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.

2012 ACM Subject Classification Theory of computation → Cryptographic protocols; Security and privacy → Information-theoretic techniques; Security and privacy → Formal security models

Keywords and phrases Secure multi-party computation, universal compositability, almost-everywhere secure computation, sparse graphs, secure message transmission

Digital Object Identifier 10.4230/LIPIcs.ITC.2022.14

Funding Juan Garay: Work partially supported by NSF grants CNS-2001082 and CNS-2055694. Rafail Ostrovsky: 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
14:2 Universally Composable Almost-Everywhere Secure Computation

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.

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

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

1 Introduction

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

Despite great progress on the problem since it was first introduced and proven feasible [44, 27, 8, 18] 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 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 [6]: The adversary is able to break down the player set into connected components, so that parties in different connected components compute different instances of the function with inputs from the component – and all other inputs chosen by the adversary, and potentially different for each component. Even this weak form of security is only achievable for computationally bounded adversaries; if one is after information-theoretic (aka unconditional) security, where the adversary is unbounded, then the above guarantee is too much to ask for.

Notwithstanding, even in the latter case, where the adversary cannot disconnect the network, the situation is trickier than one might expect. Indeed, if a PKI-like setup is not assumed\(^1\) then it is known that secure communication between any two parties requires the existence of \( O(n) \) paths among them (known to or discoverable by the receiver), the majority of which must remain uncorrupted. This is the well-known secure message transmission (SMT) problem [20]. 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 [25] introduced the properties of almost-everywhere MPC (AE-MPC), which extended the concept of AE reliable communication previously studied by Dwork et al. [21]. In a nutshell, the paradigm of almost-everywhere

\(^1\) A PKI trivializes this case as a complete graph can be built by gossip (i.e., flooding) of signed messages.
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 which does not leave more doomed parties than the
underlying AE-SMT construction. Given the state of the art, such a modular statement
would be impossible, and one would need to prove AE-MPC security from scratch. Instead,
a simulation-based treatment in one of the composable security frameworks would inherit a
modular composition theorem making such statements tractable and simpler.

This work’s main goal is to derive such a treatment in the Universal Composability
(UC) framework of Canetti [13]. A major challenge, which we tackle, is to obtain a generic
definition of AE-security which can be applied to any type of functionality and captures
both AE-communication and AE-computation, two primitives whose treatment has been
very different. In fact, we achieve this goal by introducing a generic, composition-preserving
transformation from a secure variant of a functionality to its AE-secure counterpart. We
show that the derived AE-secure functionalities for secure communication (AE-RMT and
AE-SMT) and for secure MPC (AE-MPC): (1) preserve all the desired properties of the
previous definitions, and (2) are realized by (straightforward UC adaptations of) classical
AE-secure protocols. Since our treatment preserves composability of the (AE-)security
statements, we obtain, as a simple corollary, the first simulation-based proof of AE-MPC.

In passing, we note that although we adopt the language of UC, our definitional framework
is generic and can be applied to any of the main-stream composable security frameworks for
cryptographic protocols [3, 15, 37, 31, 11, 4]. Before providing more details on our results,
we first provide some necessary literature background.

1.1 Related Work

The origins of the “almost-everywhere” (AE) notion can be traced back to the work of Dwork
et al. [21], who considered the task of Byzantine agreement [39, 36] 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 [19] 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. [21] 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 AE agreement to 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 most important is a constant-degree graph admitting an AE-RMT scheme tolerating $t = O(n / \log n)$ corruptions while dooming $O(t)$ nodes. Several follow-up works have obtained improved parameters for AE-RMT (and thus also for AE agreement). Upfal [43] gave a transmission scheme tolerating $t = O(n)$ corruptions and dooming $O(t)$ nodes in a network of constant degree, which is the optimal set of parameters, but the protocol is inefficient. Chandran et al. [17] proposed a scheme tolerating $t = O(n)$ corruptions and dooming $O(t / \log n)$ nodes in a network of polylogarithmic degree. Most recently, Jayanti et al. [32] used the probabilistic method to show the existence of a logarithmic-degree graph admitting an AE-RMT scheme with the same parameters, strictly improving the [17] result.

Due to the results in [19, 20], 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 [25] introduced the notion of AE-MPC, which guarantees correctness and privacy only for the privileged parties\(^2\).

“Regular” information-theoretic MPC (i.e., MPC over a complete network) requires $t < n/3$ [8, 18]. 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. [21] 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).

The original SMT problem [20] considers two honest parties, a sender $S$ and a receiver $R$, connected by $n$ disjoint “wires”. The task is for $S$ to send a message to $R$ in the presence of a computationally unbounded adversary $A$ 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 $A$. 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. [20] showed that SMT is also possible if and only if $t < n/2$. Since their initial feasibility result, much more efficient protocols have been introduced [40, 42, 1, 35, 28, 41].

The SMT-PD model overcomes the necessity of $2t + 1$ wires in SMT by allowing access to an authentic and reliable public channel. Given such a channel (which can be constructed using, e.g., a broadcast protocol), Garay and Ostrovsky [25] gave a protocol that is secure as long as one wire 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 showed how to construct graphs that admit SMT-PD from any of the networks in

\(^2\) 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.
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, dooming the same number of parties as the underlying network.

To reiterate, all the above constructions are shown secure in a property-based manner. Other related notions include hybrid failure models (e.g., [26, 23]) and the model of Alon et al. [2]. In the AE setting, adversarial corruptions also have the effect of indirectly influencing the behavior of some of the honest parties (those who are doomed), but 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 [34] and follow-ups (e.g., [9, 10]), who considered full (not AE) Byzantine agreement over complete networks, but without all-to-all communication.

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 [13] and prove that the (UC adaptation of) existing AE-secure communication/computation protocols achieve this definition. We emphasize that all of our constructions tolerate adaptive corruptions.

There are several challenges associated with such a task. First, as should be evident from the above discussion, the related literature – from RMT/SMT, to Byzantine agreement, to MPC, and even their AE counterparts – treats the underlying network in different ways: e.g., in MPC, the network is typically a complete graph of point-to-point channels, 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.

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

An attempt to fix the above issue would be to define weaker corruption types corresponding to the flexible guarantees offered to the doomed parties. This, however, is also problematic, as corruptions in UC are by default known to (and declared by) the adversary/environment, whereas the actual identities of doomed parties are not, and depend on the behavior of the adversary (not just the identities of malicious parties). In particular, an adversary following, 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, its behavior still 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 which 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 this new 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. This way, 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 [5]: 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 \( (\alpha, \beta) \) 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 [29, 7]. That is, we parameterize the wrapper with a corruption/doom structure (“doom structure” for short), consisting 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 representation.

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

**Almost-Everywhere RMT and SMT.** We start in Section 3 by 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. To apply a unified treatment, we cast the problem by modeling each wire with a (corruptible) dummy party called a “wire-party,” which simply relays messages between \(S\) and \(R\). In Section 3.1, we confirm that classical RMT/SMT protocols [20] 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 that is parameterized by a doom structure as defined above. We can then state the AE-security of RMT/SMT protocols, by using a simple doom structure like the one that allows dooming \(S\) or \(R\) 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 [25]. We model the public channel using access to the same functionality that we use to capture RMT security. Looking ahead, we will need SMT-PD when we want to elevate RMT to SMT over some classes of sparse graphs.
Almost-Everybody Secure Computation. In Section 4, we consider 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 even against corrupted majorities of wires, can also be used to model AE-security of the all-to-all versions of those tasks. In particular, in Section 4.1 we use the same ideal functionalities and wrapper (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 [21, 43, 17, 32], 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 a perfect SMT protocol from [20] can be adapted to realize perfectly secure AE-SMT over a class of sparse graphs constructed in [21]. 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 [25] to realize AE-SMT over the graphs in [43, 17, 32], at the cost of obtaining only statistical UC security.

Almost-Everybody Secure Computation. Lastly, we study the composability of AE-security guarantees, with the ultimate goal of realizing AE-MPC. In Section 5.1, we 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. We emphasize that this AE compiler need not replace all of the hybrids with AE-wrapped versions using the same doom structure; thus, we are able to explain, for example, what happens when a protocol uses subprotocols that 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. 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 wrapped hybrid. In Section 5.2, we apply this corollary to obtain the first simulation-based proof of AE-MPC, over any of the classes of sparse graphs considered in the AE agreement literature [21, 43, 17, 32]. Depending on which class of sparse graphs is used, we obtain either perfect or statistical UC security.

Next, we review some preliminaries. Due to space limitations, some of the functionalities, protocols, and proofs are presented in the appendix or in the full version of the paper [16].

2 Preliminaries

2.1 UC Basics

We briefly summarize the UC framework [13] 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 $\pi$ consists of a series of activations of ITIs, starting with the environment $Z$ who provides inputs to and collects outputs from the parties and the adversary $A$; parties can also give input to and collect output from sub-parties, and $A$ can communicate with parties via messages. Corruption of parties is modeled by a special corrupt message sent from $A$ to the party; upon receipt of this message, the party sends its entire local state to $A$, and in all future activations follows the instructions of $A$. Note that a party $p_i$ can only be corrupted once $A$ receives a special (corrupt $p_i$) input from $Z$. Denote by $exec_{\pi,A,Z}$ the probability distribution ensemble corresponding to the output of $Z$ at the end of an execution of $\pi$ with adversary $A$. The ideal-world process for functionality $F$ is simply defined as an execution of the ideal protocol $\text{IDEAL}_F$, in which the so-called “dummy” parties just forward inputs from $Z$ to $F$ and forward outputs from $F$ to $Z$. The corresponding ensemble is denoted by $\text{IDEAL}_{F,S,Z}$. 
We are interested in unconditional security. Thus, we say that a protocol $\pi$ UC-realizes an ideal functionality $F$ if for any computationally unbounded adversary $A$, there exists a simulator $S$ (polynomial in the complexity of $A$) such that for any computationally unbounded environment $Z$, we have $\text{ideal}_{F,S,Z} \equiv \text{exec}_{\pi,A,Z}$. Statistical UC-realization requires only that the two ensembles be indistinguishable. When $\pi$ is a $(G_1, \ldots, G_n)$-hybrid protocol (i.e., making subroutine calls to $\text{ideal}_{G_1, \ldots, G_n}$), we say that $\pi$ UC-realizes $F$ in the $(G_1, \ldots, G_n)$-hybrid model. It turns out that (regular) UC-realization is equivalent to UC-realization with respect to the “dummy” adversary $D$, which simply follows the instructions of $Z$ on which messages to send, and reports all received messages to $Z$.

We will assume synchronous computation (i.e., our protocols proceed in rounds), and the adversary is assumed to be rushing. Although our treatment is in the (G)UC setting, to avoid over-complicating the exposition we use the standard round-based language of, e.g., [12, 38]. Notwithstanding, such specifications can be translated to the synchronous UC model of Katz et al. [33] by assuming a clock functionality and bounded (zero) delay channels.

### 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 UC formulations.

**Definition 1 (SMT).** A protocol $\Pi$ achieves $(t)$-SMT if it allows $S$ to send a message $m \in M$ to $R$ such that the following hold for any adversary $A$ corrupting up to $t$ of the wires:

- **Reliability:** $R$ correctly outputs $m' = m$.
- **Secrecy:** $A$ learns no information about $m$.

We define RMT by omitting the secrecy property, and AE-RMT and AE-SMT are defined by only requiring the properties to hold for privileged $S$ and $R$ (over some sparse graph).

For simplicity, we will use the 3-phase SMT protocol $\Pi_{\text{SMT}}(\gamma, \tau, m)$ presented in Figure 3 (Appendix A), which is essentially the FastSMT protocol from [20]. The $n$ wires are denoted by $\gamma = (\gamma_1, \ldots, \gamma_n)$, and $\tau = \lceil \frac{2}{3} \rceil - 1$ 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 over all wires and having $R$ take majority.

We will sometimes need the SMT-PD protocol $\Pi_{\text{PD-SMT}}(\gamma, \text{Pub}, m, l)$ presented in Figure 4 (Appendix A), which was given in [25] and tolerates $n - 1$ wire corruptions, allowing access to a public channel $\text{Pub}$ and allowing a small probability of error (determined by $l$).

Finally, we present the security definition for (property-based) AE-MPC that was given in [25]. Recall that $W$ is the set of privileged nodes, as a function of the set of corruptions.

**Definition 2 (AE-MPC, [25]).** An $n$-player two-phase protocol $\Pi$ 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 $A$ 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 all $P_i$ there is a uniquely defined value $x^*_i$; if $P_i \in W$, then $x^*_i = x_i$.
- **Privacy:** For all $P_i \in W$, $x^*_i$ is information-theoretically hidden.

**Computation phase:**

- **Correctness:** All $P_i \in W$ output $f(x^*_1, \ldots, x^*_n)$.
- **Privacy:** For all $P_i \in W$, no information about $x^*_i$ beyond what can be inferred from the output of the corrupted parties leaks to $A$ by this phase.
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 [20]. 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.

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 $W_1, \ldots, W_n$ (or $\vec{W}$ 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 $F_{SC}^{S,R,\vec{W}}$ (see the full version [16] for a formal specification), which provides secure communication between an honest $S$ or $R$ and an honest wire-party over a single round. 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.$^3$

For convenience, we use $F_{sc}^{S,R,\vec{W}}$ to realize the channel functionality $F_{wc}^{S,R,\vec{W}}$ presented in Figure 5 (Appendix A), which abstracts the process of sending a message to a wire-party, who then forwards it to $S$ or $R$. The functionality actually allows sending a potentially different message through each wire-party in parallel, and it provides security for a given message as long as $S$, $R$, and the wire-party in question are all honest. Since we are considering virtual wires that consist of just one intermediate node, the functionality requires two rounds to generate output. In $F_{wc}^{S,R,\vec{W}}$ (and all of our functionalities), $l(\cdot)$ refers to length and $\text{Infl}$ is short for “influence” (see, e.g., [24]).

We can use the protocol $\Pi_{wc}(S, R, \vec{W})$, which simply routes each message $m_i$ from $S$ to $R$ (or $R$ to $S$) via $W_i$ using two instances of $F_{sc}^{S,R,\vec{W}}$, to realize $F_{wc}^{S,R,\vec{W}}$. The proof, as well as a formal specification of $\Pi_{wc}(S, R, \vec{W})$, can be found in the full version [16].

▶ Proposition 3. Protocol $\Pi_{wc}(S, R, \vec{W})$ UC-realizes $F_{wc}^{S,R,\vec{W}}$ in the $F_{sc}^{S,R,\vec{W}}$-hybrid model.

3.1 Universally Composable RMT and SMT

We model the task of RMT in UC with the authenticated channel functionality $F_{auth}^{P,\text{rnd}}$ (see the full version [16] for a formal specification), which is essentially Canetti’s $F_{auth}$ [14] with synchrony (the $\text{rnd}$ parameter). There is also a parameter $P$ 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.

$^3$ Our RMT protocols only require 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.
To realize $\mathcal{F}^{\text{P}, \text{rand}}_{\text{AUTH}}$ 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 using a single instance of $\mathcal{F}^{S, R, \overline{W}}_{\text{WC}}$, and have the receiver (who may actually be $S$) take majority. We formally define protocol $\Pi_{\text{AUTH}}(S, R, \overline{W})$ and give a proof in the full version [16].

**Theorem 4.** Protocol $\Pi_{\text{AUTH}}(S, R, \overline{W})$ UC realizes $\mathcal{F}^{(S,R), \text{rand}}_{\text{AUTH}}$ for $\text{rand} = 2$ in the $\mathcal{F}^{S, R, \overline{W}}_{\text{WC}}$-hybrid model, against an adversary corrupting up to a minority of the wire-parties.

Next, we capture SMT in UC with the secure channel functionality $\mathcal{F}^{\text{P}, \text{rand}}_{\text{SMT}}$ (see the full version [16] for a formal specification), which is essentially Canetti’s $\mathcal{F}^{\text{SMT}}_{\text{SMT}}$ [14] with synchrony.

To realize $\mathcal{F}^{\text{S}, \text{rand}}_{\text{SMT}}$ in the wire-party model assuming only a minority of the wire-parties get corrupted, we can use protocol $\Pi_{\text{SMT}}(S, R, \overline{W})$ (outlined in the full version [16]), which is essentially the FastSMT protocol from [20] adapted for our UC treatment. That is, the sender (who may actually be $R$) runs protocol $\Pi_{\text{HowV}}(\gamma, \tau, m)$ (Figure 3) with the receiver, using $\mathcal{F}^{S, R, \overline{W}}_{\text{WC}}$ in phase 1 as a substitute for sending messages through the wires in $\gamma$, and separate instances of $\mathcal{F}^{(S,R), \text{rand}}_{\text{AUTH}}$ in phases 2 and 3 as a substitute for the authenticated channel.

**Theorem 5.** Protocol $\Pi_{\text{SMT}}(S, R, \overline{W})$ UC realizes $\mathcal{F}^{(S,R), \text{rand}}_{\text{SMT}}$ for $\text{rand} = 6$ in the ($\mathcal{F}^{S, R, \overline{W}}_{\text{WC}}, \mathcal{F}^{(S,R), \text{rand}}_{\text{AUTH}}$)-hybrid model, against an adversary corrupting up to a minority of the wire-parties.

### 3.2 Corrupted Majorities of Wire-Parties

In the wire-party model, $\mathcal{F}^{\text{AUTH}}_{\text{SMT}}$ and $\mathcal{F}^{\text{AUTH}}_{\text{SMT}}$ 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 (Figure 1) that allows parties to be marked as doomed according to the current set of corruptions. The wrapper accepts “doom” requests according to an adversary structure, and it processes them by simply having the underlying functionality treat doomed parties as fully corrupted. Recall that an adversary structure is a set of $c$-vectors of subsets of a participant set $\mathcal{P}$, where each component of a vector represents corruptions of a certain type. We consider adversary structures that consist of doubles of subsets, corresponding to a corrupted set and a doomed set, respectively, although the two may intersect$^4$. We call such structures doom structures.

In the wire-party model, we can realize wrapped $\mathcal{F}^{(S,R), \text{rand}}_{\text{AUTH}}$ and $\mathcal{F}^{(S,R), \text{rand}}_{\text{SMT}}$ with doom structure $\mathcal{D}_{\text{SMT}}$, defined as follows using participant set $\mathcal{P} = \{S, R, W_1, \ldots, W_n\}$:

$$\mathcal{D}_{\text{SMT}}(T, D) \in \mathcal{D}_{\text{SMT}} \text{ if and only if either } |T - \{S, R\}| < \frac{n}{2} \text{ and } D_i = \emptyset \text{ or } |T_i - \{S, R\}| \geq \frac{n}{2} \text{ and } D_i \subseteq \{S, R\}$$

**Theorem 6.** Protocol $\Pi_{\text{AUTH}}(S, R, \overline{W})$ UC realizes $\mathcal{W}^{(S,R), \text{rand}}_{\text{AE}}(\mathcal{F}^{(S,R), \text{rand}}_{\text{AUTH}})$ for $\text{rand} = 2$ in the $\mathcal{F}^{S, R, \overline{W}}_{\text{WC}}$-hybrid model, even against corrupted majorities of wire-parties.

To realize wrapped $\mathcal{F}^{(S,R), \text{rand}}_{\text{SMT}}$ define protocol $\Pi_{\text{SMT}}(S, R, \overline{W})$ by replacing invocations of $\mathcal{F}^{(S,R), \text{rand}}_{\text{AUTH}}$ in protocol $\Pi_{\text{SMT}}(S, R, \overline{W})$ with invocations of $\mathcal{W}^{(S,R), \text{rand}}_{\text{AE}}(\mathcal{F}^{(S,R), \text{rand}}_{\text{AUTH}})$.

**Theorem 7.** Protocol $\Pi_{\text{SMT}}(S, R, \overline{W})$ UC realizes $\mathcal{W}^{(S,R), \text{rand}}_{\text{AE}}(\mathcal{F}^{(S,R), \text{rand}}_{\text{SMT}})$ for $\text{rand} = 6$ in the ($\mathcal{F}^{S, R, \overline{W}}_{\text{WC}}, \mathcal{W}^{(S,R), \text{rand}}_{\text{AE}}(\mathcal{F}^{(S,R), \text{rand}}_{\text{AUTH}})$)-hybrid model, even against corrupted majorities of wire-parties.

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.

---

$^4$ This is a technicality, which simplifies some of our definitions and results. For example, the definition of AE-monotonicity (Section 5.1) would not be quite as short and intuitive otherwise.
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 $F_{\text{auth}}^{(S,R)}$ for some $\text{rnd}'$. Protocol $\Pi_{\text{SMT-PD}}(S, R, \tilde{W}, l)$ (see the full version [16] for a formal specification) is essentially the Pub-SMT protocol from [25] adapted for our UC treatment. In particular, the sender transmits a message $v$ to the receiver by essentially executing protocol $\Pi_{\text{Pub-SMT}}(\gamma, \text{Pub}, v, l)$ (Figure 4), where $F_{\text{wc}}^{S,R,\tilde{W}}$ is used in the first phase as a substitute for sending messages through the wires in $\gamma$, and separate instances of $F_{\text{auth}}^{(S,R)}$ are used in the remaining three phases as a substitute for $\text{Pub}$.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure1}
\caption{AE wrapper functionality.}
\end{figure}

Theorem 8. Protocol $\Pi_{\text{SMT-PD}}(S, R, \tilde{W}, l)$ statistically UC-realizes $F_{\text{smt}}^{(S,R)}$ for $\text{rnd}' = 2 + 3 \cdot \text{rnd}'$ in the $(F_{\text{wc}}^{S,R,\tilde{W}}, F_{\text{auth}}^{(S,R)})$-hybrid model, against an adversary corrupting all but one of the wire-parties.

4 Almost-Everywhere Remote RMT and SMT

In this section, we consider remote – i.e. over a sparse graph $G_n$ – RMT and SMT. As in Section 3, we model the network topology using the parameterized secure channel functionality $F_{\text{sc}}^{G_n}$ presented in Figure 6 (Appendix A), which provides secure channels only between parties that are connected in $G_n$. Instead of always working directly with $F_{\text{sc}}^{G_n}$, we also use it to realize the remote secure channel functionality $F_{\text{r-sc}}^{G_n}$ (see the full version [16] for a formal specification), the counterpart to $F_{\text{sc}}^{S,R,\tilde{W}}$ from Section 3. This functionality provides secure communication over a single path, as long as no node on the path is corrupted. Using protocol $\Pi_{\text{r-sc}}(G_n)$ (see the full version [16] for details), we can realize $F_{\text{r-sc}}^{G_n}$ by simply forwarding the message along the path, which leads to the following statement (proof omitted).

Proposition 9. Protocol $\Pi_{\text{r-sc}}(G_n)$ UC-realizes $F_{\text{r-sc}}^{G_n}$ in the $F_{\text{sc}}^{G_n}$-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 $F_{\text{auth}}^{(S,R)}$ functionality, using doom structures that are derived from the protocol and the underlying sparse graph.

Graphs of Constant Degree. We first describe a scheme due to Dwork et al. [21], 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 $\Gamma_{in}(v)$ and a fan-out set $\Gamma_{out}(v)$ of a fixed 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 to node $v$ by first sending it to all members of $\Gamma_{out}(u)$; each member then forwards the message to its connected (via a path) node in $\Gamma_{in}(v)$; and finally each member of $\Gamma_{in}(v)$ forwards the message to $v$, who simply takes majority.

Let $G_{DPPU} = (V_{DPPU}, E_{DPPU})$ be a graph that admits such a scheme. To realize wrapped $\mathcal{F}_{\text{AUTH}}$, we use protocol $\Pi_{\text{AUTH}}$ (outlined in the full version [16]), which is a straightforward adaptation of Upfal’s scheme in the $\mathcal{F}_{\text{SC}}^{\text{hybrid}}$-hybrid model, and the doom structure $D_{DPPU}$:

- For any corruption set $T_i$, let $D_{DPPU}(T_i)$ be a set of participants $P$ such that at least $\frac{1}{8}$ of the paths from $P$ to $\Gamma_{out}(P)$ or at least $\frac{1}{8}$ of the paths from $\Gamma_{in}(P)$ to $P$ are corrupted.
- Now, let $(T_i, D_i) \in D_{DPPU}$ if and only if $|T_i| < s/4$ and $D_i \subseteq D_{DPPU}(T_i)$.

**Theorem 10.** Protocol $\Pi_{DPPU}$ UC-realizes $\mathcal{W}_{\text{AUTH}}^{\text{hybrid}}(\mathcal{F}_{\text{AUTH}}^{\text{hybrid}})$ for some $\text{rnd} \in O(\log n)$ in the $\mathcal{F}_{\text{SC}}^{\text{hybrid}}$-hybrid model, against an adversary corrupting less than $s/4$ nodes.

Upfal [43] proposed an alternative transmission scheme for constant-degree graphs, which actually works over any graph; however, his optimal result is achieved only on constant-degree expander graphs with specific parameters. The main limitation of the scheme is that it is computationally expensive. Node $u$ transmits a message 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 at least one ID of a corrupted node, and the receiver can enumerate over all the possible corruption sets to recover the message.

Let $G_{upfal}^n = (V_{upfal}, E_{upfal})$ be a $d$-regular expander graph. To realize wrapped $\mathcal{F}_{\text{AUTH}}$, we use protocol $\Pi_{\text{AUTH}}^{\text{upfal}}$ (outlined in the full version [16]), which is a straightforward adaptation of Upfal’s scheme in the $\mathcal{F}_{\text{SC}}^{\text{hybrid}}$-hybrid model, and the following doom structure $D_{upfal}$:

- First, define $D_{upfal}(T_i)$ by the following iterative process: Starting with the set $S = T_i$, repeatedly add all participants $Q \notin S$ such that at least $\frac{1}{8}$ of $Q$’s neighbors are in $S$.
- Now, let $(T_i, D_i) \in D_{upfal}$ if and only if $|T_i| < t < 1/72n$ and $D_i \subseteq D_{upfal}(T_i)$.

**Theorem 11.** Protocol $\Pi_{upfal}^{\text{upfal}}$ UC-realizes $\mathcal{W}_{\text{AUTH}}^{\text{hybrid}}(\mathcal{F}_{\text{AUTH}}^{\text{hybrid}})$ for some $\text{rnd} \in O(\log n)$ in the $\mathcal{F}_{\text{SC}}^{\text{hybrid}}$-hybrid model, against an adversary corrupting less than $1/72n$ nodes.

Although the simulator we construct is inefficient, that seems reasonable since the protocol itself runs in exponential time.

**Graphs of Poly-Logarithmic Degree.** Chandran et al. [17] proposed an AE-RMT scheme over a randomly constructed graph of poly-logarithmic degree. A very high-level idea of their construction is to transmit a message via multiple paths, while also performing some sort of error correction along the way. Their graph is comprised of some randomly generated, overlapping, fully connected committees that are themselves connected via the butterfly network. They also assign and connect to each node a number of those committees as helpers. Node $u$ transmits a message to node $v$ by sending it to all of $u$’s helper committees, who then transmit it to $v$’s helper committees using the transmission scheme of Dwork et al. [21] at the committee level. Finally, $v$ takes majority over the values received from its helpers. As the message travels from one committee to another, error correction occurs using a differential agreement protocol [22] run by the nodes in the destination committee.

Let $G_{cgo}^n = (V_{cgo}, E_{cgo})$ be a graph constructed as above. To realize wrapped $\mathcal{F}_{\text{AUTH}}$, we use protocol $\Pi_{\text{AUTH}}^{\text{cgo}}$ (outlined in the full version [16]), which is a straightforward adaptation of the above scheme in the $\mathcal{F}_{\text{SC}}^{\text{hybrid}}$-hybrid model, and the following doom structure $D_{cgo}$:
First, for any set of corruptions $T_i$, let $D_{cgo}(T_i)$ be the set of all participants $P$ such that $P \in T_i$ or at least $\frac{1}{5}$ of $P$’s helper committees are unprivileged. A committee is considered corrupted if more than $\frac{1}{5}$ of its members are corrupted, and committees are categorized as unprivileged according to the $D_{upfal}(\cdot)$ function defined above.

Now, let $(T_i, D_i) \in \mathcal{D}_{cgo}$ if and only if corrupting the nodes in $T_i$ causes at most \( \frac{n \log^4 n}{4 \log(n \log^4 n)} \) committees to become corrupted, and $D_i \subseteq D_{cgo}(T_i)$.

Chandran et al. [17] proved that there exist constants $\alpha_{cgo}, \beta_{cgo}$ such that for any $T_i$ with $|T_i| < \alpha_{cgo} n$, it holds that $|D_{cgo}(T_i)| < \beta_{cgo}\frac{|T_i|}{\log n}$. For those constants we have:

\[ \text{Theorem 12. } \text{Protocol } \Pi_{rauth}^{cgo} \text{ UC-realizes } W_{ae}^{\mathcal{D}_{cgo}}(\mathcal{F}_{auth}^{\text{rand}}) \text{ for } \text{rand} \in O(\log n \cdot \log \log n) \text{ in the } \mathcal{F}_{SC}^{\alpha_{ae}} \text{-hybrid model, against an adversary corrupting less than } \alpha_{cgo} n \text{ nodes.} \]

Graphs of Logarithmic Degree. Jayanti et al. [32] recently proposed a transmission scheme over logarithmic-degree graphs. Their graphs consist of multiple layers that are all constructed using the same method but over randomly permuted sets of nodes. In each layer, nodes are partitioned into committees of size $s$ that are connected via the butterfly network and have Upfal’s [43] expander graph instantiated inside them. We call this family of graphs $\mathcal{G}_{jrv}$.

To transmit a message from node $u$ to node $v$, in each layer $u$ sends the message to all its committee members using Upfal’s transmission scheme, and then the committee transmits it to $v$’s committee using the transmission scheme of Dwork et al. [21] at the committee level. Finally, all of $v$’s committee members send the value to $v$ so that it can take majority over what is received from all the layers combined. There is also some type of error correction when messages travel from one committee to another.

Let $G_{jrv}^n = (V_{jrv}, E_{jrv}) \in \mathcal{G}_{jrv}$, where $|V_{jrv}| = n$. To realize wrapped $\mathcal{F}_{auth}^{\text{rand}}$, we use protocol $\Pi_{rauth}^{jrv}$ (outlined in the full version [16]), which is a straightforward adaptation of the above scheme in the $\mathcal{F}_{SC}^{\alpha_{ae}}$-hybrid model, and the following doom structure $\mathcal{D}_{jrv}$:

First, in each layer of $G_{jrv}^n$, if a committee contains more than $\frac{n}{4} s$ corruptions, call it bad. If the total number of bad committees in a layer exceeds $\frac{n}{4} s$, call the layer bad.

Next, for any set of corruptions $T_i$, let $D_{jrv}(T_i)$ be the set of all participants $P$ such that $P \in T_i$ or $P$ is doomed in more than $\frac{1}{5}$ layers among all the good layers. A node is considered doomed in a layer if it is located in a doomed committee (with respect to $D_{upfal}(\cdot)$) or is doomed itself within its committee (with respect to $D_{jrv}(\cdot)$).

Now, let $(T_i, D_i) \in \mathcal{D}_{jrv}$ if and only if corrupting the nodes in $T_i$ causes at most $\frac{1}{5}$ of the layers to become bad, and $D_i \subseteq D_{jrv}(T_i)$.

Jayanti et al. [32] proved there exists a graph $G_{jrv}^n \in \mathcal{G}_{jrv}$ and constants $\alpha_{jrv}, \beta_{jrv}$ such that for $T_i$ with $|T_i| < \alpha_{jrv} n$, it holds that $|D_{jrv}(T_i)| < \beta_{jrv}\frac{|T_i|}{\log n}$. For such a graph we have:

\[ \text{Theorem 13. } \text{Protocol } \Pi_{rauth}^{jrv} \text{ UC-realizes } W_{ae}^{\mathcal{D}_{jrv}}(\mathcal{F}_{auth}^{\text{rand}}) \text{ for some } \text{rand} \in O(\log n \cdot \log \log \log n) \text{ in the } \mathcal{F}_{SC}^{\alpha_{ae}} \text{-hybrid model, against adversaries corrupting less than } \alpha_{jrv} n \text{ nodes.} \]

4.2 AE Remote SMT

To achieve AE secure communication over the constant-degree graphs studied by Dwork et al. [21], we can apply the approach that we used in Section 3.2 to obtain AE-SMT in the wire-party model. That is, we can adapt protocol $\Pi_{smt}(S, R, \hat{W})$ to work over the three-phase paths in $G_{upfal}$, and the resulting protocol $\Pi_{rauth}^{upfal}$ (formally outlined in the full version [16]) realizes wrapped $\mathcal{F}_{smt}^{\text{rand}}$ for some $\text{rand}$ with the same doom structure $\mathcal{D}_{upfal}$ from Section 4.1. Let $\ell$ denote the maximum length of any of the three-phase paths.
Theorem 14. Protocol $\Pi_{\text{R-SMT}}^{\text{PPU}}$ UC-realizes $W_{\text{AE}}^{\text{bare}}(F_{\text{SMT}}^{V_{\text{bare}},2\cdot m+d+\ell})$ in the $(W_{\text{AE}}^{\text{bare}}(F_{\text{AUTH}}^{V_{\text{bare}}}), F_{\text{R-SC}}^{G_{\text{dppu}}})$-hybrid model for $\text{rnd} \in O(\log n)$, against an adversary corrupting less than $s/4$ nodes.

The above technique cannot in general be extended to other AE-RMT schemes, because it requires a majority of honest paths between any pair of privileged nodes to realize a secure link between them. Many transmission schemes, such as Upfal’s [43], do not guarantee such a property for privileged nodes. To realize AE-SMT using other transmission schemes, one approach is to use SMT-PD, which only requires a single honest path between sender and receiver to establish a secure channel, assuming access to a public channel. This approach can be used to make any AE-RMT scheme secure, since these schemes realize an authenticated channel (between privileged nodes) and guarantee at least one honest path between any pair of privileged nodes. The downside is that only statistical security is obtained.

We first introduce some notation. Given a doom structure $\mathcal{D}$ (with participant set $\mathcal{P}$), denote by $\text{dom}(\mathcal{D})$ the set of values that appear as a first component in $\mathcal{D}$ (in other words, the set of all corruption sets allowed by $\mathcal{D}$). Say that $\mathcal{D}$ is $t$-complete if $\max_{(T,D) \in \mathcal{D}} |T| = t$, and $T \in \text{dom}(\mathcal{D})$ for all $T \subseteq \mathcal{P}$ with $|T| \leq t$ (in other words, if all possible sets of corruptions of size at most $t$ are allowed by $\mathcal{D}$). Moreover, say that a doom structure $\mathcal{D}$ is $D$-monotone if whenever $(T_j,D_j) \in \mathcal{D}$ and $D_i \subseteq D_j$, it holds that $(T_j,D_i) \in \mathcal{D}$. We note that all of our doom structures are $t$-complete and $D$-monotone.

Now, let $G_n = (V,E)$ be a graph with polynomially many paths of length at most $\ell$ specified between every pair of nodes. Suppose we already know how to realize wrapped $F_{\text{AUTH}}^{V,\text{rnd}}$ for some $\text{rnd}$, with respect to a doom structure $\mathcal{D}_{\text{SMT-PD}}$ (with $\mathcal{P} = V$) that is $t$-complete and $D$-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 $V - T - \bigcup_{(T,D) \in \mathcal{D}_{\text{SMT-PD}}} D$ is completely contained in $V - T$. Then, we can realize wrapped $F_{\text{SMT}}^{V,\text{rnd}}$ using protocol $\Pi_{\text{R-SMT-PD}}(G_n)$ (formally outlined in the full version [16]), which is essentially protocol $\Pi_{\text{SMT-PD}}(S,R,W,l)$ from Section 3.3 adapted to work over the specified paths in $G_n$, and the same doom structure $\mathcal{D}_{\text{SMT-PD}}$. For such a doom structure we obtain the following statement:

Theorem 15. Protocol $\Pi_{\text{R-SMT-PD}}(G_n)$ statistically UC-realizes $W_{\text{AE}}^{\text{bare}}(F_{\text{SMT}}^{V,\ell+3})$ in the $(F_{\text{R-SC}}^{G_n}, W_{\text{AE}}^{\text{bare}}(F_{\text{AUTH}}^{V,\text{rnd}}))$-hybrid model against a $t$-adversary.

Observe that $t$-completeness allows for statements against threshold adversaries, and $D$-monotonicity is required in the simulation since the simulator can only doom parties one by one. According to [21], all the realizable doom structures for AE remote RMT satisfy the above condition. 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 almost-everywhere setting. We start by proving a composition theorem that shows how to compile a protocol II realizing some functionality $F$ with the help of several hybrids into an almost-everywhere version of II, by wrapping each hybrid with a potentially different doom structure $\mathcal{D}_i$. These structures can be arbitrary, subject only to a certain monotonicity property, although they must correspond to the same participant set (indeed, composition would not make much sense otherwise); the compiled protocol is then shown to realize a wrapped version of $F$, using a new doom structure $\mathcal{D}'$. Moreover, we allow the original protocol II to itself realize a wrapped functionality associated with some doom structure $\mathcal{D}$. This, along with the fact that the monotonicity property carries over to the new doom structure $\mathcal{D}'$, make the compiled protocol readily
Figure 2 The AE compiler.

Amenable to further composition. We conclude by applying a special case of the composition theorem to obtain AE-MPC over the sparse graphs that were considered in Section 4. Rather than constructing protocols from scratch, we simply apply our generic AE compiler to replace the secure channels that are used in standard MPC protocols with AE remote SMT.

5.1 A General Composition Theorem

Let us first introduce some notation. Say that a doom structure $D$ is AE-monotone if whenever $(T_i, D_i) \in D$ and $T_i \subseteq T_j$ for $T_j \in \text{dom}(D)$, it holds that $(T_j, D_i) \in D$. Different from the standard notion of monotonicity in the general adversary literature, AE-monotonicity captures the intuitive property that when additional parties are corrupted, parties that were previously doomed are still doomed (or newly corrupted). AE-monotonicity seems to be important for simulatability; for example, the simulator may want to make a doom request for a newly doomed party only after some additional parties are corrupted in the meantime, and in such a case the doom structure needs to admit that request. Fortunately, all of our doom structures are AE-monotone.

The AE compiler is shown in Figure 2. It takes as input a protocol $\Pi$ realizing some wrapped functionality $W_{\text{AE}}(F)$ in the $(F_1, \ldots, F_m)$-hybrid model and turns it into a protocol that works in the $(W_{\text{AE}}^{\mathcal{D}_1}(F_1), \ldots, W_{\text{AE}}^{\mathcal{D}_m}(F_m))$-hybrid model. Of course, the compiled protocol will not in general realize wrapped $F$ with the same doom structure $\mathcal{D}$. In the following theorem, we construct a new doom structure $\mathcal{D}'$ 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., $D$ as well as $\mathcal{D}_1, \ldots, \mathcal{D}_m$). Furthermore, the set of parties in the compiled protocol that are considered doomed (relative to $T'$) 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 $\Pi$ (the parties denoted by $A$). In fact, since $\Pi$ may already carry some level of AE-security, as captured by $D$, 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. Thus, we require that $T' \cup D(T')$ is also tolerated by $D$.

> **Theorem 16.** Let $\mathcal{D}, \mathcal{D}_1, \ldots, \mathcal{D}_m$ be AE-monotone doom structures over the same participant set $\mathcal{P}$. Let $T = \text{dom}(\mathcal{D})$ and $T' = (\bigcap_{i=1}^m \text{dom}(\mathcal{D}_i)) \cap T$. For any $T' \in T'$, define

\[
D(T') = \bigcup_{i=1}^m \left( \bigcup_{(T', D_i) \in \mathcal{D}_i} D_j \right).
\]

Suppose that for all $T' \in T'$, it holds that $T' \cup D(T') \in T$. If protocol $\Pi$ UC-realizes $W_{\text{AE}}^{\mathcal{D}}(F)$ in the $(F_1, \ldots, F_m)$-hybrid model against a $T$-adversary, then $C^{\mathcal{D}_1 \ldots \mathcal{D}_m}(\Pi)$ UC-realizes $W_{\text{AE}}^{\mathcal{D}'}(F)$ in the $(W_{\text{AE}}^{\mathcal{D}_1}(F_1), \ldots, W_{\text{AE}}^{\mathcal{D}_m}(F_m))$-hybrid model against a $T'$-adversary, where $\mathcal{D}'$ is defined as follows: For all $T' \in T'$, we have $(T', D \cup A) \in \mathcal{D}'$ if $D \subseteq D(T')$ and $(T' \cup D(T'), A) \in \mathcal{D}$. Moreover, $\mathcal{D}'$ is AE-monotone.
Universally Composable Almost-Everywhere Secure Computation

In the specific case that $\Pi$ realizes an unwrapped functionality $F$ (indeed, one can always apply our AE wrapper to $F$ 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 $G$-hybrid model against a threshold adversary, we obtain the following corollary:

**Corollary 17.** Let $\mathcal{D}$ be a $t'$-complete, $D$-monotone, and AE-monotone doom structure. Let $t = \max |T| = t' \left( \bigcup_{(T_i, D_i) \in \mathcal{G}} D_i \right) \cup T'$. If protocol $\Pi$ UC-realizes $F$ in the $G$-hybrid model against a $t$-adversary, then $\mathcal{D}(\Pi)$ UC-realizes $W_{\text{AE}}(F)$ in the $W_{\text{AE}}(G)$-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 $\mathcal{D}$ used to wrap $G$ to be preserved when wrapping $F$. By construction, all of our doom structures satisfy these two properties. We remark that $t$ has a natural interpretation: the maximum number of parties that can become unprivileged (with respect to $\mathcal{D}$) 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 graph with secure channels only between connected parties, using our AE compiler. To capture the MPC task for $n$ parties, it is clear that this holds even in the $3$-hybrid model, against an adversary corrupting less than $n$ parties. It is known [8, 18] that classical broadcast protocols are not adaptively secure in a simulation-based setting, and gave a VSS-based protocol that does in fact realize adaptively secure broadcast with perfect security for $t < \frac{n}{3}$, assuming only secure channels. Therefore, there exists a protocol that UC-realizes $F_{\text{MPC}}^{f,P,\text{rnd}}$ for any $n$-ary function $f$ and some $\text{rnd}$ in the $F_{\text{MPC}}^{P,1}$-hybrid model, against an adversary corrupting less than $\frac{n}{3}$ parties. It is clear that this holds even in the $F_{\text{MPC}}^{P,t}$-hybrid model, for arbitrary $t$. Now, by invoking Corollary 17 (which of course also offers statistical security) and then applying the (regular) UC composition theorem in tandem with our results in Theorems 14 and 15 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 18.** There exists a protocol that UC-realizes $W_{\text{AE}}(F_{\text{MPC}}^{f,P,\text{rnd}})$ in the $F_{\text{SC}}^{G_{\text{SC},\text{inv}}}$-hybrid model against a $t$-adversary, for some $\text{rnd}$ and $t \in O\left(\frac{n}{\log n}\right)$.

**Corollary 19.** Let $x \in \{\text{UPFAL}, \text{CGO}, \text{JRV}\}$. There exists a protocol statistically UC-realizing $W_{\text{AE}}(F_{\text{MPC}}^{f,P,\text{rnd}})$ in the $F_{\text{SC}}^{G_{\text{SC}}}$-hybrid model against a $t$-adversary, for some $\text{rnd}$ and $t \in O(n)$. 
References


Universally Composable Almost-Everywhere Secure Computation


A Functionalities and Protocols

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

1. The sender $S$ sends $n\tau + 1$ strong pads $SP_1, SP_2, \ldots, SP_{n\tau+1}$ to the receiver $R$ through wires $\gamma_1, \gamma_2, \ldots, \gamma_m$, respectively. Let $R_1', R_2', \ldots, R_n'$ be the strings received by $R$. $R$ rejects all wires where $|R_i'| \neq 15l$.

2. For $i \in [n]$, $S$ generates $R_i^*$ by replacing $12l$ randomly chosen positions of $R_i$ with “*.” Then $S$ sends $R_1^*, R_2^*, \ldots, R_n^*$ to $R$ over $Pub$.

3. For any $i \in [n]$, if $R_i^*$ and $R_i'$ differ in any “opened” bits, $R$ marks $\gamma_i$ as “faulty.” Then $R$ sends an $n$-bit string to $S$ over $Pub$ that identifies faulty wires. Let $\vec{\tau} = \{\tau_1, \tau_2, \ldots, \tau_m\}$, $s \leq n$ denote the set of non-faulty wires, and $\vec{R}_i$, $|\vec{R}_i| = 12l$, $1 \leq i \leq s$, denote the corresponding string of unopened bits; let $\vec{R}_i'$ 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 $S_i = E(m_i) \oplus \vec{R}_i$, $1 \leq i \leq s$, over $Pub$. $R$ computes $m'_i = D(S_i \oplus \vec{R}_i')$ for all $1 \leq i \leq s$. Then $R$ outputs $m' = m'_1 \oplus m'_2 \oplus \cdots \oplus m'_s$.

Figure 4 The SMT-PD protocol from [25].

Figure 3 The SMT protocol from [20].

Protocol $\Pi_{\text{SMT}}(\vec{\gamma}, \tau, m)$

1. (Phase 1) The sender $S$ sends $n\tau + 1$ strong pads $SP_1, SP_2, \ldots, SP_{n\tau+1}$ to the receiver $R$ choosing a random polynomial $f(x) \in \mathbb{Z}_q(x)$ of degree $\tau$ and sets $pad = f(0)$. Then for each $i \in [n]$ $S$ chooses an additional random polynomial $g_i(x) \in \mathbb{Z}_q$ of degree $\tau$ such that $g_i(0) = f(i)$. Finally, for each $i \in [n]$, $S$ sends $h_i(\cdot)$ with a vector of checking pieces $C_i = (c_{1i}, c_{2i}, \ldots, c_{ni})$ to $R$ using wire $\gamma_i$ where for all $i, j \in [n]$, $c_{ij} = h_j(i)$.

2. (Phase 2) For each $k \in [n]$, let $T_k$ be received in the attempted transmission of $SP_k$ and $g_k, D_k$ be possibly corrupted information received as $h_i, C_i$. If for any $T_k$ all the checking pieces $c_{ij}$ and all polynomials $h_i(\cdot)$ are consistent then $R$ interpolates the $pad_a$ from $T_a$ and sends “a, OK” to $S$ over the authenticated channel. Otherwise, $R$ finds an $l$ such that $\{|\text{conflicts of } T_l\} \subseteq \cup_{m \neq l}\{|\text{conflicts of } T_m\}$, where any unordered pair $(i, j)$ is called a conflict of $T_k$ if $d_{ji} \neq g_j(i)$. Then $R$ sends $l$ and all $T_m, m \neq l$ back to $S$ using authenticated channel.

3. (Phase 3) If “a, OK” received over the authenticated channel in phase 2, then $S$ sends $z = m + pad_a$ to $R$ using the authenticated channel. Otherwise, $S$ performs error detection on all $T_j$’s received from $R$ and sends detected faults and $z = m + pad_i$ to $R$ using authenticated channel.

- If $R$ previously sent “a, OK” to $S$ in phase 1, then $S$ computes $m = z - pad_a$.

- Otherwise, $R$ corrects the faults in $T_i$, obtains $pad_i$, and computes $m = z - pad_i$. 
**Functionality** $F_{wc}^{S,R,\vec{W}}$  

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 $\text{sid} = (P_s, P_r, \text{sid'})$, where $\{P_s, P_r\} = \{S, R\}$. Initialize variables $m_1, \ldots, m_n$ to $\perp$.

- Upon receiving input $(\text{SEND}, \text{sid}, W_i, v_i)$ from $P_s$ in round $\rho$ (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 $(\text{SENDLEAK}, \text{sid}, W_i, m_i)$ to the adversary; otherwise send $(\text{SENDLEAK}, \text{sid}, W_i, l(m_i))$.

- Upon receiving $(\text{INFLSEND}, \text{sid}, W_i, m_i')$ from the adversary: If any $P \in \{P_s, P_r, W_i\}$ is corrupted, and $(\text{SEND}, \text{sid}, W_i, m_i)$ has not yet been sent to $P_r$, then set $m_i \leftarrow m_i'$.

- Upon receiving $(\text{FETCH}, \text{sid}, W_i)$ from $P_r$ in round $\rho'$: If $P_r$ is corrupted, then send $(\text{FETCHLEAK}, \text{sid}, W_i)$ to the adversary; otherwise, if $\rho' = \rho + 2$, then output $(\text{SENT}, \text{sid}, W_i, m_i)$ to $P_r$ if it has not yet been sent.

- Upon receiving $(\text{OUTPUT}, \text{sid}, W_i)$ from the adversary: If $P_r$ is corrupted, then output $(\text{SENT}, \text{sid}, W_i, m_i)$ to $P_r$ if it has not yet been sent.

- Upon receiving $(\text{CORRUPT}, \text{sid}, P)$ 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 $(\text{SENDLEAK}, \text{sid}, m_i)$ to the adversary; otherwise, send $(\text{SENDLEAK}, \text{sid}, m_1, \ldots, m_n)$. If $P = P_r$, then additionally leak any previous fetch requests made by $P_r$.

**Figure 5** The Wire Channel functionality.

### B Proofs

**Proof of Theorem 5.** Let $A$ be an adversary in the real world. We construct a simulator $S$ in the ideal world, such that no environment can distinguish whether it is interacting with $\Pi_{\text{sm}}(S, R, \vec{W})$ and $A$, or with $F_{\text{sm}}^{(S,R), \text{rnd}}$ and $S$. The simulator internally runs a copy of $A$, and plays the roles of $F_{\text{auth}}^{(S,R), 2}$, $F_{wc}^{S,R,\vec{W}}$, and the parties in a simulated execution of the protocol. All inputs from $\mathcal{Z}$ are forwarded to $A$, and all outputs from $A$ are forwarded to $\mathcal{Z}$. Moreover, whenever $A$ corrupts a party in the simulation, $S$ corrupts the same party in the ideal world by interacting with $F_{\text{sm}}^{(S,R), \text{rnd}}$ (except if the party is a wire-party), and if the corruption was direct (i.e., not via one of the aiding functionalities), then $S$ sends $A$ the party’s state and follows $A$’s instructions thereafter for that party.

The simulated execution starts upon $S$ receiving $(\text{SENDLEAK}, \text{sid}, \hat{m})$ from $F_{\text{sm}}^{(S,R), \text{rnd}}$ in round $\rho$ for $\text{sid} = (P_s, P_r, \text{sid'})$, where $\hat{m} \in \{m, l(m)\}$ and $m$ is the message to be sent. Now, $S$ executes the first two phases of the protocol honestly, by simulating sending random strong pads (shares $h_i(\cdot)$ and checking pieces $\vec{C}_i = (c_{i1}, \ldots, c_{in})$) from $P_s$ to $P_r$ through the $n$ wire-parties (i.e., by simulating leakage from $F_{wc}^{S,R,\vec{W}}$ to $A$, and responding to corruption and influence requests directed from $A$ to $F_{wc}^{S,R,\vec{W}}$) and by simulating sending the response from $P_r$ to $P_s$ over the authenticated channel (i.e., by appropriately playing the role of $F_{\text{auth}}^{(S,R), 2}$ for $A$). For the third phase of the protocol, $S$ simulates honestly, except for choosing $z$ when $P_s$ and $P_r$ are both honest in which case $S$ simulates sending a random value $z$ from $P_s$ to $P_r$ over $F_{\text{auth}}^{(S,R), 2}$ instead of $z = m \oplus \text{Pad}$. When $P_s$ or $P_r$ is corrupted by $A$, $S$ learns $m$ via leakage from $F_{\text{sm}}^{(S,R), \text{rnd}}$ and thus can send $z = m \oplus \text{Pad}$ just like in the real protocol. Note that the simulated $P_s$ may need to abort, and that if the simulated $P_r$ aborts by outputting $\perp$, then $S$ can influence $F_{\text{sm}}^{(S,R), \text{rnd}}$, since this can only happen if $A$ corrupts $P_s$ or $P_r$. 

Functionality $F_{SC}^G$. 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 $m \leftarrow 1$.

- Upon receiving input $(\text{SEND}, sid, v)$ from $P_i$ in round $\rho$, 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 $(\text{SEND}, sid, m)$ has not yet been sent to $P_j$, then set $m \leftarrow m'$.
- Upon receiving $(\text{FETCH}, sid)$ from $P_j$ in round $\rho + 1$, output $(\text{SEND}, sid, m)$ 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 6 The Secure Channel functionality for (incomplete) graph $G_n$.

Next, we describe how $S$ simulates $P_r$’s response to a FETCH input from $Z$ in the real world. If $P_r$ is corrupted by $A$, then $S$ can wait to receive $(\text{FETCHLEAK}, sid)$ from $F_{\text{SMT}}^{(S,R),md}$, upon which it possibly leaks the fetch to $A$ and then sends INFLSEND and OUTPUT messages to $F_{\text{SMT}}^{(S,R),md}$ as appropriate. If $P_r$ is corrupted by $A$, then $S$ needs to constantly influence $F_{\text{SMT}}^{(S,R),md}$ during the second phase of the protocol, so that the dummy $P_r$ fetches the correct value. If neither $P_r$ nor $P_r$ is corrupted, then $S$ can simply let the dummy $P$ fetch from $F_{\text{SMT}}^{(S,R),md}$ when instructed by $Z$. An important case is when both $P_r$ and $P_r$ are honest in the beginning of the third phase (at the time $S$ decides the value of $z$) and then at least one of them gets corrupted before the protocol ends (before the output is fetched). In this case, $A$ receives enough leakage from $F_{\text{WC}}^{(S,R),\overline{W}}$ to interpolate the pad and compute the value of the message from $z$. Since $z$ is chosen randomly by $S$, the message learned by $A$ deviates from what is sent by $P_r$, causing $Z$ to distinguish the real and ideal worlds. In such a situation, $S$ learns the actual value of $m$ via leakage from $F_{\text{SMT}}^{(S,R),md}$, and hence it can cheat by calculating a fake pad' satisfying $z = m \oplus \text{pad'}$ and then simulate leaking from $F_{\text{WC}}^{(S,R),\overline{W}}$ to result in pad'.

The simulation is perfect. Indeed, by corrupting at most $\tau$ wires, $A$ learns nothing about $h_i(0)$ for honest wires, because the $h_i(\cdot)$’s are independent random polynomials of degree $\tau$. Moreover, $f(\cdot)$ is also a random polynomial of degree $\tau$, so $A$ learns nothing about $f(0)$ (i.e., pad used by the protocol looks uniformly random to $Z$). Thus, choosing a random value $z$ by $S$ looks perfectly indistinguishable from the real protocol execution to $Z$. Besides, $F_{\text{AUTH}}^{(S,R),2}$ acts like an authenticated channel in the protocol, and hence in the real world $P_r$ outputs the sender’s input. (See the full version [16] for more details).

Proof of Theorem 7 (Sketch). We construct a simulator $S$ that is very similar to the simulator in the proof of Theorem 5. However, $S$ now interacts with a wrapped functionality, and corruption messages for wire-parties are indeed sent because they can now be processed by the wrapper. Another difference concerns the case in which $P_s$ and $P_r$ are not corrupted by $A$. If $A$ corrupts only a minority of the wire-parties, then $S$ 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. Otherwise, as soon as enough wire-parties are corrupted, $S$ sends a DOOM message for $P_s$ to the wrapper, which will be accepted by definition of $F_{\text{SMT}}$, and obtains $m$ as leakage. Now, $S$ can use $z = m \oplus \text{pad}$ 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 $A$ might exceed a minority of wire-party corruptions only after $S$ has already chosen a random $z$. However, $S$ can handle this by cheating and computing a fake pad consistent with $m$, like the simulator in the proof of Theorem 5 does. Finally, $S$ may need to simulate sender or receiver aborts when $A$ corrupts a majority of wire-parties but not $P_s$ or $P_r$.

**Proof of Theorem 8.** Let $A$ be an adversary in the real world. We construct a simulator $S$ in the ideal world, such that no environment $Z$ can distinguish whether it is interacting with $\Pi_{\text{SMT-PR}}(S, R, \vec{W}, l)$ and $A$, or with $F_{\text{SMT}}^{(S, R)}$ and $S$. The simulator internally runs a copy of $A$, and plays the roles of $F_{\text{WC}}^{S, R, \vec{W}}$, $F_{\text{AUTH}}$, and the parties in a simulated execution of the protocol. All inputs from $Z$ are forwarded to $A$, and all outputs from $A$ are forwarded to $Z$. Moreover, whenever $A$ corrupts a party in the simulation, $S$ corrupts the same party in the ideal world by interacting with $F_{\text{SIT}}^{(S, R)}$ (except if the party is a wire-party), and if the corruption was direct (i.e., not via either of the aiding functionalities), then $S$ sends $A$ the party’s state and thereafter follows $A$’s instructions for that party.

The simulated execution starts upon $S$ receiving $(\text{SENDLEAK}, \text{sid}, \hat{m})$ from $F_{\text{SIT}}^{(S, R)}$ in round $p$ for $\text{sid} = (P_s, P_r, \text{sid'})$, where $\hat{m} \in \{m, l(m)\}$ and $m$ is the message to be sent. Now, $S$ simulates the first three phases of the protocol honestly, by simulating sending random bitstrings from $P_s$ to $P_r$ through the $n$ wire-parties (i.e., by simulating leakage from $F_{\text{WC}}^{S, R, \vec{W}}$ to $A$, and responding to corruption and influence requests directed from $A$ to $F_{\text{WC}}^{S, R, \vec{W}}$) 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 $F_{\text{AUTH}}^{(S, R)}$ for $A$). In the fourth phase, $S$ 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 $A$, then $S$ learns $m$ via leakage from $F_{\text{SIT}}^{(S, R)}$.

Next, we describe how $S$ simulates $P_i$’s response to a Fetch input from $Z$ in the real world. If $P_i$ is corrupted by $A$, then $S$ can wait to receive $(\text{FETCHLEAK}, \text{sid})$ from $F_{\text{SIT}}^{(S, R)}$, upon which it possibly leaks the fetch to $A$ and then sends InFLSend and OUTPUT messages to $F_{\text{SIT}}^{(S, R)}$ as appropriate. Otherwise, if $P_i$ is corrupted by $A$, then $S$ needs to constantly influence $F_{\text{SIT}}^{(S, R)}$ so that the dummy $P_i$ fetches the correct value. Finally, if neither $P_s$ nor $P_r$ is corrupted, then $S$ simply lets the dummy $P_r$ fetch from $F_{\text{SIT}}^{(S, R)}$ when instructed by $Z$. In this case, the real-world $P_i$ outputs $m$ except with the error probability.

An important issue is that when $P_s$ or $P_r$ is corrupted only after $S$ has already decided on the random $m_i$’s to be encoded in the fourth phase, $A$ may be able to recover some $m'$ from its view of the bitstrings sent in the first phase, but $m'$ may not equal $m$ and this could allow $Z$ to distinguish between the real and ideal worlds. However, $S$ can handle this case by faking what was sent in the first phase. In particular, at least one bitstring (corresponding to an uncorrupted wire-party) sent in the first phase is not visible to $A$, so $S$ can redefine it to be consistent with $m$ (which $S$ learns from leakage from $F_{\text{SIT}}^{(S, R)}$). The simulation is not perfect as there is an error probability, but $Z$ still cannot distinguish between the two worlds. In particular, when $P_s$ and $P_r$ are not corrupted by $A$, the assumption that at most all but one of the wire-parties are corrupted implies that the random bitstring sent on at least one wire in the first phase will mask the value of $m$ from $A$.

**Proof of Theorem 16.** We first prove that $\mathcal{D}'$ is AE-monotone. Suppose that $(T_i, D_i) \in \mathcal{D}'$ 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 $(T_i \cup D(T_i), A) \in \mathcal{D}$. We want to show that $(T_j, D_j) \in \mathcal{D}'$, and it suffices to show that $D \subseteq D(T_j)$ and $(T_j \cup D(T_j), A) \in \mathcal{D}$. Since $D(T_i) \subseteq D(T_j)$ (using the fact that $\mathcal{D}_1, \ldots, \mathcal{D}_m$ are all AE-monotone), it follows that $D \subseteq D(T_j)$. On the other hand, since $T_j \cup D(T_j) \subseteq T_j \cup D(T_j)$, it follows that $(T_j \cup D(T_j), A) \in \mathcal{D}$ (using the fact that $\mathcal{D}$ is AE-monotone and that $T_j \cup D(T_j) \in \mathcal{T}$). We now prove the security of the compiled protocol.
Let $S$ be a simulator (guaranteed to exist by the security of $\Pi$) such that no environment $Z$ can distinguish whether it is interacting with $\Pi$ and the dummy adversary $D$, or with $W_{\Pi}^{Z}(F)$ and $S$. We use $S$ to construct a simulator $S'$ such that no environment $Z'$ can distinguish whether it is interacting with $C^{Z_1,\ldots,Z_m}(\Pi)$ and $D$, or with $W_{\Pi}^{Z'}(F)$ and $S'$.

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

Similarly, outputs from $S$ are forwarded to $Z'$, with some additional processing. Assuming that $\Pi$ uses instances of $F_1, \ldots, F_m$ 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 $F_i$ to the dummy adversary for $\Pi$; thus, the processing done by $S'$ is that reported messages from an instance of $F_i$ are replaced by reported messages from an instance of $W_{\Pi}^{Z'}(F_i)$. Finally, $S'$ plays the role of $W_{\Pi}^{Z'}(F)$ by simply forwarding messages from $W_{\Pi}^{Z'}(F)$ to $S$ as if coming from $W_{\Pi}^{Z'}(F)$, and forwarding messages directed to $W_{\Pi}^{Z'}(F)$ (from $S$) to $W_{\Pi}^{Z'}(F)$, except that CORRUPT messages for doomed parties (i.e., parties that $Z'$ did not request to corrupt) are replaced by DOOM messages.

Suppose for a contradiction that there is an environment $Z'$ such that $\text{IDEAL}_{W_{\Pi}^{Z'}(F),S',Z} \not\equiv \text{EXEC}_{C^{Z_1,\ldots,Z_m}(\Pi),D,Z'}$. Then, we construct an environment $Z$ such that $\text{IDEAL}_{W_{\Pi}^{Z'}(F),S,Z} \not\equiv \text{EXEC}_{\Pi,D,Z}$. The environment $Z$ will simulate an interaction between $Z'$ and $D$, and output whatever $Z'$ outputs, as well as do some additional processing. Whenever $Z'$ instructs its dummy adversary to deliver a message to an instance of $W_{\Pi}^{Z'}(F_i)$, this is translated by $Z$ into a delivery request for a corresponding instance of $F_i$ and forwarded to the external adversary (either $S$ or $D$), except that a request to deliver a DOOM message is converted into a request to deliver a CORRUPT message if allowed by $\mathcal{D}_i$ and dropped otherwise. Corruption requests directed to parties are forwarded to the external adversary unmodified.

Next, whenever $Z$ receives subroutine output from the external adversary, this is forwarded to $Z'$, except that reported messages from instances of $W_{\Pi}^{Z'}(F)$ are translated into reported messages from corresponding instances of $F_i$. Finally, $Z$ simply relays inputs and outputs between $Z'$ and parties. We conclude by claiming that $\text{IDEAL}_{W_{\Pi}^{Z'}(F),S',Z} \equiv \text{IDEAL}_{W_{\Pi}^{Z'}(F),S,Z}$ and $\text{EXEC}_{C^{Z_1,\ldots,Z_m}(\Pi),D,Z'} \equiv \text{EXEC}_{\Pi,D,Z}$. Indeed, if $Z$ interacts with $W_{\Pi}^{Z'}(F)$ and $S$, then the view of the simulated $Z'$ within $Z$ is identical to the view of $Z'$ when interacting with $W_{\Pi}^{Z'}(F)$ and $S'$, and similarly if $Z$ interacts with $\Pi$ and $D$, then the view of the simulated $Z'$ within $Z$ is identical to the view of $Z'$ when interacting with $C^{Z_1,\ldots,Z_m}(\Pi)$ and $D$. □