ 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 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 INFLSEND messages sent by $\mathcal{A}$ to the simulated instance of $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$), before sending an OUTPUT message; if the corruption was in fact direct, then $\mathcal{S}$ simulates P_r reporting to $\mathcal{A}$ that a FETCH input from $\mathcal{Z}$ was received, and then sends appropriate INFLSEND and 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 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. □

Modeling SMT. Next, we capture SMT in UC with the secure message transmission functionality $\mathcal{F}_{\text{SMT}}^{\mathcal{P},\mathsf{rnd}}$ shown in Figure 5, which is essentially Canetti's $\mathcal{F}_{\text{SMT}}$ [16] with synchrony.

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

The secure message transmission 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.

Figure 5: The secure message transmission functionality.

Achieving SMT in the wire-party model. To realize $\mathcal{F}_{\text{SMT}}^{\mathcal{P},\mathsf{rnd}}$ in the wire-party model assuming only a minority of the wire-parties get corrupted, we can use the protocol $\Pi_{\text{SMT}}(S, R, \vec{W})$ shown in Figure 6, which is essentially the FastSMT protocol from [25] adapted for our UC treatment (see Section 2.2).

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

Upon receiving input (SEND, sid, v) from $\mathcal{Z}$ in round ρ, where $sid = (P_s, P_r, sid')$ and $\{P_s, P_r\} = \{S, R\}$, party P_s executes protocol $\Pi_{\text{DDWY}}(\vec{\gamma}, v)$ 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 phases, P_r and P_s substitute the authenticated channel with separate instances of $\mathcal{F}_{\text{AUTH}}^{\{S,R\},2}$ with $sid_2 = (P_r, P_s, sid', 2)$ 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 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 $\mathcal{F}_{\text{AUTH}}^{\{S,R\},2}$ are two-round functionalities.
2. Upon receiving input (FETCH, sid) from $\mathcal{Z}$ in round $\rho' = \rho + 6$, P_r outputs (SENT, sid, m') to $\mathcal{Z}$ if it receives m' as the output of this protocol.

Figure 6: The SMT protocol in the wire-party model.

Theorem 5. *Protocol $\Pi_{\text{SMT}}(S, R, \vec{W})$ UC-realizes $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ for $\mathsf{rnd} = 6$ in the $(\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}, \mathcal{F}_{\text{AUTH}}^{\{S,R\},2})$-hybrid model, against an adversary corrupting up to 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 (unbounded) 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 $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$, $\mathcal{F}_{\text{AUTH}}^{\{S,R\},2}$, 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 $(\text{SENDLEAK}, sid, \hat{m})$ 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 two phases of the protocol honestly, by simulating sending random strong pads (shares $h_i(\cdot)$ and checking pieces $C_i = (c_{1i}, \ldots, c_{ni})$) 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 $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$) and by simulating sending the response from P_r to P_s over the authenticated channel (i.e., by appropriately playing the role of $\mathcal{F}_{\text{AUTH}}^{\{S,R\},2}$ for $\mathcal{A}$). For the third phase of the protocol, $\mathcal{S}$ simulates honestly, except for choosing z when P_s and P_r are both honest in which case $\mathcal{S}$ simulates sending a random value z from P_s to P_r over $\mathcal{F}_{\text{AUTH}}^{\{S,R\},2}$ instead of $z = m + p$. When P_s or P_r is corrupted by $\mathcal{A}$, $\mathcal{S}$ learns m via leakage from $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ and thus can send $z = m + p$ just like in the real protocol. Note that if the simulated P_r aborts by outputting $\perp$, then $\mathcal{S}$ can influence $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$, since this can only happen if $\mathcal{A}$ corrupts P_s or P_r.

If P_s or P_r is corrupted before $\mathcal{S}$ decides about z in its internal execution, $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ leaks m to $\mathcal{S}$ allowing for perfect simulation. It is also easy to perfectly simulate when both P_s and P_r are honest throughout the whole execution. 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 $i \in [n]$. Since all $h_i(\cdot)$'s are random polynomials of degree τ, $\Pr[h_i(0) = a \mid \{c_{ij}\}_{j \in I}] = \Pr[h_i(0) = a]$ and since the $h_i(\cdot)$'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 $\Pr[f(0) = a \mid \{h_i(0)\}_{i\in I}] = \Pr[f(0) = a]$ (where $f(0)$ 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, $z = m+p$ looks uniformly random to $\mathcal{A}$ and $\mathcal{Z}$ if no more than 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, $\mathcal{F}_{\text{AUTH}}^{\{S,R\},2}$ provides genuine authentication of messages intended to be sent on the authenticated channel in the protocol, and hence in the real world P_r outputs the sender's input.

An important case arises when both P_s and P_r are honest at the beginning of the third phase (at the time $\mathcal{S}$ decides the value of z), but then at least one of them gets corrupted later on. In this scenario, $\mathcal{A}$ receives sufficient 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 randomly chosen by $\mathcal{S}$, the message learned by $\mathcal{A}$ deviates from what is originally sent by P_s, causing $\mathcal{Z}$ to distinguish between the real and ideal worlds. In such a situation, $\mathcal{S}$ learns the actual value of m through leakage from $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ and can cheat by calculating a fake pad p' that satisfies $z = m + p'$. $\mathcal{S}$ can then simulate leakage from $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ to reveal p'. Therefore, even in this last scenario, the simulation is perfect.

Note that during the simulation, $\mathcal{S}$ has the capability to send an InflSend message to $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ when P_s or P_r is corrupted in order to modify the message as necessary, taking into account the behavior of $\mathcal{A}$ during its internal execution. This allows $\mathcal{S}$ to actively control and manipulate the message so that P_r always gets the updated message. Furthermore, in cases where P_r is corrupted and $\mathcal{S}$ receives a FetchLeak message from $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$, $\mathcal{S}$ can proceed with its internal execution by sending a Fetch input to (the simulated) P_r. By doing so, $\mathcal{S}$ can observe the internal execution and based on the behavior of $\mathcal{A}$, exert influence and potentially send an Output message to $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$. □

3.2 Corrupted Majorities of Wire-Parties

'Doom' structures and an AE-security wrapper. 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* (shown in Figure 7) that allows parties to be marked as doomed according to the current set of corruptions.

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

The wrapper is parameterized by a doom structure $\mathscr{D} = \{(T_1, D_1), \ldots, (T_m, D_m)\}$, where each $(T_i, D_i) \in 2^{\mathcal{P}} \times 2^{\mathcal{P}}$. 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 (Corrupt, sid, P_i) from the adversary for $P_i \in \mathcal{P}$: If $(T \cup \{P_i\}, D) \in \mathscr{D}$, then set $T \leftarrow T \cup \{P_i\}$, relay the message to $\mathcal{F}$, and send back $\mathcal{F}$'s response.
- Upon receiving (Doom, sid, P_i) from the adversary for $P_i \in \mathcal{P}$: If $(T, D \cup \{P_i\}) \in \mathscr{D}$, then set $D \leftarrow D \cup \{P_i\}$, send (Corrupt, sid, P_i) to $\mathcal{F}$, and send back $\mathcal{F}$'s response.
- 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 $\mathcal{F}$.

Figure 7: The AE wrapper functionality.

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.[7] We call such structures *doom structures*.

We are now equipped to realize *wrapped* $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}}$ and $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$, with respect to the following doom structure $\mathscr{D}_{\text{PSMT}}$ (PSMT stands for "perfectly secure message transmission," as it was called by Dolev *et al.* [25]):

Definition 6 (doom structure over $\mathcal{P} = \{S, R, W_1, \ldots, W_n\}$)**.** Let $(T_i, D_i) \in \mathscr{D}_{\text{PSMT}}$ if and only if either $|T_i - \{S, R\}| < \frac{n}{2}$ and $D_i = \emptyset$, or $|T_i - \{S, R\}| \geq \frac{n}{2}$ and $D_i \subseteq \{S, R\}$.

RMT against corrupted majorities of wire-parties. The following theorem is a straightforward extension of Theorem 4, but we sketch a proof in Appendix D.2. Note that the protocol is unchanged (see Figure 4).

Theorem 7. *Protocol* $\Pi_{\text{AUTH}}(S, R, \vec{W})$ *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.*

SMT against corrupted majorities of wire-parties. To realize wrapped $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$, define protocol $\Pi'_{\text{SMT}}(S, R, \vec{W})$ by replacing invocations of $\mathcal{F}_{\text{AUTH}}^{\{S,R\},2}$ in protocol $\Pi_{\text{SMT}}(S, R, \vec{W})$ (Figure 6) with invocations of $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{PSMT}}}(\mathcal{F}_{\text{AUTH}}^{\{S,R\},2})$. We sketch a proof here to illustrate the usage of our wrapper.

Theorem 8. *Protocol* $\Pi'_{\text{SMT}}(S, R, \vec{W})$ *UC-realizes* $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{PSMT}}}(\mathcal{F}_{\text{SMT}}^{\{S,R\},6})$ *in the* $(\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}, \mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{PSMT}}}(\mathcal{F}_{\text{AUTH}}^{\{S,R\},2}))$*-hybrid model, even against corrupted majorities of wire-parties.*

Proof. [Sketch] We construct a simulator $\mathcal{S}$ that is very similar to the simulator in the proof of Theorem 5. However, $\mathcal{S}$ now interacts with a *wrapped* $\mathcal{F}_{\text{SMT}}$ functionality, and corruption messages for wire-parties are indeed sent because they can now be processed by the wrapper. Another difference concerns the case in which P_s and P_r are not corrupted by $\mathcal{A}$. 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 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 $\mathcal{F}_{\text{SMT}}$ functionality. Now, $\mathcal{S}$ can use $z = m + p$ in the third phase, and influences the wrapper every time the value that the real-world P_r would have output changes (these influence messages will be accepted by the wrapper). Another 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 with m, like the simulator in the proof of Theorem 5 does. Once again, the simulation is perfect. □

Next, we turn to SMT-PD, 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 before, with the public channel modeled by assuming access to $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}'}$ for some rnd'. Assuming at least one wire-party remains honest, we can realize (unwrapped) $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ using the protocol $\Pi_{\text{SMT-PD}}(S, R, \vec{W})$ shown in Figure 8, which is essentially a UC adaptation of the classical SMT-PD protocol $\Pi_{\text{PUB-SMT}}(\vec{\gamma}, \text{Pub}, m)$ that was presented in Section 2.2. However, the trade-off is that we obtain only statistical security.

Theorem 9. *Protocol* $\Pi_{\text{SMT-PD}}(S, R, \vec{W})$ *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 up to all but one of the wire-parties.*

[7] This is a technicality, which simplifies some of our definitions and results. For example, the definition of AE-monotonicity (introduced in Definition 24) would not be quite as short and intuitive otherwise.

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

Upon receiving input (SEND, sid, v) from $\mathcal{Z}$ in round ρ, where $sid = (P_s, P_r, sid')$ and $\{P_s, P_r\} = \{S, R\}$, party P_s executes protocol $\Pi_{\text{PUB-SMT}}(\vec{\gamma}, \text{Pub}, v)$ 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 $W_1, \ldots, W_n$ in $\vec{W}$:

1. In the first step, 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 $\vec{\gamma}$. In the second, third, and fourth steps, 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 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 for the invocation of $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ and rnd' rounds per invocation of $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}'}$.
2. Upon receiving input (FETCH, sid) from $\mathcal{Z}$ in round $\rho' = \rho + 2 + 3 \cdot \mathsf{rnd}'$, P_r outputs (SENT, sid, m') to $\mathcal{Z}$ if it receives m' as the output of this protocol.

Figure 8: The SMT-PD protocol in the wire-party model.

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 $\mathcal{Z}$ can distinguish whether it is interacting with $\Pi_{\text{SMT-PD}}(S, R, \vec{W})$ 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 $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$, $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}'}$, 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 $(\text{SENDLEAK}, sid, \hat{m})$ 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 steps of the protocol honestly, by simulating sending random bit strings 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 $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$), and by simulating sending a message from P_s to P_r or vice versa over the public channel (by appropriately playing the role of $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}'}$ for $\mathcal{A}$). In the fourth step, $\mathcal{S}$ 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 via leakage from $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$.

Next, we describe how $\mathcal{S}$ simulates the remaining part of the execution until P_r (or $\mathcal{A}$ when P_r is corrupted) generates the output. If P_s or P_r was corrupted prior to $\mathcal{S}$ making a decision about the m_i's, the simulation becomes straightforward because $\mathcal{S}$ has access to the private input. Consequently, everything will naturally add up in the view generated by $\mathcal{S}$. If both P_s and P_r are honest throughout the whole execution, then $\mathcal{S}$ does not learn the message m at all and needs to simulate using some message m' that with high probability is not equal to m. However, in that case, the adversarial view would be independent of the message, which does not let $\mathcal{Z}$ distinguish. The main reason for this is that we assumed adversaries who will not corrupt all the wire-parties. Therefore, there will be at least one m_i that $\mathcal{A}$ would not learn (the one whose corresponding bit string is not known), which allows the adversarial view to remain consistent with any message m provided by $\mathcal{Z}$.

A more intricate case arises 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 step. In this scenario, $\mathcal{A}$ can recover some m' from its view by receiving leakages from all the instances of $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$. If m' does not match m, then $\mathcal{Z}$ can distinguish between the real and ideal worlds. The challenge lies in the fact that $\mathcal{S}$ had to choose a message before learning m, which, with high probability, makes it different from m. However, $\mathcal{S}$ can handle this case by simulating what was sent in the first step. In particular, at least one bit string (corresponding to an uncorrupted wire-party) sent in the first step is not visible to $\mathcal{A}$, so $\mathcal{S}$ can redefine it to be consistent with m (which $\mathcal{S}$ learns from leakage

from $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$).

There are some important details regarding the simulation that we should mention here. $\mathcal{S}$ has the capability to send an InflSend message to $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$ when P_s or P_r is corrupted in order to modify the message as necessary, taking into account the behavior of $\mathcal{A}$ during its internal execution. This allows $\mathcal{S}$ to actively control and manipulate the message so that P_r always gets the updated message. Furthermore, in cases where P_r is corrupted and $\mathcal{S}$ receives a FetchLeak message from $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$, $\mathcal{S}$ can proceed with its internal execution by sending a Fetch input to (the simulated) P_r. By doing so, $\mathcal{S}$ can observe the internal execution and based on the behavior of $\mathcal{A}$, exert influence and potentially send an Output message to $\mathcal{F}_{\text{SMT}}^{\{S,R\},\mathsf{rnd}}$.

The views generated in the real and ideal worlds are identical whenever the protocol succeeds. However, in the real world, with a small probability $\mathcal{A}$ can manipulate the random bit strings sent through corrupted wire-parties such that they cannot be corrected by the error-correcting code, and yet will not be detected when P_s opens some random bits of the bit strings to identify and discard corrupted wires. Such a scenario cannot happen in the ideal world, which may allow $\mathcal{Z}$ to distinguish. Therefore, the simulation is not perfect and it only yields statistical UC-realization. □

4 Almost-Everywhere *Remote* RMT and SMT

In this section, we consider *remote*—i.e., over a (possibly sparse) graph G_n—RMT and SMT. As in Section 3, we model the network topology using a parameterized secure channel functionality $\mathcal{F}_{\text{SC}}^{G_n}$, shown in Figure 9, which provides secure channels only between parties that are connected in G_n.

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

The functionality is parameterized by a graph $G_n = (V, E)$ of party identities and communication edges. 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.

Figure 9: The secure channel functionality for (incomplete) graph G_n.

Abstracting communication over paths. For convenience, instead of always 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 $\mathcal{F}_{\text{R-SC}}^{G_n}$ (shown in Figure 10), which is 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. We emphasize that even privileged nodes may not be able to use $\mathcal{F}_{\text{R-SC}}^{G_n}$ to communicate securely, if the chosen path is corrupted. Indeed, by corrupting a party along the path, the adversary can influence the message in transit (or even drop it entirely, by replacing it with $\perp$).

Using protocol $\Pi_{\text{R-SC}}(G_n)$ (shown in Figure 23 in Appendix C), we can realize $\mathcal{F}_{\text{R-SC}}^{G_n}$ by simply forwarding the message along the path, which leads to the following statement. We omit the proof, as it is very similar to the proof of Proposition 3.

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.

Figure 10: The remote secure channel functionality for single-path communication.

Proposition 10. *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

We first show how classical AE-RMT protocols from the AE agreement literature can be adapted to UC-realize our wrapped $\mathcal{F}_{\text{AUTH}}^{\mathcal{P},\mathsf{rnd}}$ functionality, using doom structures that are derived from the protocol and the underlying sparse graph.

4.1.1 Graphs of Constant Degree

We first describe a scheme due to Dwork *et al.* [27], which guarantees AE *reliable* communication in classes of constant-degree graphs (such as their "butterfly" network) that admit a certain three-phase transmission scheme. At a high level, the scheme associates with every node v a *fan-in* set $\Gamma_{\text{in}}(v)$ and a *fan-out* set $\Gamma_{\text{out}}(v)$ of a fixed (but not necessarily constant) size s. In addition, (not necessarily vertex-disjoint) paths from a node to its sets are specified, as well as (vertex-disjoint) paths from one node's fan-out set to another node's fan-in set. Node u transmits a message m to node v by running the following protocol $\Pi_{\text{DPPU}}(u, v, m)$: first u sends m to all members of $\Gamma_{\text{out}}(u)$; each member then forwards the message to its connected (via a path) node in $\Gamma_{\text{in}}(v)$; and finally each member of $\Gamma_{\text{in}}(v)$ forwards the message to v, who simply takes majority. Now, let t denote the maximum number of corruptions, and for every node v suppose that $|\Gamma_{\text{in}}(v)| = |\Gamma_{\text{out}}(v)| = s > 4t$. Given a set of adversarial nodes T, it is shown in [27] that if less than $\frac{1}{8}$ 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 $\frac{1}{4}$ of the paths from $\Gamma_{\text{out}}(u)$ to $\Gamma_{\text{in}}(v)$ are corrupted (which is guaranteed as long as $s > 4t$), and less than $\frac{1}{8}$ of the paths from $\Gamma_{\text{out}}(v)$ to node v are corrupted, then u and v can communicate reliably.

Furthermore, it is shown in [27] how to construct the above three-phase transmission scheme for several classes of graphs. Using the terminology from Section 1, the set of privileged nodes $W(T)$ in this case consists of the nodes u such that less than $\frac{1}{8}$ of both the paths from u to $\Gamma_{\text{out}}(u)$ and the paths from $\Gamma_{\text{in}}(u)$ to u are corrupted, and $D(T)$ is the corresponding set of doomed nodes. Let x be the maximum size of $D(T)$

over all T of size at most t. Recall that we would like to obtain as low a value of x as possible ($O(t)$ is optimal in this case), while tolerating a large value of t (ideally $\Omega(n)$). Although Dwork *et al.* constructed several classes of graphs admitting AE-RMT with various combinations of parameters, we consider only the graphs that use the three-phase transmission scheme described above (the other graphs use an appended or different transmission scheme). The parameters achieved are as follows:

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

A UC treatment of the protocol. Let $G_n^{\text{DPPU}} = (V_{\text{DPPU}}, E_{\text{DPPU}})$ be a graph that admits such a three-phase transmission scheme. To realize wrapped $\mathcal{F}_{\text{AUTH}}^{V_{\text{DPPU}},\mathsf{rnd}}$, we use protocol $\Pi_{\text{R-AUTH}}^{\text{DPPU}}$ (presented in Figure 11), which is a straightforward UC adaptation of Π_{DPPU} in the $\mathcal{F}_{\text{R-SC}}^{G_n^{\text{DPPU}}}$ -hybrid model, and the following doom structure $\mathscr{D}_{\text{DPPU}}$.

Definition 11 (doom structure over V_{DPPU})**.** For any corruption set T_i, let $D_{\text{DPPU}}(T_i)$ be the set of all participants P such that 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. Then, let $(T_i, D_i) \in \mathscr{D}_{\text{DPPU}}$ if and only if $|T_i| < s/4$ and $D_i \subseteq D_{\text{DPPU}}(T_i)$.

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

1. Upon receiving input (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 (SEND, sid_i, v) to an instance of $\mathcal{F}_{\text{R-SC}}^{G_n^{\text{DPPU}}}$ 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 Π_{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 (FETCH, sid_i) to $\mathcal{F}_{\text{R-SC}}^{G_n^{\text{DPPU}}}$. Upon receiving back $(\text{SENT}, sid_i, m_i)$, R stores m_i as the value received on path γ_i.
3. Upon receiving input (FETCH, sid) from $\mathcal{Z}$ in round $\rho + \mathsf{rnd}$, where rnd is the maximum length of *any* three-step path (i.e., not just one from S to R) specified by Π_{DPPU}, 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 (SENT, sid, m') to $\mathcal{Z}$. (If not enough copies were received, then R outputs $\perp$.)

Figure 11: The AE-RMT protocol over G_n^{DPPU}.

Theorem 12. *Protocol $\Pi_{\text{R-AUTH}}^{\text{DPPU}}$ UC-realizes $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{DPPU}}}(\mathcal{F}_{\text{AUTH}}^{V_{\text{DPPU}},\mathsf{rnd}})$ for some $\mathsf{rnd} \in O(\log n)$ in the $\mathcal{F}_{\text{R-SC}}^{G_n^{\text{DPPU}}}$ -hybrid model, against an adversary corrupting less than $s/4$ 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 environment $\mathcal{Z}$ can distinguish whether it is interacting with $\Pi_{\text{R-AUTH}}^{\text{DPPU}}$ and $\mathcal{A}$, or with $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{DPPU}}}(\mathcal{F}_{\text{AUTH}}^{V_{\text{DPPU}},\mathsf{rnd}})$ and $\mathcal{S}$. The simulator internally runs a copy of $\mathcal{A}$, and plays the roles of $\mathcal{F}_{\text{R-SC}}^{G_n^{\text{DPPU}}}$ and the parties in a simulated execution of the protocol, which starts when $\mathcal{S}$ receives $(\text{SENDLEAK}, sid, m)$ 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 at least $\frac{1}{8}$ of its 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 their paths to Γ_{out} and from Γ_{in} corrupted. However, it follows from the results in [27] 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 the Π_{DPPU} transmission scheme have length $O(\log n)$. Since $\Pi_{\text{R-AUTH}}^{\text{DPPU}}$ transmits the message from S to R by sending it through the specified paths, its execution requires only $\mathsf{rnd} \in O(\log n)$ rounds. □

We can also formulate the above result in threshold (as opposed to doom-structure) terms (cf. [27]):

Corollary 13. *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 $32t \log 16t$ honest nodes.*

Building on [27], Upfal [49] proposed an alternative transmission scheme for constant-degree graphs, which actually works over *any* graph; however, optimality is achieved only on constant-degree expander graphs with specific parameters. The main limitation of the scheme is that it is computationally inefficient. Node u transmits a message m to node v by sending it through all the simple paths connecting them. As the message travels along a path to v, 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 the ID of at least one corrupted node, and v can enumerate over all the possible corruption sets to recover m. Denote this protocol by $\Pi_{\text{UPFAL}}(u, v, m)$.

A UC treatment of the protocol. Let $G^{\text{UPFAL}}_n = (V_{\text{UPFAL}}, E_{\text{UPFAL}})$ be an n-node Ramanujan graph (see Appendix B). To realize wrapped $\mathcal{F}^{V_{\text{UPFAL}},\mathsf{rnd}}_{\text{AUTH}}$, we use protocol $\Pi^{\text{UPFAL}}_{\text{R-AUTH}}$ (presented in Figure 12), which is a straightforward UC adaptation of Π_{UPFAL} in the $\mathcal{F}^{G^{\text{UPFAL}}_n}_{\text{SC}}$-hybrid model, and the following doom structure $\mathscr{D}_{\text{UPFAL}}$.

Definition 14 (doom structure over V_{UPFAL})**.** First, define $D_{\text{UPFAL}}(T_i)$ 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 are in S. Then, let $(T_i, D_i) \in \mathscr{D}_{\text{UPFAL}}$ if and only if $|T_i| \leq 1/72n$ and $D_i \subseteq D_{\text{UPFAL}}(T_i)$.

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

1. Upon receiving input (SEND, sid, v) from $\mathcal{Z}$ in round ρ, where $sid = (S, R, sid')$ for $S, R \in V_{\text{UPFAL}}$, 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 of $\mathcal{F}^{G^{\text{UPFAL}}_n}_{\text{SC}}$ (note that we do not use $\mathcal{F}^{G^{\text{UPFAL}}_n}_{\text{R-SC}}$ 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 $\mathcal{F}^{G^{\text{UPFAL}}_n}_{\text{SC}}$, 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 length of any path used in the protocol, R outputs (SENT, sid, m') to $\mathcal{Z}$ if it receives m' as the output of this protocol.

Figure 12: The AE-RMT protocol over G^{UPFAL}_n.

Theorem 15. *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 $\mathsf{rnd} \in O(\log n)$ in the $\mathcal{F}^{G^{\text{UPFAL}}_n}_{\text{SC}}$-hybrid model, against an adversary corrupting up to $1/72n$ 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 environment $\mathcal{Z}$ can distinguish whether it is interacting with $\Pi^{\text{UPFAL}}_{\text{R-AUTH}}$ and $\mathcal{A}$, or with $\mathcal{W}^{\mathscr{D}_{\text{UPFAL}}}_{\text{AE}}(\mathcal{F}^{V_{\text{UPFAL}},\mathsf{rnd}}_{\text{AUTH}})$ and $\mathcal{S}$. The simulator internally runs a copy of $\mathcal{A}$, and plays the roles of $\mathcal{F}^{G^{\text{UPFAL}}_n}_{\text{SC}}$ and the parties in a simulated execution of the protocol, which starts when $\mathcal{S}$ receives $(\text{SENDLEAK}, sid, m)$ 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 $D_{\text{UPFAL}}(T)$, 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 [49] 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 $\mathcal{Z}$.

It is well-known that the diameter of an expander graph is $O(\log n)$. Since we are working over expander graphs and the Π_{UPFAL} transmission scheme uses only simple paths between S and R, all the messages are received by R in $O(\log n)$ rounds. Therefore, $\Pi^{\text{UPFAL}}_{\text{R-AUTH}}$ requires $\mathsf{rnd} \in O(\log n)$ rounds to terminate. □

The simulator we construct needs to run the potentially exponential-time process that the receiver does at the end of the protocol, to determine the output when the sender or receiver is doomed, although that seems reasonable since the protocol itself is inefficient. As before, in threshold terms (cf. [49]), we obtain:

Corollary 16. *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.*

4.1.2 Graphs of Poly-Logarithmic Degree

Chandran *et al.* [20] proposed a randomly constructed graph $G^{\text{CGO}}_n = (V_{\text{CGO}}, E_{\text{CGO}})$ of poly-logarithmic degree, admitting an AE-RMT scheme that tolerates $\Omega(n)$ corruptions while dooming only $O(t/\log n)$ nodes. The graph is constructed by first forming $n \log^k n$ overlapping committees (for some constant k) of size $O(\log \log n)$ via random walks on an expander graph. All nodes within a committee are connected by a clique, and the committees (viewed as "super-nodes") are connected by "super-edges" according to the constant degree butterfly network from [27]. A super-edge between two committees is formed by connecting the i'th node in one committee to the i'th node in the other committee (according to, say, a lexicographic ordering of nodes). Finally, each node is assigned to $O(\log^k n)$ "helper" committees according to a bipartite expander; a node is connected by an edge to each node in each of its helper committees.

The proposed protocol $\Pi_{\text{CGO}}(u, v, m)$ for transmitting a message m from node u to node v is as follows: first u sends m to all of its helper committees; those committees then transmit the message to v's helper committees using Π_{DPPU} (at the committee level); finally, v takes majority over the values received from its helpers. To send a message from one committee to another over a super-edge, each node in the source committee sends the message to its corresponding node in the other committee, and then all nodes in the destination committee run a *differential agreement*[8] protocol [28] over the values they have received. This serves as a sort of error correction as the message travels from committee to committee, ensuring that two "honest" committees (defined below) that are connected by a super-edge can communicate reliably.

A UC treatment of the protocol. To realize wrapped $\mathcal{F}^{V_{\text{CGO}},\mathsf{rnd}}_{\text{AUTH}}$, we use protocol $\Pi^{\text{CGO}}_{\text{R-AUTH}}$ (presented in Figure 13), which is a straightforward UC adaptation of Π_{CGO} in the $\mathcal{F}^{G^{\text{CGO}}_n}_{\text{SC}}$-hybrid model, and the following doom structure $\mathscr{D}_{\text{CGO}}$.

Definition 17 (doom structure over V_{CGO})**.** First, for any corruption set T_i, let $D_{\text{CGO}}(T_i)$ be the set of all participants P such that at least $\frac{1}{6}$ of P's helper committees are unprivileged. A committee is considered corrupted if at least $\frac{1}{4}$ of its members are corrupted, and committees are categorized as unprivileged according to the $D_{\text{DPPU}}(\cdot)$ function (Definition 11). Then, let $(T_i, D_i) \in \mathscr{D}_{\text{CGO}}$ if and only if corrupting the nodes in T_i causes at most $\frac{n \log^k n}{4 \log(n \log^k n)}$ committees to become corrupted, and $D_i \subseteq D_{\text{CGO}}(T_i)$.

Chandran *et al.* [20] proved that there exists a constant α_{CGO} such that for any adversary corrupting at most $\alpha_{\text{CGO}} n$ nodes, at most $\frac{n \log^k n}{4 \log(n \log^k n)}$ committees become corrupted. Thus, for that constant, we have the following theorem.

Theorem 18. *Protocol $\Pi^{\text{CGO}}_{\text{R-AUTH}}$ UC-realizes $\mathcal{W}^{\mathscr{D}_{\text{CGO}}}_{\text{AE}}(\mathcal{F}^{V_{\text{CGO}},\mathsf{rnd}}_{\text{AUTH}})$ for some $\mathsf{rnd} \in O(\log n \cdot \log \log n)$ in the $\mathcal{F}^{G^{\text{CGO}}_n}_{\text{SC}}$-hybrid model, against an adversary corrupting at most $\alpha_{\text{CGO}} n$ 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 environment $\mathcal{Z}$ can distinguish whether it is interacting with $\Pi^{\text{CGO}}_{\text{R-AUTH}}$ and $\mathcal{A}$, or with $\mathcal{W}^{\mathscr{D}_{\text{CGO}}}_{\text{AE}}(\mathcal{F}^{V_{\text{CGO}},\mathsf{rnd}}_{\text{AUTH}})$ and $\mathcal{S}$. The simulator internally runs a copy of $\mathcal{A}$, and plays the roles of $\mathcal{F}^{G^{\text{CGO}}_n}_{\text{SC}}$ and the parties in a simulated

[8] Differential agreement guarantees that if many (not necessarily all) of the honest parties begin with the same value, then all of the honest parties will output that value.

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

1. Upon receiving input $(\textsc{Send}, sid, v)$ from $\mathcal{Z}$ in round ρ, where $sid = (S, R, sid')$ for $S, R \in V_{\text{CGO}}$, 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 of $\mathcal{F}_{\text{SC}}^{G_n^{\text{CGO}}}$. To receive output from the instances of $\mathcal{F}_{\text{SC}}^{G_n^{\text{CGO}}}$, all nodes involved have to send $\textsc{Fetch}$ messages in the correct rounds.

2. Upon receiving input $(\textsc{Fetch}, sid)$ from $\mathcal{Z}$ in round $\rho + \mathsf{rnd}$, where rnd is (two plus) the maximum number of rounds required by the Π_{DPPU} transmission scheme over the committees multiplied by one plus the maximum number of rounds required by differential agreement inside the committees, R outputs $(\textsc{Sent}, sid, m')$ to $\mathcal{Z}$ if it receives m' as the output of this protocol.

Figure 13: The AE-RMT protocol over G_n^{CGO}.

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 $\frac{1}{6}$ of the helper committees of S or R are unprivileged, 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 $\mathscr{D}_{\text{CGO}}$. The only case in which $\mathcal{S}$ cannot influence is when both S and R are privileged, which means less than $\frac{1}{6}$ of their helpers are unprivileged. As it is shown in [20], $\mathcal{A}$ also cannot influence the communication in this case, so $\mathcal{S}$ can simply let the dummy R fetch from the wrapper when instructed by $\mathcal{Z}$.

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 [28], deterministic differential agreement requires at most a linear number of rounds. Since in the Π_{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 also know that in the Π_{CGO} transmission scheme, there are $n\log^k n$ committees communicating using the Π_{DPPU} transmission scheme over super-edges. We discussed earlier that the Π_{DPPU} transmission scheme requires a logarithmic number of rounds. Therefore, the total number of rounds required by $\Pi^{\text{CGO}}_{\text{R-AUTH}}$ is $O(\log(n\log^k n)\cdot\log\log n) = O(\log n\cdot\log\log n)$. □

Chandran *et al.* also showed that there exists a constant β_{CGO} such that for any T_i with $|T_i| \le t = \alpha_{\text{CGO}} n$, it holds that $|D_{\text{CGO}}(T_i) - T_i| \le \beta_{\text{CGO}}\frac{t}{\log n}$. Now we can formulate the above result in terms of thresholds over the number of corrupted and doomed nodes, as stated in [20] in the property-based setting:

Corollary 19. *There exist constants $\alpha_{\text{CGO}}, \beta_{\text{CGO}}$ such that over G_n^{CGO} 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 at most $\beta_{\text{CGO}}\frac{t}{\log n}$ honest nodes.*

4.1.3 Graphs of Logarithmic Degree

Jayanti *et al.* [37] recently proposed a family of randomly constructed graphs of logarithmic degree, admitting AE-RMT with $\Omega(n)$ corruptions tolerated and $O(t/\log n)$ doomed nodes (this is the best-known *efficient* AE-RMT protocol tolerating a constant fraction of corruptions). Their graphs consist of $z = k\log n$ layers, where k is a constant, which are constructed using the same method but over a random permutation of the vertex set. To form a layer, the nodes are arbitrarily partitioned into n/s committees of size $s = c\log\log n$, where c is a constant. Each committee is instantiated with G_{UPFAL}, and committees are connected by "super-edges" according to the butterfly network G_{DPPU}. A super-edge is simply a perfect matching between the node sets of two committees. Let $\mathcal{G}_{\text{JRV}}$ be the family of graphs constructed as above.

The proposed protocol $\Pi_{\text{JRV}}(u, v, m)$ for reliable message transmission over graphs in $\mathcal{G}_{\text{JRV}}$ is as follows: node u transmits a message m to node v by sending it through each layer separately, and in the end v takes majority over the values received from all the layers. In each layer, if u and v are in the same committee, they simply invoke $\Pi_{\text{UPFAL}}(u, v, m)$ within the committee. Otherwise, if u and v are located in different committees, u first sends m to all the nodes in its committee using Π_{UPFAL}; then u's committee sends the

message to v's committee over super-edges using Π_{DPPU}; every node in v's committee (except v) now sends the message to v using Π_{UPFAL}, and v takes majority over all the incoming messages. To send a message over a super-edge between committees, each node in the source committee sends the message to its matched node in the destination committee, who then sends it to all other nodes in the committee using Π_{UPFAL}, and finally, each node locally computes the majority of the received messages. As in [20], this procedure provides a type of error correction as messages travel from committee to committee across super-edges.

A UC treatment of the protocol. Let $G_n^{\text{JRV}} = (V_{\text{JRV}}, E_{\text{JRV}}) \in \mathcal{G}_{\text{JRV}}$ (see below). To realize wrapped $\mathcal{F}_{\text{AUTH}}^{V_{\text{JRV}},\mathsf{rnd}}$, we use protocol $\Pi_{\text{R-AUTH}}^{\text{JRV}}$ (presented in Figure 14), which is a straightforward UC adaptation of Π_{JRV} in the $\mathcal{F}_{\text{SC}}^{G_n^{\text{JRV}}}$ -hybrid model, and the following doom structure $\mathscr{D}_{\text{JRV}}$.

Definition 20 (doom structure over V_{JRV})**.** First, in each layer of G_n^{JRV}, if a committee contains more than $\frac{1}{72}s$ corruptions, call it *bad*. If the total number of bad committees in a layer exceeds $\frac{n/s}{4\log(n/s)}$, call the layer *bad*. Next, for any corruption set T_i, let $D_{\text{JRV}}(T_i)$ be the set of all participants P that are doomed in more than $\frac{1}{10}z$ *good* layers. A node is considered doomed in a layer if it is located in a doomed committee (with respect to the $D_{\text{DPPU}}(\cdot)$ function from Definition 11) or is doomed itself within its committee (with respect to the $D_{\text{UPFAL}}(\cdot)$ function from Definition 14). Then, let $(T_i, D_i) \in \mathscr{D}_{\text{JRV}}$ if and only if corrupting the nodes in T_i causes at most $\frac{1}{5}$ of the layers to become bad, and $D_i \subseteq D_{\text{JRV}}(T_i)$.

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

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 of $\mathcal{F}_{\text{SC}}^{G_n^{\text{JRV}}}$. 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 two times the maximum number of rounds required by the Π_{UPFAL} transmission scheme inside the committees, plus the maximum number of rounds required by the Π_{DPPU} transmission scheme over the committees multiplied by one plus the maximum number of rounds required by the Π_{UPFAL} transmission scheme inside the committees, R outputs (SENT, sid, m') to $\mathcal{Z}$ if it receives m' as the output of this protocol.

Figure 14: The AE-RMT protocol over G_n^{JRV}.

Jayanti *et al.* [37] proved that there exist a graph $G_n^{\text{JRV}} \in \mathcal{G}_{\text{JRV}}$ and constant α_{JRV} such that for any adversary corrupting at most $\alpha_{\text{JRV}}n$ nodes, at most $\frac{1}{5}$ of the layers become bad. For such a graph and constant, we have the following theorem.

Theorem 21. *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 $\mathsf{rnd} \in O(\log n \cdot \log\log\log n)$ in the $\mathcal{F}_{\text{SC}}^{G_n^{\text{JRV}}}$ -hybrid model, against an adversary corrupting at most $\alpha_{\text{JRV}}n$ 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 environment $\mathcal{Z}$ 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 $(\text{SENDLEAK}, sid, m)$ 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 either S or R is doomed in more than $\frac{1}{10}z$ 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 $\mathscr{D}_{\text{JRV}}$. The only case in which $\mathcal{S}$ cannot influence is when both S and R are privileged, which means they are doomed in at most $\frac{1}{10}z$ good layers. As it is shown in [37], $\mathcal{A}$ also cannot influence the communication in this case, so $\mathcal{S}$ can simply let the dummy R fetch from the wrapper when instructed by $\mathcal{Z}$.

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 the Π_{UPFAL} transmission scheme inside the destination committee. As discussed earlier, the Π_{UPFAL} transmission scheme requires a logarithmic number of rounds. Since in the Π_{JRV} transmission scheme each committee has size $s = O(\log \log n)$, each super-edge transmission takes $O(\log \log \log n)$ rounds. We also know that in the Π_{JRV} transmission scheme, there are n/s committees communicating using the Π_{DPPU} transmission scheme over super-edges. We discussed earlier that the Π_{DPPU} transmission scheme requires a logarithmic number of rounds. Therefore, the total number of rounds required by $\Pi_{\text{R-AUTH}}^{\text{JRV}}$ is $O\left(\log(n/s) \cdot \log \log \log n\right) = O(\log n \cdot \log \log \log n)$. □

Jayanti *et al.* also showed that there exists a constant β_{JRV} such that for any T_i with $|T_i| \leq t = \alpha_{\text{JRV}} n$, it holds that $|D_{\text{JRV}}(T_i) - T_i| \leq \beta_{\text{JRV}} \frac{t}{\log n}$. Again, in threshold terms, we obtain (cf. [37]):

Corollary 22. *There exist a graph G_n^{JRV} 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 at most $\beta_{\text{JRV}} \frac{t}{\log n}$ honest nodes.*

4.2 AE Remote SMT

To achieve AE *secure* communication over the constant-degree graphs studied by Dwork *et al.* [27], we can apply the approach that was used in Section 3.2 to obtain AE-SMT in the wire-party model. Specifically, we can adapt protocol $\Pi'_{\text{SMT}}(S, R, \vec{W})$ to work over the s paths used by the three-phase transmission scheme over G_n^{DPPU}. The resulting protocol $\Pi_{\text{R-SMT}}^{\text{DPPU}}$ (shown in Figure 15) realizes wrapped $\mathcal{F}_{\text{SMT}}^{V_{\text{DPPU}}, \mathsf{rnd}'}$ for some rnd' with the same doom structure $\mathscr{D}_{\text{DPPU}}$ from Section 4.1.

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

1. Upon receiving input (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 $\Pi_{\text{DDWY}}(\vec{\gamma}, v)$ with the receiver 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 Π_{DPPU} transmission scheme), respectively. More precisely, in the first phase S and R use s different instances of $\mathcal{F}_{\text{R-SC}}^{G_n^{\text{DPPU}}}$ with SIDs $sid_i = (\lambda_i, sid')$ 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 SIDs $sid_{\text{AUTH}_1} = (R, S, 1, sid')$ and $sid_{\text{AUTH}_2} = (S, R, 2, sid')$, respectively, where rnd is the maximum length of any three-step path specified by Π_{DPPU}. To receive output from the aiding functionalities, S and R have to send 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 path λ_i, the second-phase message is sent in round $\rho + \ell$ (ℓ is the maximum value of the l_i's), and the second-phase and third-phase messages are respectively fetched in rounds $\rho + \ell + \mathsf{rnd}$ and $\rho + \ell + 2 \cdot \mathsf{rnd}$.
2. Upon receiving input (FETCH, sid) from $\mathcal{Z}$ in round $\rho + \ell + 2 \cdot \mathsf{rnd}$, R outputs (SENT, sid, m') to $\mathcal{Z}$ if it receives m' as the output of this protocol.

Figure 15: The AE-SMT protocol over G_n^{DPPU}.

In more detail, the sender runs the classical SMT protocol $\Pi_{\text{DDWY}}(\vec{\gamma}, m)$ in [25] (see Figure 20 in Appendix C) with the receiver, using separate instances of $\mathcal{F}_{\text{R-SC}}^{G_n^{\text{DPPU}}}$ in phase 1 as a substitute for sending messages through the wires in $\vec{\gamma}$, and separate instances of $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{DPPU}}}(\mathcal{F}_{\text{AUTH}}^{V_{\text{DPPU}}, \mathsf{rnd}})$ (for some rnd depending on the length of the used paths) in phases 2 and 3 as a substitute for the authenticated channel. Let ℓ denote the maximum length of any of the paths used in the protocol. We stress that the s paths from S to R are not necessarily the same (except reversed) as the s paths from R to S.

Theorem 23. *Protocol $\Pi_{\text{R-SMT}}^{\text{DPPU}}$ UC-realizes $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{DPPU}}}(\mathcal{F}_{\text{SMT}}^{V_{\text{DPPU}}, \ell + 2 \cdot \mathsf{rnd}})$ in the $(\mathcal{F}_{\text{R-SC}}^{G_n^{\text{DPPU}}}, \mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{DPPU}}}(\mathcal{F}_{\text{AUTH}}^{V_{\text{DPPU}}, \mathsf{rnd}}))$-hybrid model for some $\mathsf{rnd} \in O(\log n)$, against an adversary corrupting less than $s/4$ nodes.*

Proof. [Sketch] 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 $\mathcal{W}^{\mathscr{D}_{\text{DPPU}}}_{\text{AE}}(\mathcal{F}^{V_{\text{DPPU}},\ell+2\cdot\mathsf{rnd}}_{\text{SMT}})$ and $\mathcal{S}$. The simulator $\mathcal{S}$ is similar to the simulator in the proof of Theorem 8. 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 8 needs to doom at least one of P_s and P_r (denoted S and R here) when a majority of the wire-parties between them are corrupted (which is allowed by $\mathscr{D}_{\text{PSMT}}$) but $\mathcal{S}$ can do that if at least $\frac{1}{8}$ of the paths to Γ_{out} or from Γ_{in} are corrupted for at least one of S and R. As it is discussed in [27], whenever a majority of wires between S and R are corrupted, at least $\frac{1}{8}$ of the paths to Γ_{out} or from Γ_{in} are corrupted for at least one of S and R. Therefore, the difference in the doom structures does not affect the correctness of the simulation.

The round complexity is easily established by inspection (recall that AE-RMT over G^{DPPU}_n takes $O(\log n)$ rounds). □

Note that the bounds on the number of corruptions and the number of doomed parties are encoded in the doom structure and were already mentioned in Section 4.1.1.

Generalizing via SMT-PD. The technique described above cannot generally be extended to other AE-RMT schemes because it relies on a majority of honest paths between privileged nodes to establish a secure link between them. However, many transmission schemes, including Upfal's [49], do not guarantee this property for privileged nodes. Actually, according to [27], this condition is not necessary at all to achieve AE-RMT. To achieve AE-SMT using other transmission schemes, one approach is to work in the SMT-PD model, which only requires a single honest path between the sender and receiver to establish a secure channel, assuming access to a public channel. This approach can be employed to add privacy to any AE-RMT scheme, as these schemes provide an authenticated (public) channel between privileged nodes and ensure at least one honest path between them. We employ a UC adaptation of the classical SMT-PD protocol $\Pi_{\text{PUB-SMT}}(\vec{\gamma}, \text{Pub}, m)$ in [31] (see Figure 21 in Appendix C), which we modify to operate over the graphs utilized by the transmission schemes. Hence, it is crucial to emphasize, as discussed in Section 3.3, that this approach only offers statistical security. Before presenting further details, we first introduce some notation.

Definition 24 (properties of doom structures)**.** Let $\mathscr{D}$ be a doom structure with participant set $\mathcal{P}$, and 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 $\mathscr{D}$). We say that $\mathscr{D}$ is:

1. *t-complete* if $\mathsf{dom}(\mathscr{D}) = \{T \subseteq \mathcal{P} : |T| \leq t\}$ (in other words, if all possible sets of corruptions of size at most t are allowed by $\mathscr{D}$);
2. *D-monotone* if whenever $(T_j, D_j) \in \mathscr{D}$ and $D_i \subseteq D_j$, it holds that $(T_j, D_i) \in \mathscr{D}$; and
3. *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 $(T_j, D_i) \in \mathscr{D}$.

We remark that t-completeness simply allows for UC-security statements in threshold terms. Moreover, D-monotonicity, akin to the standard notion of monotonicity in the general adversary literature,[9] simply ensures compatibility with our AE wrapper, which does not accept batched doom requests. On the other hand, AE-monotonicity is different in that it 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 [31] included a similar assumption: that the function mapping the set of corrupted parties to the corresponding set of unprivileged parties is monotonically increasing. AE-monotonicity seems to be important for simulatability; for example, after a new corruption, all of the previously doomed parties should still be allowed to be doomed, since otherwise the wrapper would ignore the new corruption request and the simulation would fail. A more subtle issue, which does not hinge on the specifics of our wrapper, is discussed in Section 5.1 in the context of proving our composition theorem. Fortunately, all of our doom structures are t-complete, D-monotone, and AE-monotone.

[9] One could similarly define a notion of *T-monotonicity*, but such a property would not be satisfied by realistic doom structures.

Now, let $G_n = (V, E)$ be a graph with polynomially many paths of length at most ℓ specified between every pair of nodes. Suppose we already know how to realize wrapped $\mathcal{F}_{\text{AUTH}}^{V,\mathsf{rnd}}$ for some rnd, with respect to a doom structure $\mathscr{D}_{\text{SMT-PD}}$ (with $\mathcal{P} = V$) that is t-complete, D-monotone, and AE-monotone and moreover satisfies the following condition: For all $T \subseteq V$ with $|T| \leq t$, at least one of the specified paths between any pair of nodes in $V - T - \cup_{(T,D_i)\in\mathscr{D}_{\text{SMT-PD}}} D_i$ is completely contained in $V - T$. Then, we can realize wrapped $\mathcal{F}_{\text{SMT}}^{V,\mathsf{rnd}'}$ (for some rnd' depending on rnd and ℓ) using protocol $\Pi_{\text{R-SMT-PD}}(G_n)$ (shown in Figure 16), which is essentially protocol $\Pi_{\text{SMT-PD}}(S, R, \vec{W})$ from Section 3.3 adapted to work over the specified paths in G_n, and the same doom structure $\mathscr{D}_{\text{SMT-PD}}$. In more detail, we substitute functionalities $\mathcal{F}_{\text{WC}}^{S,R,\vec{W}}$ and $\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}'}$ with $\mathcal{F}_{\text{R-SC}}^{G_n}$ and $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{SMT-PD}}}(\mathcal{F}_{\text{AUTH}}^{V,\mathsf{rnd}})$, respectively.

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

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

Figure 16: The SMT-PD protocol over G_n.

For such a doom structure and parameters, we obtain the following statement.

Theorem 25. *Protocol $\Pi_{\text{R-SMT-PD}}(G_n)$ statistically UC-realizes $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{SMT-PD}}}(\mathcal{F}_{\text{SMT}}^{V,\mathsf{rnd}'})$ for $\mathsf{rnd}' = \ell + 3 \cdot \mathsf{rnd}$ 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 a t-adversary.*

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 $\mathcal{Z}$ can distinguish whether it is interacting with $\Pi_{\text{R-SMT-PD}}(G_n)$ and $\mathcal{A}$, or with $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{SMT-PD}}}(\mathcal{F}_{\text{SMT}}^{V,\ell+3\cdot\mathsf{rnd}})$ and $\mathcal{S}$. The simulator $\mathcal{S}$ is very similar to the simulator that was constructed in the proof of Theorem 9. However, $\mathcal{S}$ now interacts with a *wrapped* $\mathcal{F}_{\text{SMT}}$ 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* $\mathcal{F}_{\text{AUTH}}$ functionality rather than an ideal authenticated channel.

If $\mathcal{S}$ can doom either S or R according to the doom structure $\mathscr{D}_{\text{SMT-PD}}$ before the fourth step, then $\mathcal{S}$ does not need to cheat because it learns m through leakage from $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{SMT-PD}}}(\mathcal{F}_{\text{SMT}}^{V,\mathsf{rnd}'})$. However, $\mathcal{S}$ still needs to exert constant influence on $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{SMT-PD}}}(\mathcal{F}_{\text{SMT}}^{V,\mathsf{rnd}'})$.

Furthermore, if $\mathcal{S}$ cannot doom S and R throughout the entire execution, it implies that $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{SMT-PD}}}(\mathcal{F}_{\text{AUTH}}^{V,\mathsf{rnd}})$ behaves like an ideal authenticated channel (since it is wrapped using the same doom structure). In addition, according to the assumed properties of $\mathscr{D}_{\text{SMT-PD}}$, at least one path between S and R must remain uncorrupted. Therefore $\mathcal{S}$, who does not know m in this case, can choose m_i's randomly for the fourth step and still make the simulation work. This simulation strategy is effective because $\mathcal{A}$, who cannot tamper with $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{SMT-PD}}}(\mathcal{F}_{\text{AUTH}}^{V,\mathsf{rnd}})$ or corrupt all the paths, will not learn all the pads and hence gains no information about the message.

In a more elaborate scenario, neither S nor R is doomed at the beginning of the fourth step. However, at a later point, one of them becomes doomed due to the evolution of the corruption set. In such a case, $\mathcal{S}$ cannot learn the value of m before deciding on the m_i's. Consequently, instead of selecting m_i's that satisfy $m = m_1 \oplus \cdots \oplus m_s$, $\mathcal{S}$ chooses them randomly. There are two possibilities to consider, in both of which the aforementioned strategy works effectively. First, suppose that the new corruption, which leads to the doom of either S or R, does not corrupt any existing honest path between them. As a result, there remains at least one bit string that is unknown to $\mathcal{A}$, making it impossible for $\mathcal{A}$ to gain any information about m. It is important to note that since S or R only becomes doomed after the third step, $\mathcal{A}$ cannot manipulate $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{SMT-PD}}}(\mathcal{F}_{\text{AUTH}}^{V,\mathsf{rnd}})$ to eliminate all honest paths and force S to transmit messages solely through corrupted paths. Therefore, choosing random m_i's is a viable approach in this situation. Now, let us consider the second possibility where the doom of S or R is associated with the corruption of a new path between them. In such circumstances, it is possible for $\mathcal{A}$ to learn the actual message m through leakages from $\mathcal{F}_{\text{R-SC}}^{G_n}$. However, since S or R is now doomed, $\mathcal{S}$ also obtains the correct value of m from $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{SMT-PD}}}(\mathcal{F}_{\text{SMT}}^{V,\mathsf{rnd}'})$. $\mathcal{S}$ can then cheat by recalculating the bit string transmitted via the last (newly) corrupted path in a way that allows the desired message to be recovered at the end. Afterward, $\mathcal{S}$ simulates the leakage of this adjusted bit string from the instance of $\mathcal{F}_{\text{R-SC}}^{G_n}$ corresponding to the last corrupted path. This guarantees that $\mathcal{A}$ acquires the correct message m. Hence, the simulation strategy is effective in both cases.

It should be noted that once S or R becomes doomed, $\mathcal{S}$ gains complete control over $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{SMT-PD}}}(\mathcal{F}_{\text{SMT}}^{V,\mathsf{rnd}'})$ and can exert influence as required to emulate $\mathcal{A}$'s behavior regarding the correctness of the message.

The round complexity is easily established by inspection. □

It is worth mentioning that a similar statement can be proven for general adversaries by considering a doom structure $\mathscr{D}_{\text{SMT-PD}}$ that is D-monotone, AE-monotone, and satisfies the following condition: For all $T \in \mathsf{dom}(\mathscr{D}_{\text{SMT-PD}})$, at least one of the specified paths between any pair of nodes in $V - T - \cup_{(T,D_i)\in\mathscr{D}_{\text{SMT-PD}}} D_i$ is completely contained in $V - T$.

According to [27], all the realizable doom structures for AE remote RMT satisfy the former condition above. In short, the 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, protocol $\Pi_{\text{R-SMT-PD}}(G_n)$ can be used with any of the classes of sparse graphs discussed in Section 4.1 to achieve AE remote SMT with statistical security.

5 Almost-Everywhere Secure Computation

In this section, we consider general UC-secure computation in the AE 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 the AE-monotonicity property that was presented in Definition 24, 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 moreover 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 AE-monotonicity 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 classes of 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

The AE compiler is shown in Figure 17. 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 for the $(\mathcal{W}_{\text{AE}}^{\mathscr{D}_1}(\mathcal{F}_1), \ldots, \mathcal{W}_{\text{AE}}^{\mathscr{D}_m}(\mathcal{F}_m))$-hybrid model. Of course, the compiled protocol will not in general realize wrapped $\mathcal{F}$ with the same doom

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 $\mathcal{W}_{\mathrm{AE}}^{\mathscr{D}_i}(\mathcal{F}_i)$ (which has the same input/output format to the parties).

Figure 17: The AE compiler.

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 $T' \cup D$ is tolerated as a corrupted set by $\mathscr{D}$, for all $D \subseteq D(T')$. As mentioned above, we additionally require that $\mathscr{D}, \mathscr{D}_1, \ldots, \mathscr{D}_m$ are all AE-monotone, so that the constructed $\mathscr{D}'$ is AE-monotone. This is used in the reduction, as 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. Indeed, AE-monotonicity seems generally important for simulatability in the AE setting. We summarize the notation used in this section in Table 2.

$\mathcal{P}$: participant set (i.e., the set of all parties)
$\mathscr{D}, \mathscr{D}_1, \ldots, \mathscr{D}_m, \mathscr{D}' \subseteq 2^{\mathcal{P}} \times 2^{\mathcal{P}}$: doom structures (see Section 3.2)
$\mathsf{dom}\colon 2^{2^{\mathcal{P}} \times 2^{\mathcal{P}}} \to 2^{2^{\mathcal{P}}}$: function mapping a doom structure $\mathscr{D}$ to a set $\mathsf{dom}(\mathscr{D})$ of sets of corruptible parties (see Definition 24)
$\mathcal{T}, \mathcal{T}' \subseteq 2^{\mathcal{P}}$: sets of sets of corruptible parties (i.e., adversary structures in the usual sense)
$T', T_i, T_j \subseteq \mathcal{P}$: sets of corrupted parties
$D\colon 2^{\mathcal{P}} \to 2^{\mathcal{P}}$: function mapping a set T' of corrupted parties to a set $D(T')$ of doomed parties
$D, D_i, D_j, A \subseteq 2^{\mathcal{P}}$: sets of doomed parties
$\mathcal{D}$: the dummy adversary (see Section 2.1)
t, t': corruption thresholds

Table 2: Summary of notation used in this section.

Theorem 26. *Let $\mathscr{D}, \mathscr{D}_1, \ldots, \mathscr{D}_m$ be AE-monotone doom structures over the same participant set $\mathcal{P}$. Let $\mathcal{T} = \mathsf{dom}(\mathscr{D})$ and $\mathcal{T}' = (\bigcap_{i=1}^m \mathsf{dom}(\mathscr{D}_i)) \cap \mathcal{T}$. For any $T' \in \mathcal{T}'$, define*

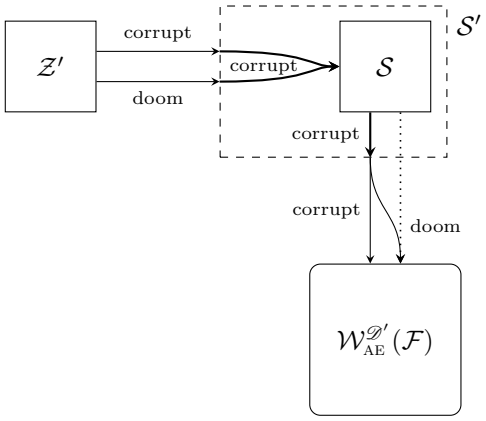
$$D(T') = \bigcup_{i=1}^{m} \left(\bigcup_{(T', D_j) \in \mathscr{D}_i} D_j \right).$$

Suppose that for all $T' \in \mathcal{T}'$ and $D \subseteq D(T')$, it holds that $T' \cup D \in \mathcal{T}$. If protocol Π UC-realizes $\mathcal{W}_{\mathrm{AE}}^{\mathscr{D}}(\mathcal{F})$ 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 $\mathcal{W}_{\mathrm{AE}}^{\mathscr{D}'}(\mathcal{F})$ in the $(\mathcal{W}_{\mathrm{AE}}^{\mathscr{D}_1}(\mathcal{F}_1),\ldots,\mathcal{W}_{\mathrm{AE}}^{\mathscr{D}_m}(\mathcal{F}_m))$-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.

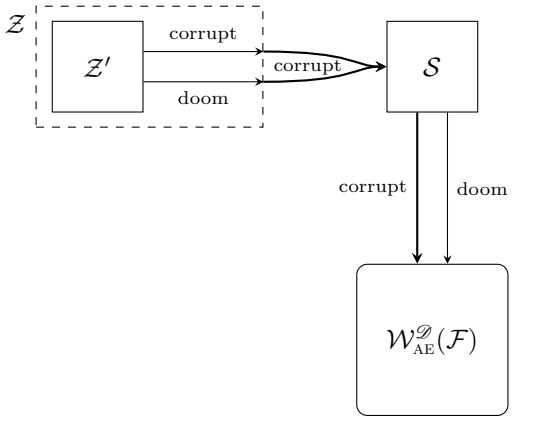
Proof. We first prove that $\mathscr{D}'$ is AE-monotone. Suppose that $(T_i, D_i) \in \mathscr{D}'$ and $T_i \subseteq T_j$ for $T_j \in \mathcal{T}'$. This means that $D_i = D \cup A$ for some D, A such that $D \subseteq D(T_i)$ and $(T_i \cup D(T_i), A) \in \mathscr{D}$. We want to show that $(T_j, D_i) \in \mathscr{D}'$, 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 assumption 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 assumptions 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 environment $\mathcal{Z}$ can distinguish whether it is interacting with Π and the dummy adversary $\mathcal{D}$, or with $\mathcal{W}_{\text{AE}}^{\mathscr{D}}(\mathcal{F})$ and $\mathcal{S}$. We use $\mathcal{S}$ to construct a simulator $\mathcal{S}'$ (see Figure 18(a)) such that no environment $\mathcal{Z}'$ can distinguish whether it is interacting with $\mathcal{C}^{\mathscr{D}_1,\ldots,\mathscr{D}_m}(\Pi)$ and $\mathcal{D}$, or with $\mathcal{W}_{\text{AE}}^{\mathscr{D}'}(\mathcal{F})$ and $\mathcal{S}'$.



(a) $\text{IDEAL}_{\mathcal{W}_{\text{AE}}^{\mathscr{D}'}(\mathcal{F}),\mathcal{S}',\mathcal{Z}'}$



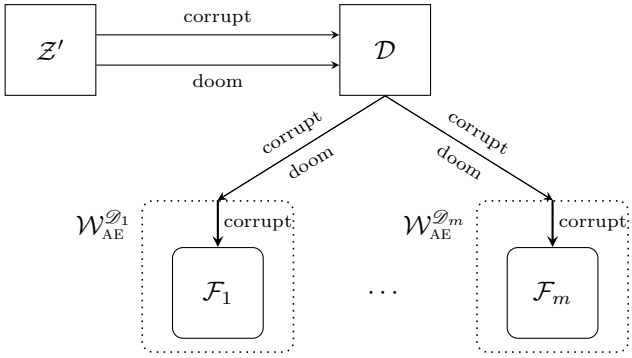
(b) $\text{IDEAL}_{\mathcal{W}_{\text{AE}}^{\mathscr{D}}(\mathcal{F}),\mathcal{S},\mathcal{Z}}$

Figure 18: The ideal-world experiments considered in the proof of Theorem 26. The corresponding distributions are shown to be identical. To minimize clutter, many interactions are omitted and the dummy parties are not shown.

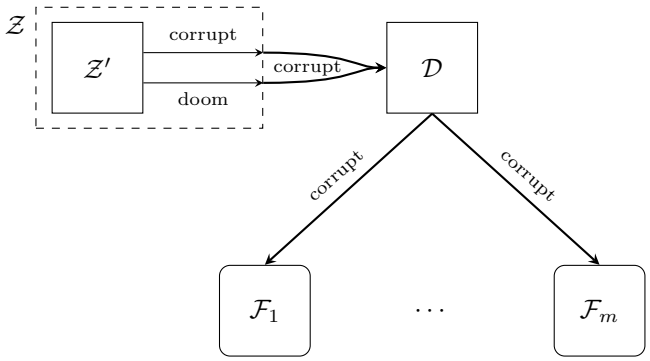
$\mathcal{S}'$ internally runs $\mathcal{S}$ and plays the role of the environment and $\mathcal{W}_{\text{AE}}^{\mathscr{D}}(\mathcal{F})$ for it. Inputs from $\mathcal{Z}'$ are forwarded to $\mathcal{S}$, with some additional processing. When $\mathcal{Z}'$ sends a corruption request directed to a 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, 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.



(a) $\text{EXEC}_{\mathcal{C}^{\mathscr{D}_1,\ldots,\mathscr{D}_m}(\Pi),\mathcal{D},\mathcal{Z}'}$



(b) $\text{EXEC}_{\Pi,\mathcal{D},\mathcal{Z}}$

Figure 19: The real-world experiments considered in the proof of Theorem 26. The corresponding distributions are shown to be identical. To minimize clutter, many interactions are omitted and only one instance of each hybrid functionality is shown.

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 $\mathcal{Z}$ (see Figure 18(b) or Figure 19(b)) 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 $\text{EXEC}_{\mathcal{C}^{\mathscr{D}_1,\ldots,\mathscr{D}_m}(\Pi),\mathcal{D},\mathcal{Z}'} \equiv \text{EXEC}_{\Pi,\mathcal{D},\mathcal{Z}}$. In-

deed, 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 $\mathcal{S}'$ (see Figure 18). 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 $\mathcal{D}$ (see Figure 19). That concludes the proof. □

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 properties that were introduced in Definition 24.

Corollary 27. *Let $\mathscr{D}$ be a t'-complete, D-monotone, and AE-monotone doom structure, and let*

$$t = \max_{|T'|=t'} \left| \left(\bigcup_{(T',D_i)\in\mathscr{D}} D_i \right) \cup T' \right|.$$

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 allows the simulator to handle a threshold adversary that can corrupt *any* t' parties, and D-monotonicity is needed for the doom structure $\mathscr{D}$ that is used to wrap $\mathcal{G}$ to be preserved when wrapping $\mathcal{F}$. We remark that t has a very natural interpretation: 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 over a complete 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 network with secure channels only between connected parties, using our AE compiler. To capture the MPC task for n-ary function f, we use the functionality $\mathcal{F}_{\text{MPC}}^{f,\mathcal{P},\mathsf{rnd}}$ (shown in Figure 24 in Appendix C), which is essentially Canetti's $\mathcal{F}_{\text{SFE}}$ [16] with synchrony.

While (regular) perfectly secure MPC protocols tolerating $t < \frac{n}{3}$ corruptions are known [10],[10] they assume access to a broadcast channel, noting that broadcast can be realized when $t < \frac{n}{3}$. 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. Clearly this holds even in the $\mathcal{F}_{\text{SMT}}^{\mathcal{P},\ell}$-hybrid model, for arbitrary ℓ. Now, by invoking Corollary 27 (which of course also offers statistical security) and then applying the (regular) UC composition theorem in tandem with our results in Theorems 23 and 25 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), for any n-ary function f:

Corollary 28. *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 $t \in \Omega(\frac{n}{\log n})$.*

Corollary 29. *Let $\mathtt{x} \in \{\text{UPFAL}, \text{CGO}, \text{JRV}\}$. There exists a protocol statistically UC-realizing $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\mathtt{x}}}(\mathcal{F}_{\text{MPC}}^{f,V_{\mathtt{x}},\mathsf{rnd}})$ in the $\mathcal{F}_{\text{SC}}^{G_n^{\mathtt{x}}}$-hybrid model against a t-adversary, for some rnd and $t \in \Omega(n)$.*

Acknowledgements. The authors are grateful to Ran Canetti for useful discussions during preliminary stages of this work.

[10] A full simulation-based proof of the protocol has been given in [4]. Although this proof implies adaptive UC security only with inefficient simulation, using the methodology in [3], one can show adaptive security with efficient simulation.

References

[1] S. Agarwal, R. Cramer, and R. de Haan. Asymptotically optimal two-round perfectly secure message transmission. In C. Dwork, editor, *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, volume 4117 of *Lecture Notes in Computer Science*, pages 394–408. Springer, 2006.

[2] B. Alon, E. Omri, and A. Paskin-Cherniavsky. MPC with friends and foes. In D. Micciancio and T. Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 677–706. Springer, 2020.

[3] G. Asharov, R. Cohen, and O. Shochat. Static vs. adaptive security in perfect MPC: A separation and the adaptive security of BGW. In D. Dachman-Soled, editor, *3rd Conference on Information-Theoretic Cryptography, ITC 2022, July 5-7, 2022, Cambridge, MA, USA*, volume 230 of *LIPIcs*, pages 15:1–15:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[4] G. Asharov and Y. Lindell. A full proof of the BGW protocol for perfectly secure multiparty computation. *J. Cryptol.*, 30(1):58–151, 2017.

[5] M. Backes, B. Pfitzmann, and M. Waidner. A composable cryptographic library with nested operations. In S. Jajodia, V. Atluri, and T. Jaeger, editors, *Proceedings of the 10th ACM Conference on Computer and Communications Security, CCS 2003, Washington, DC, USA, October 27-30, 2003*, pages 220–230. ACM, 2003.

[6] C. Badertscher, R. Canetti, J. Hesse, B. Tackmann, and V. Zikas. Universal composition with global subroutines: Capturing global setup within plain UC. In R. Pass and K. Pietrzak, editors, *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part III*, volume 12552 of *Lecture Notes in Computer Science*, pages 1–30. Springer, 2020.

[7] C. Badertscher, U. Maurer, D. Tschudi, and V. Zikas. Bitcoin as a transaction ledger: A composable treatment. In J. Katz and H. Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 324–356. Springer, 2017.

[8] B. Barak, R. Canetti, Y. Lindell, R. Pass, and T. Rabin. Secure computation without authentication. *J. Cryptol.*, 24(4):720–760, 2011.

[9] Z. Beerliová-Trubíniová, M. Fitzi, M. Hirt, U. M. Maurer, and V. Zikas. MPC vs. SFE: perfect security in a unified corruption model. In R. Canetti, editor, *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008*, volume 4948 of *Lecture Notes in Computer Science*, pages 231–250. Springer, 2008.

[10] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In J. Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 1–10. ACM, 1988.

[11] E. Boyle, R. Cohen, D. Data, and P. Hubácek. Must the communication graph of MPC protocols be an expander? In H. Shacham and A. Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 243–272. Springer, 2018.

[12] E. Boyle, R. Cohen, and A. Goel. Breaking the o($\sqrt{}$ n)-bit barrier: Byzantine agreement with polylog bits per party. In A. Miller, K. Censor-Hillel, and J. H. Korhonen, editors, *PODC '21: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, July 26-30, 2021*, pages 319–330. ACM, 2021.

[13] J. Camenisch, S. Krenn, R. Küsters, and D. Rausch. iuc: Flexible universal composability made simple. In S. D. Galbraith and S. 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.

[14] R. Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptol.*, 13(1):143–202, 2000.

[15] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 136–145. IEEE Computer Society, 2001.

[16] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, December 2005. Latest version at https://ia.cr/2000/067.

[17] R. Canetti, Y. Dodis, R. Pass, and S. Walfish. Universally composable security with global setup. In S. P. Vadhan, editor, *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, volume 4392 of *Lecture Notes in Computer Science*, pages 61–85. Springer, 2007.

[18] N. Chandran, P. Forghani, J. Garay, R. Ostrovsky, R. Patel, and V. Zikas. Universally composable almost-everywhere secure computation. *J. Cryptol.*, 39(1):12, 2026.

[19] N. Chandran, P. Forghani, J. A. Garay, R. Ostrovsky, R. Patel, and V. Zikas. Universally composable almost-everywhere secure computation. In D. Dachman-Soled, editor, *3rd Conference on Information-Theoretic Cryptography, ITC 2022, July 5-7, 2022, Cambridge, MA, USA*, volume 230 of *LIPIcs*, pages 14:1–14:25. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[20] N. Chandran, J. A. Garay, and R. Ostrovsky. Improved fault tolerance and secure computation on sparse networks. In S. Abramsky, C. Gavoille, C. Kirchner, F. M. auf der Heide, and P. 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.

[21] N. Chandran, J. A. Garay, and R. Ostrovsky. Almost-everywhere secure computation with edge corruptions. *J. Cryptol.*, 28(4):745–768, 2015.

[22] D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In J. Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 11–19. ACM, 1988.

[23] A. Choudhury, A. Patra, B. V. Ashwinkumar, K. Srinathan, and C. P. Rangan. Secure message transmission in asynchronous networks. *J. Parallel Distributed Comput.*, 71(8):1067–1074, 2011.

[24] D. Dolev. Unanimity in an unknown and unreliable environment. In *22nd Annual Symposium on Foundations of Computer Science, Nashville, Tennessee, USA, 28-30 October 1981*, pages 159–168. IEEE Computer Society, 1981.

[25] D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly secure message transmission. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I*, pages 36–45. IEEE Computer Society, 1990.

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

[27] C. Dwork, D. Peleg, N. Pippenger, and E. Upfal. Fault tolerance in networks of bounded degree (preliminary version). In J. Hartmanis, editor, *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 370–379. ACM, 1986.

[28] M. Fitzi and J. A. Garay. Efficient player-optimal protocols for strong and differential consensus. In E. Borowsky and S. Rajsbaum, editors, *Proceedings of the Twenty-Second ACM Symposium on Principles of Distributed Computing, PODC 2003, Boston, Massachusetts, USA, July 13-16, 2003*, pages 211–220. ACM, 2003.

[29] M. Fitzi, M. Hirt, and U. M. Maurer. Trading correctness for privacy in unconditional multi-party computation (extended abstract). In H. Krawczyk, editor, *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, volume 1462 of *Lecture Notes in Computer Science*, pages 121–136. Springer, 1998.

[30] J. A. Garay, A. Kiayias, and H. Zhou. A framework for the sound specification of cryptographic tasks. In *Proceedings of the 23rd IEEE Computer Security Foundations Symposium, CSF 2010, Edinburgh, United Kingdom, July 17-19, 2010*, pages 277–289. IEEE Computer Society, 2010.

[31] J. A. Garay and R. Ostrovsky. Almost-everywhere secure computation. In N. P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965 of *Lecture Notes in Computer Science*, pages 307–323. Springer, 2008.

[32] J. A. Garay and K. J. Perry. A continuum of failure models for distributed computing. In A. Segall and S. Zaks, editors, *Distributed Algorithms, 6th International Workshop, WDAG '92, Haifa, Israel, November 2-4, 1992, Proceedings*, volume 647 of *Lecture Notes in Computer Science*, pages 153–165. Springer, 1992.

[33] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In A. V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 218–229. ACM, 1987.

[34] J. Griggio. *Perfectly secure message transmission protocols with low communication overhead and their generalization*. PhD thesis, Universiteit Leiden, 2012.

[35] M. Hirt and U. M. Maurer. Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract). In J. E. Burns and H. Attiya, editors, *Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing, Santa Barbara, California, USA, August 21-24, 1997*, pages 25–34. ACM, 1997.

[36] D. Hofheinz and V. Shoup. GNUC: A new universal composability framework. *J. Cryptol.*, 28(3):423–508, 2015.

[37] S. Jayanti, S. Raghuraman, and N. Vyas. Efficient constructions for almost-everywhere secure computation. In A. Canteaut and Y. Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II*, volume 12106 of *Lecture Notes in Computer Science*, pages 159–183. Springer, 2020.

[38] J. Katz, U. Maurer, B. Tackmann, and V. Zikas. Universally composable synchronous computation. In A. Sahai, editor, *Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*, volume 7785 of *Lecture Notes in Computer Science*, pages 477–498. Springer, 2013.

[39] V. King and J. Saia. From almost everywhere to everywhere: Byzantine agreement with $\tilde{o}(n^{3/2})$ bits. In I. Keidar, editor, *Distributed Computing, 23rd International Symposium, DISC 2009, Elche, Spain, September 23-25, 2009. Proceedings*, volume 5805 of *Lecture Notes in Computer Science*, pages 464–478. Springer, 2009.

[40] K. Kurosawa and K. Suzuki. Truly efficient 2-round perfectly secure message transmission scheme. In N. P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965 of *Lecture Notes in Computer Science*, pages 324–340. Springer, 2008.

[41] L. Lamport, R. E. Shostak, and M. C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[42] A. Lubotzky, R. Phillips, and P. Sarnak. Explicit expanders and the ramanujan conjectures. In J. Hartmanis, editor, *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 240–246. ACM, 1986.

[43] U. Maurer and R. Renner. Abstract cryptography. In B. Chazelle, editor, *Innovations in Computer Science - ICS 2011, Tsinghua University, Beijing, China, January 7-9, 2011. Proceedings*, pages 1–21. Tsinghua University Press, 2011.

[44] J. Nielsen. *On Protocol Security in the Cryptographic Model*. PhD thesis, University of Aarhus, 2003.

[45] M. C. Pease, R. E. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980.

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

[47] G. Spini and G. Zémor. Perfectly secure message transmission in two rounds. In M. Hirt and A. D. Smith, editors, *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part I*, volume 9985 of *Lecture Notes in Computer Science*, pages 286–304, 2016.

[48] K. Srinathan, A. Narayanan, and C. P. Rangan. Optimal perfectly secure message transmission. In M. K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology-Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 545–561. Springer, 2004.

[49] E. Upfal. Tolerating linear number of faults in networks of bounded degree. In N. C. Hutchinson, editor, *Proceedings of the Eleventh Annual ACM Symposium on Principles of Distributed Computing, Vancouver, British Columbia, Canada, August 10-12, 1992*, pages 83–89. ACM, 1992.

[50] A. C. Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3-5 November 1982*, pages 160–164. IEEE Computer Society, 1982.

A Further Related Work

Here we continue our discussion of related work from Section 1.1.

A.1 History of SMT

The (perfectly) secure message transmission (SMT) problem was first studied by Dolev *et al.* [25], and we already introduced it in Section 1.1. Dolev *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). We use 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 [46], 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.* [48] demonstrated a lower bound of $\Omega(n)$ on the transmission rate; Agarwal *et al.* [1] constructed a protocol with the optimal transmission rate, but the computational costs are exponential; while Kurosawa and Suzuki [40] 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 [34] was able to reduce the transmission rate to $6n + o(n)$. Finally, Spini and Zémor [47] further reduced the transmission rate to $5n + o(n)$, while also improving the communication complexity to $O(n^2 \log n)$.

A.2 Related Models

We start with hybrid failure models (e.g., [32, 29]), 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.* [2], which technically considers 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, and the main motivation is to address the counterintuitive fact that standard protocols often do not provide privacy against other honest parties. While this notion of "friends-and-foes" security has special significance in the context of AE remote SMT, where a secret message shared over not necessarily disjoint paths may leak to an honest node that appears on sufficiently many of those paths, for simplicity, we do not address this in our treatment.

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 [39] (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 still to obtain full (not AE) agreement. The same approach is also explored by Boyle *et al.* [11], who investigated special characteristics (in particular, expansion) of the communication graph that is dynamically determined as a part of the protocol. In a recent follow-up, Boyle *et al.* [12] 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 Graph Theory Background

Preliminaries. We consider undirected graphs $G = (V, E)$ with order $|V| = n$. Parties will correspond to vertices and communication links to edges. The *degree* of a vertex $v \in V$ is defined as the number of edges in E that are incident on v (i.e., the number of neighbors of v). The *degree* of a graph is the maximum degree of a vertex in that graph, and the graph is said to be *d-regular* if all its vertices have degree d. In a *multigraph*, multiple edges are allowed between two vertices (i.e., E is a multiset).

Bipartite multigraphs are multigraphs whose vertex set V can be partitioned into a left set L of vertices and a right set R of vertices, such that no two left vertices are connected by an edge and no two right vertices are connected by an edge. These graphs are said to be *d-left-regular* when the degree of every left vertex is d, and *D-right-regular* when the degree of every right vertex is D. A d-left-regular bipartite multigraph with n left vertices and m right vertices can be described by a function $\Gamma\colon [n] \times [d] \to [m]$; in this case, for a given subset $S \subseteq [n]$ of (indices of) left vertices, we write $\Gamma(S)$ for the set of (indices of) right vertices that are neighbors of vertices in S. In case $n = m$, we say that the bipartite multigraph is *balanced*.

A path γ between two vertices u and v in a graph is *simple* if γ does not traverse any vertex more than once. Two (or more) paths are said to be *vertex-disjoint* if they do not share any vertices, except possibly their endpoints. A *clique* is a fully connected graph (with every pair of vertices connected by an edge).

Butterfly networks. For $m \in \mathbb{N}$, the *m-butterfly* is a graph $G_n = (V, E)$ with $n = m2^m$ vertices. The vertex set consists of pairs (a, i) for $0 \leq a \leq m-1$ and $0 \leq i \leq 2^m - 1$. Two vertices (a, i) and (b, j) are connected if and only if $b \equiv a + 1 \pmod{m}$ and either $i = j$ or i differs from j only in the a'th least significant bit.

Ramanujan graphs. Given a d-regular graph G, let $\lambda(G)$ be the maximum absolute value of any eigenvalue of the adjacency matrix of G other than d. In case $\lambda(G) \leq 2\sqrt{d-1}$, we say that G is a *Ramanujan graph*. We note that explicit constructions of Ramanujan graphs exist, for any $d = p + 1$, p a prime [42].

Expander graphs. Given an n-vertex graph $G = (V, E)$ and subset $S \subseteq V$ of vertices, let $e(S, \bar{S})$ be the number of edges between vertices in S and vertices in the complement of S in V (i.e., the number of edges leaving S). The *edge expansion* of G is defined as

$$h(G) = \min_{S \subset V, |S| \leq \frac{n}{2}} \frac{e(S, \bar{S})}{|S|}.$$

A d-regular graph G is an *(n, d, ϵ)-expander* if $h(G) \geq \epsilon$.

Similarly, a balanced, d-left-regular bipartite multigraph $\Gamma\colon [n] \times [d] \to [n]$ is an *(n, d, l, A)-bipartite-expander* if

$$\min_{S \subseteq [n], |S| \leq l} \frac{|\Gamma(S)|}{|S|} \geq A.$$

C Functionalities and Protocols

Protocol $\Pi_{\text{DDWY}}(\vec{\gamma}, m)$[a]

1. **(Phase 1)** The sender S sends $n\tau+1$ strong pads $SP_1, SP_2, \ldots, SP_{n\tau+1}$ to the receiver R, where n is the number of wires in $\vec{\gamma}$ and $\tau = \lceil \frac{n}{2} \rceil - 1$. To send each strong pad, S chooses a random polynomial $f(x) \in \mathbb{Z}_q(x)$ of degree τ and computes the pad $p = f(0)$. Then for each $i \in [n]$, S chooses an additional random polynomial $h_i(x) \in \mathbb{Z}_q(x)$ of degree τ such that $h_i(0) = f(i)$. Finally, for each $i \in [n]$, S sends $h_i(\cdot)$ along with a vector of checking pieces $C_i = (c_{1i}, c_{2i}, \ldots, c_{ni})$ to R on wire γ_i, where $c_{ji} = h_j(i)$ for all $j \in [n]$.

2. **(Phase 2)** For each $k \in [n\tau+1]$, let T_k denote what is received by R in the attempted transmission of SP_k, and $g_i(\cdot)$ and $D_i = (d_{1i}, d_{2i}, \ldots, d_{ni})$ denote the possibly corrupted information received on wire γ_i (any wires on which syntactically incorrect messages were received are thrown out). If for any T_a the received shares $\{g_i(0)\}$ can be interpolated by a polynomial of degree τ, then R computes the pad p_a and sends "a, OK" to S over the authenticated channel. Otherwise, R finds a k such that $\{\text{conflicts of } T_k\} \subseteq \cup_{l \neq k}\{\text{conflicts of } T_l\}$, where a pair (i, j) is called a conflict of T_k if $d_{ji} \neq g_j(i)$. Then R sends k and all T_l, $l \neq k$ back to S over the authenticated channel.

3. **(Phase 3)**

   - If "a, OK" was received over the authenticated channel in phase 2, then S sends $z = m + p_a$ to R over the authenticated channel. Otherwise, S performs error detection on all T_l's received from R, and sends any detected faults and $z = m + p_k$ to R over the authenticated channel.
   - If R previously sent "a, OK" to S in phase 1, then s/he outputs $m = z - p_a$. Otherwise, R corrects the faults in the retained T_k to interpolate the pad p_k, and outputs $m = z - p_k$.

[a] The protocol works for messages in the field $\mathcal{M} = \mathbb{Z}_q$ for prime $q > n$, and all the arithmetic is modular.

Figure 20: The SMT protocol from [25].

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

1. The sender S sends n uniformly random bit strings $R_1, R_2, \ldots, R_n$ of length $15l$ to the receiver R on wires $\gamma_1, \gamma_2, \ldots, \gamma_n$, respectively. Let $R'_1, R'_2, \ldots, R'_n$ be the strings received by R; any wires for which $|R'_i| \neq 15l$ are thrown out.

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 in 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 $1 \leq i \leq s$.[a] Then R outputs $m' = m'_1 \oplus m'_2 \oplus \cdots \oplus m'_s$.

[a] Recall that E and D are respectively the encoding and decoding algorithms for an error-correcting code with appropriate parameters.

Figure 21: The SMT-PD protocol from [31].

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

1. Upon receiving input $(\text{SEND}, sid, W_i, v_i)$ from $\mathcal{Z}$ in round ρ (which is the same for all W_i), where $sid = (P_s, P_r, sid')$ and $\{P_s, P_r\} = \{S, R\}$, party P_s sends $(\text{SEND}, sid_i^1, v_i)$ 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 (FETCH, sid_i^1) to $\mathcal{F}_{\text{SC}}^{S,R,\vec{W}}$. Upon receiving back $(\text{SENT}, sid_i^1, m_i)$, W_i sends $(\text{SEND}, sid_i^2, m_i)$ 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 (FETCH, sid, W_i) from $\mathcal{Z}$ in round $\rho + 2$, party P_r sends (FETCH, sid_i^2) to $\mathcal{F}_{\text{SC}}^{S,R,\vec{W}}$. Upon receiving back $(\text{SENT}, sid_i^2, m_i')$, P_r outputs $(\text{SENT}, sid, W_i, m_i')$ to $\mathcal{Z}$.

Figure 22: The wire communication protocol.

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

1. Upon receiving input (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 (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 (FETCH, sid_i) to $\mathcal{F}_{\text{SC}}^{G_n}$. Upon receiving back (SENT, sid_i, m), P_i sends $(\text{SEND}, sid_{i+1}, m)$ 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 (FETCH, sid) from $\mathcal{Z}$ in round $\rho + k$, the receiver R sends (FETCH, sid_k) to $\mathcal{F}_{\text{SC}}^{G_n}$. Upon receiving back (SENT, sid_k, m'), R outputs (SENT, sid, m') to $\mathcal{Z}$.

Figure 23: The remote secure channel protocol over graph G_n.

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.

Figure 24: The MPC functionality.

D Proofs Omitted from the Main Body

For convenience, we restate the propositions and theorems before proving them.

D.1 Proof of Proposition 3

Proposition 3. *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.*

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 $\mathcal{Z}$ 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 $(\textsc{SendLeak}, sid, W_i, \hat{m}_i)$ 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 $\textsc{Fetch}$ input from $\mathcal{Z}$ in round $\rho + 2$. If P_r is corrupted by $\mathcal{A}$, then $\mathcal{S}$ can wait to receive $(\textsc{FetchLeak}, sid, W_i)$ 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 $\textsc{InflSend}$ and $\textsc{Output}$ 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. □

D.2 Proof of Theorem 7

Theorem 7. *Protocol* $\Pi_{\text{AUTH}}(S, R, \vec{W})$ *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. [Sketch] 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 $\mathcal{W}_{\text{AE}}^{\mathscr{D}_{\text{PSMT}}}(\mathcal{F}_{\text{AUTH}}^{\{S,R\},\mathsf{rnd}})$ and $\mathcal{S}$. The simulator $\mathcal{S}$ is very similar to the simulator that was constructed in the proof of Theorem 4. However, $\mathcal{S}$ now interacts with a *wrapped* $\mathcal{F}_{\text{AUTH}}$ 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 $\mathcal{F}_{\text{AUTH}}$ functionality). Once again, the simulation is perfect. □