Infrastructures for Secure Multiparty Computation

by

Srinivasan Raghuraman

B.Tech. (Honors), Indian Institute of Technology Madras (2015)
S.M., Massachusetts Institute of Technology (2017)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2020

Author .....
Department of Electrical Engineering and Computer Science
May 15, 2020

Certified by .....
Shafrira Goldwasser
RSA Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Infrastructures for Secure Multiparty Computation

by

Srinivasan Raghuraman

Submitted to the Department of Electrical Engineering and Computer Science
on May 15, 2020, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science and Engineering

Abstract

We study the problem of implementing an infrastructure for secure multiparty computation (MPC). The goal of our infrastructure is to enable reliable communication, secure computation and fair computation in a network. The desiderata for an infrastructure include reusability, transferability and fault-tolerance. It is not hard to see that the above criteria are fulfilled for infrastructures that we use in daily life, for e.g., the infrastructure for online communication (e-mail, instant messaging, etc.) consisting of transatlantic undersea cables, routers, wireless access points, etc. We consider which cryptographic primitives would be good building blocks for a secure computation infrastructure.

The first, reliable communication. We study the problem of almost-everywhere reliable message transmission. The goal is to design low-degree networks which allow a large fraction of honest nodes to communicate reliably even while linearly many nodes can experience byzantine corruptions and deviate arbitrarily from the assigned protocol. We consider both the worst-case and randomized corruption scheduling models. In the worst-case model, we achieve a log-degree network with a polylogarithmic work complexity protocol improving over the state-of-the-art results that required a polylogarithmic-degree network and had a linear work complexity. In the randomized model, we improve upon the state of the art protocols, both in work-efficiency and in resilience.

Next, we propose an infrastructure for secure computation, which would consist of OT channels between some pairs of parties in the network. We devise information-theoretically secure protocols that allow additional pairs of parties to establish secure OT correlations using the help of other parties in the network in the presence of a dishonest majority. Our main technical contribution is an upper bound that matches known lower bounds regarding the number of OT channels necessary and sufficient for MPC. In particular, we characterize which n-party OT graphs G allow t-secure computation of OT correlations between all pairs of parties, showing that this is possible if and only if the complement of G does not contain the complete bipartite graph $K_{n-t,n-t}$ as a subgraph.

Finally, we study the problem of building an infrastructure for fair secure computation, where we guarantee that if any party receives the output of the secure computation, then all honest parties do as well. Toward this goal, we introduce a new 2-party primitive $\mathcal{F}_{\mathsf{SyX}}$ ("synchronizable fair exchange") and show that it is complete for realizing any n-party functionality with fairness in a setting where all n parties are pairwise connected by independent instances of $\mathcal{F}_{\mathsf{SyX}}$. Additionally, a pair of parties may reuse a single instance of $\mathcal{F}_{\mathsf{SyX}}$ in any number of multiparty protocols (possibly involving different sets of parties).

Thesis Supervisor: Shafrira Goldwasser
Title: RSA Professor of Electrical Engineering and Computer Science

Acknowledgments

It is a challenging task to thank everyone involved in the completion of this work, but I will make an honest attempt to do so. First and foremost, I thank my thesis supervisor, **Prof. Shafi Goldwasser** for her unending guidance and support. However crazy the problem or the idea I may have, Shafi always lent an open ear, an open and resourceful mind and a supporting hand. Aside from all those tremendously helpful interactions, I thank you from the bottom of my heart for being my mentor for matters both professional and otherwise. She was constantly on the lookout for opportunities that would be exciting to me. Even if we weren't in the same place physically for a portion of my program, she always kept a watchful eye making completing this work that much easier. I feel grateful to have been advised by her.

I would like to thank the other members of my thesis committee, **Prof. Mike Sipser** and **Prof. Vinod Vaikuntanathan**. I have been a teaching assistant to Mike for the past three years and I must say that working with has been no short of an absolute pleasure. Impeccably organized, dedicated, caring and nurturing, Mike is truly a role model instructor. Were I to return to MIT any day, I would gladly teach 18.404 again. I thank Vinod for useful discussions through the course of this work. Practical, well-informed and extremely quick, Vinod taught me a lot about taking a step back and always looking at the bigger picture.

I would like to thank my collaborators for this work, **Siddhartha Jayanti**, **Ranjit Kumaresan**, **Adam Sealfon** and **Nikhil Vyas**. I recall the countless meetings we have had both face-to-face and cross-country. These fine colleagues of mine at the time taught me how to pick problems, work on them, write solutions, and so much more. It would be an understatement to say that solving these problems was a lot of fun.

I must mention and thank **Janet Fischer** for being a terrific graduate administrator. Whatever the issue, whether it was visa applications, internships or simply getting done with program requirements, Janet was always there to help.

And now comes the long list of friends and family. I would like to thank all the members of the Theory Group at CSAIL. In particular, I would like to thank **Deborah Goodwin** and **Patrice Macaluso**, the great admins who always have the right answers to my questions, for all their support. I extend a warm hug to all the members of the **MIT Concert Choir**, **MIT Chamber Chorus**, **MIT MTA Events Office**, **MIT HSC**, **Ramakrishna Vedanta Society in Boston**, **Bill Cutter**, **Kerry Deal**, **Karen Harvey**, **Eileen Huang**, **Yukiko Oba**, **Clera Ryu** and **Swami Tyagananda** who have a been a huge part of my life outside of academics for some or all of the past three years.

A special shoutout to all my roommates over the years and to all my **my truly phenomenal friends** the world-over, for being there, for their care and support through some of the rough patches in the past three years, for everything they've taught me and for all the amazing experiences we have shared together. There are too many of them, and hence, I will refrain from naming names in order to avoid committing the crime of leaving someone out. While that may seem like a cop-out, I am sure we know of my gratitude to you and gratitude for you.

I would like to thank **my family** for their unconditional love and affection, their support and their confidence in me.

Finally, I would like to thank the **Lord Almighty**, my constant source of faith even when there seems to be no other.

Original Manuscripts

The results presented in this thesis also appear in joint works with:

1. "Efficient constructions for almost-everywhere secure computation." *Siddhartha Jayanti and Nikhil Vyas [87].*

2. "Network oblivious transfer." *Ranjit Kumaresan and Adam Sealfon [96].*

3. "Synchronizable fair exchange." *Ranjit Kumaresan and Adam Sealfon [97].*

"The best and most beautiful things in the world cannot be seen or even touched—they must be felt with the heart." — Helen Keller

मयात्मने लोकमस्ति सर्वम्मयास्ति सर्वञ्चिदस्मि सर्वम्।
तदस्ति सत्यन्तदेव नित्यन्तदस्म्यहन्न्वै न शेषदृष्ट्यम्॥
परोऽपि नास्तीति सत्यमेतद्विधाय मुक्तिञ्जिगाति लोकात्।
स्वलोकबन्धात्थस्वदेहपीडात्थस्वचित्तशोकात्थस्वदृष्ट्यभावात्॥

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Many real world applications involve computing functions on large data sets that are distributed across machines in a global network. In many such applications, the data held by any particular agent may need to be kept private. For instance, hospitals across the world have confidential patient data that can be used to create accurate disease models and improve treatment plans.

Protocols for secure multiparty computation [126, 68, 17, 37] allow a set of mutually distrusting parties to carry out a distributed computation without compromising the privacy of inputs or the correctness of the end result. The problem of constructing protocols for secure multiparty computation gained much attention in the '80s. Yao [126] came out with the first two-party protocol using oblivious transfer, quickly followed by Goldreich, Micali and Wigderson [68] who came out with an n-party protocol for all $n \geq 2$, also using oblivious transfer. Ben-Or, Goldwasser and Wigderson [17] went on to show that in the presence of an honest majority (assuming broadcast), one could construct information-theoretically secure protocols. Yet another question that emerged was that of fairness. In fair secure computation, not only are we concerned with the privacy of inputs or the correctness of the end result, but also the delivery of output. We would like that if any party receives the output of a fair secure computation, then so do all the parties. This isn't true of the protocols of [126] or [68], and in general, such protocols are impossible to construct, as was demonstrated by the impossibility result of Cleve [41] with regards to fair coin-tossing.

As a research area, secure computation has witnessed several breakthroughs in the last decade [127, 94, 84, 113, 104, 103, 81, 110, 82, 106]. However, despite a wide array of potential game-changing applications, there is nearly no practical adoption of secure computation today (with the notable exceptions of [24, 25]). Computations wrapped in a secure computation protocol do not yet deliver results efficiently enough to be acceptable in many cloud-computing applications. For instance, state-of-the-art semi-honest 2-party protocols incur a factor ≈ 100 slowdown even for simple computations. The ubiquitous need for such distributed private computations has motivated research on efficient multiparty computation (MPC) [125, 68, 17, 37].

In the absence of practical real-world protocols for secure computation which are secure in the presence of any number of dishonest parties, there is a need for relaxations that are meaningful and yet provide significant performance benefits. As an example, classic protocols for secure computation [17, 37, 119] (with subsequent improvements e.g., [43, 19, 12, 47, 45, 44]) offer vastly better efficiency at the cost of tolerating only a small constant fraction of adversaries. The resilience offered is certainly acceptable when the number of participating parties is large, e.g., the setting of *large-scale* secure computation [27, 49, 128, 28]. Although large-scale secure computation is well-suited for several interesting applications (such as voting, census, surveys), we posit that typical settings involve computations over data supplied by a few end users. In such cases, the overhead associated with interaction among a large number of *helper parties* is likely to render these protocols more expensive than a standard secure computation protocol among the end users. If the number of helper parties is small, security against a small fraction of corrupt parties may be a very weak guarantee, since a handful of corrupt parties could render the protocol insecure.

Thus, in the presence of a larger fraction of corrupted parties, several avenues for theoretical and practical research remain open:

- Is it possible to get faster protocols, either through better assumptions, primitives, or special hardware?

- Is it possible to reduce the amount of communication (number of bits) in the

protocol [91, 49, 28, 34]?

- Is it possible to reduce the number of rounds in the protocol [11]?
- Is it possible to alleviate the need for synchronicity of nodes in the network?
- Is it possible to minimize the use of physical resources?

Indeed, each of these questions have been considered in the past and remain hotbeds of fruitful research.

An orthogonal approach for reducing the online cost of secure computation protocols is the use of *preprocessing* [9, 48, 20, 3]. This approach can dramatically reduce the cost of secure computation: for instance, given preprocessing [9], the ≈ 100 factor slowdown for simple computations no longer applies. Recent theoretical research has shown that many primitives can even be made *reusable* (e.g. [71]). Perhaps the most important drawback of this approach (other than the fact that the preprocessing phase is typically very expensive) is that the preprocessing is not *transferable*. Clearly, a pair of parties that want to perform a secure computation cannot benefit from this approach without performing the expensive preprocessing step. Moreover, this seems to hold even if each of the two parties have set up the preprocessing with multiple others. Typically, the cost of the preprocessing phase is quite high, presenting a barrier for the practical use of preprocessed protocols. This is especially true in settings where parties are unlikely to run many secure computations that would amortize the cost of preprocessing.

Motivated by the discussion above, we conclude that some directions that seem to offer efficiency benefits for secure computation are (1) highly resilient protocols that use only a small number of helper parties, and (2) a preprocessing procedure that allows a notion of transferability between users. Taken together, these two ideas have the potential to provide an *infrastructure* for efficient secure computation. Some sets of parties might run a preprocessing phase among themselves. These parties can then act as helper parties and "transfer" their preprocessing to help users who want to run a secure computation protocol. We informally describe some desiderata for such an infrastructure.

- *Reusability/Amortization.* Setting up an infrastructure component could be expensive, but using it and maintaining it should be inexpensive relative to setting up a new component.

- *Transferability/Routing.* It should be possible to combine different components of the infrastructure to deliver benefits to the end users.

- *Robustness/Fault-tolerance.* Failure or unavailability of some components of the infrastructure should not nullify the usefulness of the infrastructure.

It is not hard to see that the above criteria are fulfilled for infrastructures that we use in daily life, for e.g., the infrastructure for online communication (e-mail, instant messaging, etc.) consisting of transatlantic undersea cables, routers, wireless access points, etc. What cryptographic primitives would be good candidates for a *secure computation infrastructure*? In answering this question, it is instructive to discover the components of the infrastructure.

1.1 Infrastructure for Reliable Communication

MPC protocols for various important tasks, such as elections, were discovered in the twentieth century, but most of these protocols have not seen practical application as they were designed for densely connected networks. For MPC to see widespread use, it is important for protocols to rely only on the sparse connectivity that is available in modern large scale networks while simultaneously meeting the efficiency needs of practice. All secure distributed protocols rely on the ability of machines to communicate. In particular, if A and B are two nodes in a network, A must be able to send a message to B in a way that satisfies the following two properties: (1) *reliable transmission*: B receives the message that A intended to send, and (2) *authentication*: B must be able to confirm that A was indeed the sender of the received message [8]. The first—reliable transmission—is the focus of our work. Reliable transmission becomes trivial if we assume every pair of nodes has a dedicated *secure link* to pass messages over. However, it is impractical to create pairwise secure links in modern large scale networks—a network on even just a thousand nodes would need half a million secure links!

In a seminal work, Dwork, Peleg, Pippenger and Upfal [54] considered the question of designing *sparse* networks that are tolerant to nodes experiencing *byzantine* failures - nodes that fail can deviate arbitrarily from the protocol. The problem is to design a network G of degree d on n nodes in which *honest nodes* can continue to communicate and execute protocols, even after t nodes are *corrupted*, i.e., experience byzantine failures. The challenge is to make the degree d as small as possible (ideally constant), even while allowing up to $t = \varepsilon n$ corruptions for some constant ε. Since we allow many more corruptions, t, than the degree of the graph, d, any set of $\Omega(t/d)$ honest nodes can be isolated from the other nodes if all of their neighbors are corrupted. Thereby, it is impossible for all the honest nodes to communicate with each other in this failure model. So, [54] allow x honest nodes to become *doomed*, and only require that a set of $n - t - x$ honest nodes be able to pairwise communicate with each other after t corruptions occur. This set of honest nodes are called *privileged nodes*, and the

class of primitives that work on these privileged nodes in the presence of byzantine failures are called *almost-everywhere (AE)* primitives. Our main result is the design of a new sparse graph and a corresponding communication protocol that improve the state of the art in AE reliable message transmission.

Our protocol for AE reliable message transmission immediately implies an improved protocol for AE Secure Multi-party Computation (MPC) in the following way. The problem of byzantine agreement [115, 100] is one where nodes start with an initial value but wish to agree, at the end of execution of some protocol, on some value, despite malicious or byzantine behavior of some subset of nodes. Prior to [54], this problem was considered assuming all pairs of nodes had a secure link for communication [115, 100, 51]. [54] introduced the notion of *almost-everywhere agreement* where only privileged nodes need to reach agreement. We note that *AE reliable message transmission*, which would guarantee that a large subset of the network can transmit messages to each other reliably, implies a protocol for *AE agreement*, and an AE agreement protocol implies a protocol for *AE secure MPC* that is unconditionally or information-theoretically secure as formulated in the work of Garay and Ostrovsky [64].

Previous Work on Reliable Communication Protocols

AE reliable transmission protocols are generally compared by the following three properties:

1. *degree*: the degree, d, of graph of secure links needed for the protocol.

2. *resilience*: a protocol is $(f(n), g(t))$-resilient if it can sustain up to $t = f(n)$ corruptions while dooming at most $x = g(t)$ nodes when t nodes are corrupted.

3. *work complexity*: the total amount of work (both local and message passing) required for a single communication from node u to node v in the network.

The ideal solution would give a protocol on a constant degree graph that is $(\varepsilon n, O(t))$-resilient for a small constant $\varepsilon \in (0, 1)$, and have low work complexity.

This ideal remains an open problem. In the remainder of this section, we discuss the three previous results which are mutually incomparable, and thereby, jointly form the state-of-the-art for the AE reliable transmission problem. We remark that ε continues to be used in resilience guarantees throughout the work, and it simply represents some constant in $(0, 1)$ when it appears.

[54] introduced the AE reliable transmission problem, and gave the first solution to the problem. Their famous *Butterfly network* is a constant degree graph and their protocol is $(\varepsilon n/\log n, O(t))$-resilient, and has linear work complexity. While the Butterfly network is a simple network and [54]'s protocol, the possibility of increasing the resilience of the network to be resistant to a linear number of corruptions spurred further research into the AE reliable transmission.

Upfal [122] showed the remarkable result that both optimal graph degree and optimal resilience were simultaneously possible. He produced a constant degree graph and a protocol that is $(\varepsilon n, O(t))$-resilient on that graph. While these advantages make [122] of great information theoretic importance, his protocol is practically intractable, since it requires nodes to do an exponential amount of computation. In particular, when a node u is sending a message to a node v, [122]'s algorithm requires v to loop through all possible subsets of corrupted nodes before it can correctly decipher the message it has received (even when u and v are both privileged). Thus, the work complexity of [122]'s algorithm is the exponential $O\left(\binom{n}{t}\right)$.

The third work at the frontier of the field was that of Chandran, Garay and Ostrovsky [35]. This work tries to combine the work efficiency of [54]'s protocol with the resiliency of [122]'s work. [35] succeed in getting a linear work protocol, and in fact achieve the very strong property of $(\varepsilon n, O(t/\log n))$-resilience. However, the significant weakness of their work is the complexity and degree of their graph. Unlike the other two works, their protocol is designed for a graph of polylogarithmic-degree.

In summary, the state-of-the-art on the AE reliable transmission problem consisted of three incomparable results: [54] linear work protocol with low resiliency on a constant degree graph, [122]'s exponential work protocol with high resiliency on a constant degree graph, and [35]'s linear work protocol with high resiliency on a

polylogarithmic degree graph.

1.1.1 Our Contributions

The results presented here also appear in a joint work with Siddhartha Jayanti and Nikhil Vyas [87]. **The primary contribution of our work** is an AE reliable transmission protocol on a graph of logarithmic degree that is $(\varepsilon n, O(t/\log n))$-resilient while requiring only polylogarithmic work per communication. The significance of our result lies in the low degree of the graph and the work-efficiency of the protocol. Our result is a strict improvement over [35]'s result, as our graph's degree is smaller - only logarithmic, compared to polylogarithmic—and our protocol's work complexity is polylogarithmic as opposed to linear, while our protocol's resiliency is the same as their protocol's. Also, our protocol is the first AE reliable transmission protocol to achieve sublinear work complexity. In particular, the small work complexity of our message-passing protocol enables us to simulate any protocol on a (dense) complete graph with only polylogarithmic multiplicative overhead on our nearly-sparse logarithmic degree graph, while all previous protocols required at least linear multiplicative overhead. The primary result of our work is stated as a theorem below.

Theorem (**Main Theorem for Communication Infrastructure: Worst-case Corruptions**). *For sufficiently large n, there exists an n-node network $G_{wc} = (V, E)$, a protocol $\Pi_{wc,eff}$ for message transmission on it, and constants α and β, such that:*

1. *The network G_{wc} is of degree $O(\log n)$.*
2. *The Work complexity of $\Pi_{wc,eff}$ is $O(\text{polylog}(n))$.*
3. *$\Pi_{wc,eff}$ is $(\alpha n, \beta t/\log n)$-resilient.*

We compare our work to previous works in Table 1.1.1.

A secondary contribution of our work is an improvement over the state of the art in AE reliable transmission when the adversary corrupts nodes at *random*. Ben-or and Ron [18] introduced the *random corruption model* in which nodes are corrupted

Result	Degree	Corruptions	Doomed	Total Work
Dwork et. al. [54]	$O(1)$	$\varepsilon n/\log n$	$O(t)$	$O(n)$
Upfal [122]	$O(1)$	εn	$O(t)$	$O\left(\binom{n}{t}\right)$
Chandran et. al. [35]	$\text{polylog}(n)$	εn	$O(t/\log n)$	$O(n)$
This work	$O(\log n)$	εn	$O(t/\log n)$	$\text{polylog}(n)$

Table 1.1: In general, total work can be further broken down into message passing work and internal computations of the nodes. For [122]'s protocol the message passing work is linear, and internal computations take exponential work. For the rest of the protocols, message passing work and internal computation work are identical.

independently and at random and the protocol only needs to be resilient with some large probability, called the *probability of resiliency*. So, algorithms in this model are evaluated by *four* parameters: degree, resiliency, work complexity, and probability of resiliency. (If the probability of resiliency becomes equal to one, then the protocol is resilient in the standard model.) [18] exhibited a constant degree network that is $(\varepsilon n, O(t))$-resilient with high probability, and thereby almost resolved the random corruption model. En route to our main construction, we produce a different constant degree network that has the same $(\varepsilon n, O(t))$-resilience, just with even higher probability than [18]. The result of our work is stated as a theorem below.

Theorem (Main Theorem for Communication Infrastructure: Random Corruptions). *For sufficiently large n, there exists an n-node network $G_{rand} = (V, E)$, a protocol Π_{rand} for message transmission on it, and constants α_3 and β_3, such that:*

1. *The network G_{rand} is of constant degree.*

2. *The Work complexity of $\Pi_{wc,eff}$ is $O(\text{polylog}(n))$.*

3. *If a subset of nodes $T \subset V$ is randomly corrupt, where $|T| \leq \alpha_3 n$, with probability $1 - (t/n)^{\alpha_2 t/(4\log(n))}$, there exists a set of nodes $S \subset V$ where $|S| \geq n - \beta_3|T|$ such that every pair of nodes in S can communicate reliably with each other by invoking Π_{rand}.*

Arriving at the main theorems

At a high level, our AE transmission protocol is constructed in two parts.

1. *Achieving high resilience and low degree:* The first part yields a network and protocol that have the $(\varepsilon n, O(t/\log n))$-resilience and logarithmic-degree; this immediately yields an improvement over [35]'s protocol, which has the same resiliency but polylogarithmic degree. Our construction in the first part, uses the protocol of [54] on the Butterfly Network as a black box.

2. *Achieving work-efficiency:* In the second part, we improve this communication protocol significantly—reducing the linear work to only polylogarithmic work, while maintaining the resiliency parameters. Modularly replacing the protocol from [54] with the new efficient protocol immediately yields our main theorem: a logarithmic degree graph and a polylogarithmic work protocol with $(\varepsilon n, O(t/\log n))$-resilience.

Step 1: Achieving high resilience and low degree. We achieve a highly resilient graph with low degree in two steps.

(a) *Achieving high resilience against random corruptions:* In the first step, we combine the ideas of [54] and [122] to construct a constant degree graph that is resilient to a linear number of corruptions with high probability in the random corruption model.

(b) *Reducing worst-case corruptions to random corruptions:* In the second step, we strengthen the graph from the first step by adding multiple (perturbed) copies of the edges to it and modify the protocol to get a graph that is resilient to linearly many worst-case corruptions.

Step 2: Achieving work-efficiency. Our protocols for network resiliency used the G_{But} and [54]'s protocol designed for the graph as primitives. In this part of the work we design a communication protocol on the Butterfly Network that is more

work-efficient than [54]'s protocol. A communication from node u to node v in [54]'s protocol floods many paths between u and v in G_{But} with the message and makes v take the majority of the messages it receives to decipher the true message reliably. In this step of our work, we show that if paths are chosen correctly, it suffices to use only polylogarithmically many paths per pair of nodes.

Putting it all together: Substituting [54]'s protocol in the highly resilient network from Step 1 with the more efficient protocol on the Butterfly graph from Step 2 yields the main results of our work.

1.1.2 Other Related work

There have been a plethora of works asking for various different measures of quality of an agreement or MPC protocol. A sequence of works seek to improve the round complexity of protocols for byzantine consensus [22, 23]. Another goal is to optimize the communication complexity of byzantine agreement protocols [52, 93, 92, 90]. Another model of corruptions is that of edge corruptions [36]. As observed in [35], an almost-everywhere secure computation protocol for node corruptions can be readily transformed into a corresponding almost-everywhere protocol also tolerating edge corruptions, for a reduced fraction of edge corruptions (by a factor of d, the degree of the network). We note that all our results hence also extend to the edge corruption model, both worst-case and random.

1.2 Infrastructure for Secure Computation

The results thus far provide various insights into establishing a sparse yet functional authenticated communication network in the presence of various kinds of adversaries. While this is by no means a complete characterization, it provides a basis for setting up communication and authentication links in our network. Following this stage, we turn to another primitive, namely *oblivious transfer* [118, 55]. Oblivious transfer (OT) is a two-party primitive that involves a sender and a receiver. The sender holds two bits x_0, x_1 while the receiver holds a single bit b. Using OT, the sender can send x_b to the receiver without learning b while simultaneously hiding x_{1-b}. We know that OT is necessary and sufficient for performing secure computations in the presence of a dishonest majority assuming an authenticated communication network.

OT is a fundamental building block of secure computation [89, 86]. As discussed in [86], a few benefits of basing secure computation on OT include:

- *Preprocessing.* OT enables precomputation in an offline stage before the inputs or the function to be computed are known. The subsequent online phase is extremely efficient [9].

- *Amortization.* The cost of computing OTs can be accelerated using efficient OT extension techniques [10, 84, 86, 113].

- *Security.* OTs can be realized under a wide variety of computational assumptions [116, 55, 118, 111, 39] or under physical assumptions.

Infrastructure as an OT graph. In this work, we consider n parties connected by a synchronous network with secure point-to-point private communication channels between every pair of parties. In addition, some pairs of parties on the network have established *OT channels* between them providing them with the ability to perform arbitrarily many OT operations. We represent the OT channel network via an *OT graph* G. The vertices of G represent the n parties, and pairs of parties that have an established OT channel are connected by an edge in G. Since OT can be reversed

unconditionally [123], we make no distinction between the sender and the receiver in an OT channel. This OT graph represents the infrastructure we begin with. The OT channels could either represent poly(λ) 1-out-of-2 OT correlations for a computational security parameter λ, or a physical channel (e.g., noisy channel) that realizes, say δ-Rabin OT [118].[1] We are interested in obtaining security against adaptive semihonest adversaries. We also discuss security against adaptive malicious adversaries under computational assumptions.

Two parties that are connected by an edge can use the corresponding existing OT channel to run a secure computation protocol between themselves. What about parties that are not connected by an edge? Clearly, they can establish an OT channel between themselves via an OT protocol [116, 39] or perhaps using a physical channel. The latter option, if possible, is likely to be expensive and the costs of setting up a physical channel may be infeasible unless the two parties are likely to execute many secure computation protocols. The former option is also expensive as it involves use of public-key cryptography which is somewhat necessary in the light of [83].[2] This motivates the question of whether additional parties can use an existing OT infrastructure to establish an OT channel between themselves unconditionally or relying only on the existence of symmetric-key cryptography. A positive result to this question would show that expensive cryptographic operations are not required to set up additional OT channels. In this work we construct OT protocols with information-theoretic security against a threshold adversary.

The generality of an OT infrastructure. Consider the following candidate for an infrastructure. Suppose there is a channel between a pair of parties that allows them to securely evaluate any function. Since OT is complete for secure computation, one can apply the results of [86, 89] to use the OT channel to implement a secure evaluation channel. In the other direction, one can use a secure evaluation channel

[1] Recall that λ 1-out-of-2 OT correlations can be extended to poly(λ) 1-out-of-2 OT correlations via OT extension using just symmetric-key cryptography (e.g. one-way functions [10] or correlation-robust hash functions [84]).

[2] As a rule of thumb, use of public-key cryptography is computationally around 4-6 orders of magnitude more expensive than using symmetric-key cryptography [16].

to trivially implement OT channels. Consequently, such a channel is equivalent to an OT channel. The same argument extends to channels that implement any 2-party primitive that is complete for secure computation [108, 14]. Furthermore, the above argument also applies to the setting where a *set* of parties have a secure evaluation channel. Such a channel is equivalent to an OT graph where parties in the set have pairwise OT channels with everyone in the set.

Assuming a full network of secure channels. We have already discussed emulating secure communication channels using a relatively sparse infrastructure of secure channels. We note that that the secure channels between two parties in the sparse infrastructure can be implemented either via non-interactive key exchange and hybrid encryption or via a physical assumption. We emphasize that the one-time setup cost of emulating a secure channel (e.g. via Diffie-Hellman key exchange) is much lower than the one-time setup cost of emulating an OT channel that allows unbounded OT calls via an OT protocol even using OT extension. Furthermore, our assumption of secure channels is identical to the setting of [89, 70, 86], who show that secure computation reduces to OT under information-theoretic reductions.

1.2.1 Our Contributions

The results presented here also appear in a joint work with Ranjit Kumaresan and Adam Sealfon [96]. We introduce our main result for OT protocols with information-theoretic security against a threshold adversary.

Theorem (**Main Theorem for OT Infrastructure**). *Let $G = (V, E)$ be an OT graph on n parties $P_1, \ldots, P_n$, so that any pair of parties P_i, P_j which are connected by an edge may make an unbounded number of calls to an OT oracle. Let $\mathbb{A}$ be the class of semi-honest t-threshold adversaries which may adaptively corrupt at most t parties.*[3] *Then two parties A and B in $\{P_1, \ldots, P_n\}$ can information-theoretically emulate an OT oracle while being secure against all adversaries $\mathcal{A} \in \mathbb{A}$ if and only if*

[3] Combining our work with results from [76, 69], we can also obtain computational security against malicious adversaries in both the nonadaptive and adaptive settings.

1. *(honest majority) it holds that $t < n/2$; or*

2. *(trivial) A and B are connected by an edge in G; or*

3. *(partition) there exists no partition V_1, V_2, V_3 of G such that all of the following conditions are satisfied: (a) $|V_1| = |V_2| = n - t$ and $|V_3| = 2t - n$; (b) $A \in V_1$ and $B \in V_2$; and (c) for every $A' \in V_1$ and $B' \in V_2$ it holds that $(A', B') \notin E$.*

Our main theorem gives a complete characterization of networks for which a pair of parties can utilize the OT network infrastructure to execute a secure computation protocol. The first two conditions in our theorem are straightforward: (1) if $t < n/2$, then we are in the honest majority regime, and thus it is possible to implement secure computation (or emulate an OT oracle) using the honest majority information-theoretically secure protocols of [119]; (2) clearly if A and B are connected by an OT edge then by definition they can emulate an OT oracle.

Condition (3) applies when $t \geq n/2$ and when A and B do not have an OT edge between them. This condition is effectively the converse of the impossibility result of [77], which states that any n-party OT graph whose complement contains $K_{n-t,n-t}$ as a subgraph cannot allow a n-party secure computation that tolerates t semi-honest corruptions. Condition (3) implies that any n-party OT graph whose complement does not contain $K_{n-t,n-t}$ as a subgraph can run n-party secure computations tolerating t semi-honest corruptions.

Applying our main theorem

We first compare our positive results to those of [77]. They investigate how to construct an OT graph with the minimum number of edges allowing n parties to execute a secure computation protocol. They show a construction for a graph with $(n+o(n))\binom{\lceil 1/\delta \rceil}{2}$ edges which they prove is sufficient for resilience against an adversary that corrupts $(1-\delta)n$ parties. Our result provides a complete, simple characterization of which OT graphs on n vertices are sufficient to run a t-secure protocol generating OT correlations between all pairs of vertices for any $t \geq n/2$, which is sufficient to

obtain a protocol for secure computation among the n parties [89, 86]. Our main theorem also implies that determining the minimum number of OT edges needed to execute a secure computation protocol for general $n, t \geq n/2$ is equivalent to an open problem in graph theory posed by Zarankiewicz in 1951 [95].

Our results immediately imply that for some values of t, extremely simple sparse OT graphs suffice for achieving secure multiparty computation. For n even and $t = n/2$, we have that the t-claw graph (cf. Fig. 4-4(a)) has t edges and suffices to achieve t-secure multiparty computation. For n odd and $t = (n+1)/2$, the $(t+1)$-cycle has $t+1$ edges and suffices to achieve t-secure multiparty computation. We show in Chapter 4.9 that these examples are the sparsest possible graphs which can achieve $\lfloor (n+1)/2 \rfloor$-secure multiparty computation.

Next, our results are also well-suited to make use of an OT infrastructure for secure computation. Specifically, let G_I denote the OT graph consisting of existing OT edges between parties that are part of the infrastructure. Now suppose a pair of parties A, B not connected by an OT edge wish to execute a secure computation protocol. Then they can find a subgraph G of G_I with $A, B \in G$ and $|G| = n$ such that they agree that at most t out of the n parties can be corrupt and the partition condition in our main theorem holds for G. Since it is possible to handle a dishonest majority, parties do not have to settle for a lower threshold and can enjoy increased confidence in the security of their protocol by making use of the infrastructure. Surprisingly, it turns out the OT subgraph G need not even contain t OT edges to offer resilience against t corruptions (cf. Fig. 4-2(c) with $n = 4, t = 2$).

A pair of parties may use the OT correlations generated as the base OTs for an OT extension protocol and inexpensively generate many OT correlations that can be saved for future use or to add to the OT infrastructure. In any case, it should be clear that our protocols readily allow load-balancing across the OT infrastructure and are also abort-tolerant in the sense that if some subgraph G ends up not delivering the output, then one can readily use a different subgraph G'. Thus we believe that our results can be used to build a *scalable* infrastructure for secure computation that allows (1) amortization, (2) routing, and (3) is robust.

An important caveat regarding efficiency

In the special cases $t = n/2 + \mathcal{O}(1)$ and $t = n - \mathcal{O}(1)$, determining whether a graph satisfies the partition condition requires at most poly(n) time. However, in general the problem is coNP-complete, since it can be restated in the graph complement as subgraph isomorphism of a complete bipartite graph [66]. Our protocols are efficient in n only for $t = n/2 + \mathcal{O}(1)$ and $t = n - \mathcal{O}(1)$.[4] In particular, our protocol is quite efficient for small values of n, a setting in which computing OT correlations in the presence of a dishonest majority may be especially useful in practice.

1.2.2 Other Related work

As mentioned previously, there is a large body of work on secure computation in the offline/online model (cf. [105, 101, 48, 20, 113] and references therein). These protocols exhibit an extremely fast online phase at the expense of a slow preprocessing phase (sometimes using MPC [105] or more typically, OT correlations [113] or a somewhat homomorphic encryption scheme [48]). To the best of our knowledge, the question of *transferability* of preprocessing has not been explicitly investigated in the literature with the notable exception of [77], which we will discuss in greater detail below. There is a large body of work on secure computation against a threshold adversary (e.g. [17, 37, 119, 68]). Popular regimes where secure computation against threshold adversaries have been investigated are for $t < n/3$, $t < n/2$, or $t = n - 1$. In this work we are interested in threshold adversaries for a dishonest majority, that is, adversaries which can corrupt t out of n parties for $n/2 \leq t < n$.[5] Such regimes were investigated in other contexts such as authenticated broadcast [61] and fairness in secure computation [15, 80, 85]. Infrastructures for *perfectly secure message transmission* (PSMT) were investigated in the seminal work of [50] (see also [56] and references therein). While the task of PSMT is similar to our question regarding OT channels, there are inherent differences. For example, our protocols can implement

[4] For $t = n/2 + \mathcal{O}(1)$, we achieve efficiency using computationally-secure OT extension (e.g. [10, 84]). Our protocol with information-theoretic security is quasi-polynomial-time for $t = n/2 + \mathcal{O}(1)$. We do, however, achieve information-theoretic security in polynomial time for $t = n - \mathcal{O}(1)$.

[5] When $t < n/2$, there is no need to rely on an OT infrastructure [119].

OT even between two parties that are isolated in the OT graph (i.e., not connected to any other party via an OT channel).[6] In PSMT, on the other hand, there is no hope of achieving secure communication with a node that is not connected by any secure channel.

Most relevant to our results is the work of Harnik, Ishai, and Kushilevitz [77]. The main question in their work is an investigation of the number of OT channels sufficient to implement a n-party secure computation protocol. In a nutshell, they show against an adaptive t-threshold adversary for $t = (1-\delta)n$, an explicit construction of an OT graph consisting of $(n + o(n))\binom{\lceil 1/\delta \rceil}{2}$ OT channels that suffices to implement secure computation among the n parties. They note further that against a static adversary, $\binom{\lceil s/\delta \rceil}{2}$ OT channels suffice, where s denotes a statistical security parameter. On the negative side, they show that a complete OT graph is necessary for secure computation when dealing with an adversary that can corrupt $t = n - 1$ parties. They derive this result by showing that in a 3-party OT graph with two OT channels, it is not possible to obtain OT correlations between the third pair of parties with security against two corruptions. Moreover they generalize their 3-party negative result to any OT graph whose complement contains the complete bipartite graph $K_{n-t,n-t}$ as a subgraph. In this work we extend the results of [77], fully characterizing the networks for which it is possible to obtain OT correlations between a designated pair of parties.

[6] Recall that the model considered in this work, we assume a *full* network of secure private communication channels.

1.3 Infrastructure for Fair Computation

Finally, we turn to fair computation. Fairness means that either all parties get the output of the secure computation or none do. Definitions of secure computation do vary across models, in part owing to the general impossibility results for fair coin-tossing [41]. In settings where the majority of the participating parties are dishonest (including the two party setting), a protocol for secure computation only provides *security-with-abort*, and in particular is not required to guarantee important properties such as guaranteed output delivery (all parties get the output) or even fairness. On the other hand, when up to $t < n/3$ parties are corrupt, then there exist protocols for n-party secure computation that guarantee output delivery [17, 37]. This result can be extended to a setting where up to $t < n/2$ parties are corrupt assuming the existence of a broadcast channel [68, 119].

Given the state of affairs, there has been extensive research to better understand the problem of fairness and guaranteed output delivery in secure computation in setting where $t \geq n/2$. For instance, while Cleve [41] showed that two-party fair coin tossing is impossible, the works of Gordon et al. [72, 74, 4, 5] showed the existence of non-trivial functions for which fair secure computation is possible in the dishonest majority setting. On the other hand, *partially fair* secure computation [75, 13] provides a solution for a relaxed notion of fairness in secure computation where fairness with respect to any honest party may be violated (the honest party does not receive its output although some other party does) but only with some parameterizable (inverse polynomial) probability.

Fitzi, Garay, Maurer, and Ostrovsky [57] studied complete primitives for secure computation *with guaranteed output delivery*. We recall The cardinality of a primitive is the number of parties that participate in an invocation of the primitive. For instance, OT is a primitive of cardinality 2. They showed that no primitive of cardinality $n-1$ is complete for n-party secure computation. More generally, for $n \geq 3$ and $k < n$, they show that no primitive of cardinality k is complete when $t \geq \lceil \frac{k-1}{k+1} \cdot n \rceil$. It follows that when $t \geq \lceil n/3 \rceil$, no primitive of cardinality 2 is complete for secure

computation. Also, when $t \geq n-2$, no primitive of cardinality $k < n$ is complete for secure computation. They also show a primitive of cardinality n that is complete for n-party secure computation when $t \geq n-2$.

It is interesting to note that the above impossibility results are derived in [57] by showing the *impossibility of broadcast* (or Byzantine agreement) given a primitive of cardinality k. In this context, note that Cohen and Lindell [42] showed that the presence of a broadcast channel is inconsequential to achieving the goal of fairness, i.e., they showed that any protocol for fair computation that uses a broadcast channel can be compiled into one that does not use a broadcast channel. They also showed that assuming the existence of a broadcast channel, any protocol for fair secure computation can be compiled into one that provides guaranteed output delivery. Importantly, all these transformations only require primitives of cardinality 2.

1.3.1 Our Contributions

The results presented here also appear in a joint work with Ranjit Kumaresan and Adam Sealfon [97]. The impossibility result of [57] essentially implies that one cannot hope to build an infrastructure for secure computation with guaranteed output delivery. We, however, restrict our attention to *fair secure computation* alone. In this work, we introduce a new 2-party primitive $\mathcal{F}_{\mathsf{SyX}}$ ("synchronizable fair exchange," or simply "synchronizable exchange") and show that it is complete for realizing any n-party functionality with *fairness* in a setting where all n parties are pairwise connected by independent instances of $\mathcal{F}_{\mathsf{SyX}}$. Additionally, a pair of parties may *reuse* a single instance of $\mathcal{F}_{\mathsf{SyX}}$ in any number of multiparty protocols, possibly involving different sets of parties.

Synchronizable exchange $\mathcal{F}_{\mathsf{SyX}}$ is a two-party symmetric primitive that is reactive (like the commitment functionality $\mathcal{F}_{\mathsf{com}}$ [31]) and works in two phases.

1. *Load phase:* In the first phase, which we call the *load phase*, parties submit their private inputs x_1, x_2 along with public inputs $(f_1, f_2, \phi_1, \phi_2)$. Here f_1, f_2 are 2-input 2-output functions, and ϕ_1, ϕ_2 are boolean predicates. The public input

must be submitted by both parties, and the submitted values must match. Upon receiving these inputs, $\mathcal{F}_{\mathsf{SyX}}$ computes $f_1(x_1, x_2)$ and delivers the respective outputs to both parties.

2. *Trigger phase:* Next, in the *trigger phase*, which can be initiated at any later time after the load phase, party P_i can send a "witness" w_i to $\mathcal{F}_{\mathsf{SyX}}$ following which $\mathcal{F}_{\mathsf{SyX}}$ checks if $\phi_i(w_i) = 1$. If that is indeed the case, then $\mathcal{F}_{\mathsf{SyX}}$ computes $f_2(x_1, x_2)$ and delivers the respective outputs along with w_i to both parties. We stress that $\mathcal{F}_{\mathsf{SyX}}$ guarantees that both parties get the output of f_2.

The result of our work is stated as a theorem below.

Theorem (**Main Theorem for Fair Computation Infrastructure**)**.** *Consider n parties $P_1, \ldots, P_n$ in the* point-to-point *model. Then, assuming the existence of one-way functions, there exists a protocol π which securely computes $\mathcal{F}_{\mathsf{MPC}}$ with fairness in the presence of t-threshold adversaries for any $0 \leq t < n$ in the $\mathcal{F}_{\mathsf{SyX}}$-hybrid model.*

To use multiple pairwise instances of synchronizable exchange to achieve n-wise fair secure computation, the main idea is to keep different instances of $\mathcal{F}_{\mathsf{SyX}}$ "in sync" with each other throughout the protocol execution. That is, we need to ensure that all pairwise $\mathcal{F}_{\mathsf{SyX}}$ instances are, loosely speaking, *simultaneously loaded*, and if so, *simultaneously triggered*. Ensuring this in the presence of byzantine adversaries is somewhat tricky, and we outline our techniques below.

Arriving at the main theorems

Reduction to fair reconstruction. First, we let parties run an (unfair) MPC protocol for a function f that accepts parties' inputs and computes the function output, then computes secret shares of the function output, and then computes commitments on these secret shares. Finally, the MPC outputs to all parties the set of all commitments computed above, and to each individual party the corresponding share of the function output. Since the MPC protocol itself does not guarantee fairness, it may be

that some honest party does not receive the output. In that case, all parties terminate and abort the protocol, and no party learns the function output. If the protocol has not terminated, then all that is left to perform a fair reconstruction of the function output from the secret shares. The above technique of reducing fair computation of a function to fair reconstruction of a (non-malleable) additive secret sharing scheme is a well-known technique [73].

Synchronization via trigger conditions. The commitments generated in the above step are used to define the trigger conditions, specifically the trigger witness must include (among other things) openings to the commitments (i.e., the secret shares). That is, each pair of parties initiate the load phase with their $\mathcal{F}_{\mathsf{SyX}}$ instance. We will need to ensure that the protocol proceeds only if all $\mathcal{F}_{\mathsf{SyX}}$ instances were loaded. To do this, we let the load phase of each $\mathcal{F}_{\mathsf{SyX}}$ instance to output a *receipt* (think of these as signatures on some special instance-specific message) that indicates that the $\mathcal{F}_{\mathsf{SyX}}$ instance has been loaded. Following this parties broadcast to all other parties the receipts they have obtained in the load phase. (Note that by [42], we can assume a broadcast channel while developing our protocol, and then use their compiler to remove the broadcast channel from our protocol.) In an honest execution, at the end of this broadcast step, each party would possess receipts from every pairwise $\mathcal{F}_{\mathsf{SyX}}$. On the other hand, corrupt parties may not broadcast some receipts, resulting in a setting where corrupt parties possess all receipts, but honest parties do not.

To maintain that $\mathcal{F}_{\mathsf{SyX}}$ instances remain in sync, we let the trigger conditions ask for all receipts (each individual $\mathcal{F}_{\mathsf{SyX}}$ instance can verify these load receipts using, e.g., digital signature verification). This way, we ensure that any $\mathcal{F}_{\mathsf{SyX}}$ instance can be triggered only if all $\mathcal{F}_{\mathsf{SyX}}$ instances were loaded. Recall that by definition, $\mathcal{F}_{\mathsf{SyX}}$ outputs the trigger witness along with the output of f_2. This in turn ensures that if, say an $\mathcal{F}_{\mathsf{SyX}}$ instance between P_i and P_j was triggered by P_i, then P_j would obtain the load receipts which it can then use as part of trigger witnesses for other $\mathcal{F}_{\mathsf{SyX}}$ instances associated with P_j. Finally, because parties only receive additive secret shares of the output, to get the final output the adversary will need to trigger at least

one $\mathcal{F}_{\mathsf{SyX}}$ instance associated with an honest party. The ideas outlined above ensures that that honest party (and consequently every honest party) will be able to continue triggering other $\mathcal{F}_{\mathsf{SyX}}$ instances associated with it, and obtain the final output. An additional detail to note is that in our constructions, we let the boolean predicates ϕ_1, ϕ_2 depend on *time*. This is required to ensure termination of our protocols (i.e., force a time limit on when the adversary must begin triggering the $\mathcal{F}_{\mathsf{SyX}}$ instances to obtain output). Therefore, in the terminology of [114], our functionality $\mathcal{F}_{\mathsf{SyX}}$ is clock-aware. The techniques we use to ensure termination may be reminiscent of techniques used in the design of broadcast protocols from point-to-point channels in the dishonest majority setting [53].

Complexity, preprocessing, assumptions, and implementation. The complexity of $\mathcal{F}_{\mathsf{SyX}}$ is the sum of the complexities of the functions f_1, f_2, and the predicates ϕ_1, ϕ_2. In our constructions, the complexity of each $\mathcal{F}_{\mathsf{SyX}}$ instance is $\mathcal{O}(n^2\lambda\ell_{\mathsf{out}})$ and is independent of the size of the function that is being computed.[7] With additional assumptions, specifically with a *non-interactive non-committing encryption* [112] (alternatively, a *programmable random oracle*), the use of $\mathcal{F}_{\mathsf{SyX}}$ can be preprocessed in a network-independent manner to support any number of executions.[8] That is, a pair of parties can preprocess an instance of $\mathcal{F}_{\mathsf{SyX}}$ by loading it once, and then re-using it across multiple independent (possibly concurrent) executions of secure computation involving different sets of parties. This type of preprocessing is reminiscent of OT preprocessing [86, 96]. Of course, to enable this type of preprocessing, we rely on a variant of $\mathcal{F}_{\mathsf{SyX}}$ which can be *triggered multiple times* (but loaded only once). In this case, the complexity of f_1 is $\mathcal{O}(\lambda)$, while the complexity of f_2 is $\mathcal{O}(\lambda)$ per trigger invocation, and the complexities of ϕ_1, ϕ_2 would be $\mathcal{O}(n^2\lambda)$ per trigger invocation for a protocol involving n parties. We emphasize that in the preprocessing setting, $\mathcal{F}_{\mathsf{SyX}}$ need not be triggered when the protocol participants behave honestly. Using ideas similar to [40, 121], a follow-up work [citation redacted for anonymity] shows,

[7] The dependence on ℓ_{out} can be removed in the programmable random oracle model.

[8] Preprocessing for a bounded number of executions may be achieved by assuming only *receiver non-committing encryption* [32].

among other things, how to implement a single $\mathcal{F}_{\mathsf{SyX}}$ instance using trusted execution environments (e.g., Intel SGX) and a bulletin board abstraction. Note that different $\mathcal{F}_{\mathsf{SyX}}$ instances can use different bulletin boards, and still be usable by our protocols.

1.3.2 Other Related work

[57] investigate a number of interesting primitives that are complete for secure computation with guaranteed output delivery for various parameter regimes. (See [57] for a discussion of complete primitives for secure computation with abort.) For $t < n/3$, they identify *secure channels* (with cardinality 2) as a complete primitive. For $t < n/2$, they identify two complete primitives *converge cast* and *oblivious cast*. Both these have cardinality 3. For $t < n$, they identify *universal black box* (UBB) as a complete primitive of cardinality n. Improving on this, [73] show *fair reconstruction of a non-malleable secret sharing scheme* as a complete primitive of cardinality n, whose *complexity* is independent of the function being computed (this was not the case for UBB). In addition, [73] investigate the power of primitives that guarantee fairness but are restricted in other ways (i.e., inapplicable to fairly computing arbitrary functions). For instance, they study *fair coin flipping* and *simultaneous broadcast*, and show that neither of them are complete for fair computation. Note that simultaneous broadcast was shown in [88] to be complete for partial fairness [75]. Continuing, [73] show that (1) no primitive of size $\mathcal{O}(\log \lambda)$ is complete for fair computation (where λ is the security parameter), and (2) for every "short" m (when the adversary can run in time poly(m)), no m-bit primitive can be used to construct even a $m+1$-bit simultaneous broadcast. Note that none of the primitives discussed in [57, 73, 88] are *reactive*.

Timed commitments [26] (and numerous related works such as [60, 65]) can be used to enable a fair exchange of digital signatures, fair auctions, and more under a somewhat non-standard security notion. Other works with similar security notions that consider fairness in secure computation include [117, 63, 114] (see also numerous references therein). Another line of research investigates the use of physical/hardware assumptions to enforce fairness. For example, [102] relies on physical envelopes which

provide some form of synchronizability. Recent work [40, 121] (following [114]) has shown that fair secure computation is possible assuming the existence of trusted execution environments (alternatively, witness encryption [40]) and a blockchain to which all parties have read/write access. In these works, the blockchain can be interpreted as a component that helps in synchronizing the TEEs. We note that the above works use blockchain and envelopes as a cardinality n primitive in their constructions. There are numerous works in the optimistic model (cf. [6, 7] and several follow-up works) that minimize the use of a trusted third party to restore fairness when breached. Another line of research [1, 2, 21] investigates a non-standard notion of fair secure computation, called *secure computation with penalties* [21] where participants who do not obtain output are instead compensated monetarily (via cryptocurrency). [21] identifies a cardinality 2 primitive called *claim-or-refund* which is complete for this notion. (The presentation in [21] is strictly speaking not cardinality 2, but follow-up works clarify this.) That said, claim-or-refund shares many features with synchronizable exchange in that (1) both primitives operate in two phases (in the context of claim-or-refund, these phases are deposit and claim/refund), (2) the second phase is triggered via witnesses, (3) are clock-aware, and (4) both primitives can be preprocessed. Note that the claim-or-refund primitive locks up monetary funds which can be claimed by a designated receiver within a certain time by providing a witness to the trigger conditions, else the funds are refunded to the sender. Implicit in [1] is a primitive that releases monetary funds after a certain time has elapsed but during which time, the funds can be reversed by providing a trigger witness. This primitive is complete for fair coin tossing and fair lottery under the relaxed notion of secure computation with penalties [1].

1.4 Organization

In Chapter 2, we will introduce some preliminaries that are common to all the subsequent chapters. All results regarding communication in incomplete networks are discussed in Chapter 3. Chapter 4 contains all the results pertaining to oblivious transfer in an incomplete network and how one would go about building an OT infrastructure for MPC. Finally, Chapter 5 deals with fair computation and synchronizable fair exchange. We present our conclusions in Chapter 6.

The presentation of each chapter is self-contained. A summary of the results contained in the chapter is presented at the beginning. The definitions and preliminary material required for the chapter are presented right after. This is followed by a reminder of the formal theorems proved in the chapter. Following this, we present the technical details.

Chapter 2

Common Preliminaries

In this chapter, we will introduce some preliminaries that are common to Chapters 3, 4 and 5. Each of the chapters will also have a preliminaries section that will briefly recall the relevant concepts from this chapter, as well as some preliminaries that are specific to that chapter itself.

2.1 Notation and definitions

- For $n \in \mathbb{N}$, let $[n] = \{1, 2, \ldots, n\}$.
- We assume that all logarithms are taken to the base 2.
- Let $\lambda \in \mathbb{N}$ denote the security parameter.
- Symbols in with an arrow over them such as $\overrightarrow{a}$ denote vectors. By a_i we denote the i-th element of the vector $\overrightarrow{a}$. For a vector $\overrightarrow{a}$ of length $n \in \mathbb{N}$ and an index set $I \subseteq [n]$, we denote by $\overrightarrow{a}|_I$ the vector consisting of (ordered) elements from the set $\{a_i\}_{i \in I}$.
- By poly$(\cdot)$, we denote any function which is bounded by a polynomial in its argument.
- An algorithm $\mathcal{T}$ is said to be PPT if it is modeled as a probabilistic Turing machine that runs in time polynomial in λ.

- Informally, we say that a function is negligible, denoted by negl, if it vanishes faster than the inverse of any polynomial.
- If S is a set, then $x \xleftarrow{\$} S$ indicates the process of selecting x uniformly at random over S (which in particular assumes that S can be sampled efficiently). Similarly, $x \xleftarrow{\$} \mathcal{A}(\cdot)$ denotes the random variable that is the output of a randomized algorithm $\mathcal{A}$.
- Let $\mathcal{X}, \mathcal{Y}$ be two probability distributions over some set S. Their *statistical distance* is

$$\mathbf{SD}(\mathcal{X}, \mathcal{Y}) \stackrel{\text{def}}{=} \max_{T \subseteq S} \{|\Pr[\mathcal{X} \in T] - \Pr[\mathcal{Y} \in T]|\}$$

  We say that $\mathcal{X}$ and $\mathcal{Y}$ are ϵ-close if $\mathbf{SD}(\mathcal{X}, \mathcal{Y}) \leq \epsilon$ and this is denoted by $\mathcal{X} \approx_\epsilon \mathcal{Y}$. We say that $\mathcal{X}$ and $\mathcal{Y}$ are identical if $\mathbf{SD}(\mathcal{X}, \mathcal{Y}) = 0$ and this is denoted by $\mathcal{X} \equiv \mathcal{Y}$.

2.2 Approximation and Concentration Inequalities

Chernoff bound. Let X be a random variable with $\mathbb{E}[X] = \mu$. For $0 \leq \delta \leq 1$,

$$\Pr[X \geq (1+\delta)\mu] \leq e^{-\frac{\delta^2 \mu}{3}} \tag{2.1}$$

Stirling's approximation. For any $n, t \in \mathbb{N}$ with $t \leq n$,

$$\binom{n}{t} \leq \left(\frac{en}{t}\right)^t$$

2.3 Secure Computation

We recall most of the definitions regarding secure computation from [72] and [42]. We present them here for the sake of completeness and self-containedness. Consider

the scenario of n parties $P_1, \ldots, P_n$ with private inputs $x_1, \ldots, x_n \in \mathcal{X}$[1]. We denote $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{X}^n$.

2.3.1 Functionalities

A functionality f is a randomized process that maps n-tuples of inputs to n-tuples of outputs, that is, $f : \mathcal{X}^n \to \mathcal{Y}^n$[2]. We write $f = (f^1, \ldots, f^n)$ if we wish to emphasize the n outputs of f, but stress that if $f^1, \ldots, f^n$ are randomized, then the outputs of $f^1, \ldots, f^n$ are correlated random variables.

2.3.2 Adversaries

We consider security against several classes of adversaries. We define them below:

- Based on the *number of parties corrupted*
  - *t-threshold Adversaries:* The adversary can corrupt a set of at most t parties, where $0 \leq t < n$.
- Based on *how parties are corrupted*
  - *Worst-case Adversary:* The subset of corrupt nodes can be chosen adversarially after the network topology and protocols have been fixed.
  - *Random Adversary:* The subset of corrupt nodes must be chosen uniformly at random from the set of n nodes.
- Based on *when parties are corrupted*
  - *Static Adversary:* The adversary must corrupt nodes before the protocol is begun.

[1]Here we have assumed that the domains of the inputs of all parties is $\mathcal{X}$ for simplicity of notation. This can be easily adapted to consider setting where the domains are different.

[2]Here we have assumed that the domains of the outputs of all parties is $\mathcal{Y}$ for simplicity of notation. This can be easily adapted to consider setting where the domains are different.

- *Adaptive Adversary:* The adversary can corrupt nodes before the protocol is begun, during the execution of the protocol or even after the protocol is completed.

- Based on *how corrupted parties behave*

  - *Honest-but-Curious, or, Semi-honest Adversary:* The corrupted nodes must follow the protocol, however, they may try to learn additional information about the inputs of the honest parties while doing so.
  - *Malicious Adversary:* The corrupted nodes can deviate arbitrarily from the protocol.

2.3.3 Network Model

We define here some of the network configurations that we will be considering throughout this thesis.

- point-to-point *model:* All parties are connected via a fully connected point-to-point network of communication channels. The communication lines between parties are assumed to be ideally authenticated and private (and thus an adversary cannot read or modify messages sent between two honest parties). Furthermore, the delivery of messages between honest parties is guaranteed.

- broadcast *model:* All parties are given access to a physical broadcast channel (defined in Section 2.7)[3] in addition to the point-to-point network.

- OT-network *model:* All parties are connected via a fully pairwise connected network of oblivious transfer channels (defined in Section 2.6)[4] in addition to a fully connected point-to-point network.

- OT-broadcast *model:* All parties are given access to a physical broadcast channel in addition to the complete pairwise oblivious transfer network and a fully

[3]This can also be viewed as working in the $\mathcal{F}_{\mathsf{bc}}$-hybrid model. See Section 2.4.
[4]This can also be viewed as working in the $\mathcal{F}_{\mathsf{OT}}$-hybrid model. See Section 2.4.

connected point-to-point network[5].

2.3.4 Protocol

An n-party protocol for computing a functionality f is a protocol running in polynomial time and satisfying the following functional requirement: if for every $i \in [n]$, party P_i begins with private input $x_i \in \mathcal{X}$, then the joint distribution of the outputs of the parties is statistically close to $(f^1(\overrightarrow{x}), \ldots, f^n(\overrightarrow{x}))$. We assume that the protocol is executed in a synchronous network, that is, the execution proceeds in rounds: each round consists of a *send phase* (where parties send their message for this round) followed by a *receive* phase (where they receive messages from other parties). The adversary, being malicious, is also *rushing* which means that it can see the messages the honest parties send in a round, before determining the messages that the corrupted parties send in that round.

2.3.5 Security

The security of a protocol is analyzed by comparing what an adversary can do in a real protocol execution to what it can do in an ideal scenario that is secure by definition. This is formalized by considering an *ideal* computation involving an incorruptible *trusted party* to whom the parties send their inputs. The trusted party computes the functionality on the inputs and returns to each party its respective output. Loosely speaking, a protocol is secure if any adversary interacting in the real protocol (where no trusted party exists) can do no more harm than if it were involved in the above-described ideal computation.

Security with Guaranteed Output Delivery

The security of a protocol is analyzed by comparing what an adversary can do in a real protocol execution to what it can do in an ideal scenario that is secure by definition. This is formalized by considering an *ideal* computation involving an incorruptible

[5]This can also be viewed as working in the $(\mathcal{F}_{\mathsf{bc}}, \mathcal{F}_{\mathsf{OT}})$-hybrid model. See Section 2.4.

trusted party to whom the parties send their inputs. The trusted party computes the functionality on the inputs and returns to each party its respective output. Loosely speaking, a protocol is secure if any adversary interacting in the real protocol (where no trusted party exists) can do no more harm than if it were involved in the above-described ideal computation.

Execution in the ideal model. The parties are $P_1, \ldots, P_n$, and there is an adversary $\mathcal{A}$ who has corrupted at most t parties, where $0 \leq t < n$. Denote by $\mathcal{I} \subseteq [n]$ the set of indices of the parties corrupted by $\mathcal{A}$. An ideal execution for the computation of f proceeds as follows:

- **Inputs:** $P_1, \ldots, P_n$ hold their private inputs $x_1, \ldots, x_n \in \mathcal{X}$; the adversary $\mathcal{A}$ receives an auxiliary input z.
- **Send inputs to trusted party:** The honest parties send their inputs to the trusted party. The corrupted parties controlled by $\mathcal{A}$ may send any values of their choice. Denote the inputs sent to the trusted party by $x'_1, \ldots, x'_n$.
- **Trusted party sends outputs:** If $x'_i \notin \mathcal{X}$ for any $i \in [n]$, the trusted party sets x'_i to some default input in $\mathcal{X}$. Then, the trusted party chooses r uniformly at random and sends $f^i(x'_1, \ldots, x'_n; r)$ to party P_i for every $i \in [n]$.
- **Outputs:** The honest parties output whatever was sent by the trusted party. The corrupted parties output nothing and $\mathcal{A}$ outputs an arbitrary (probabilistic polynomial-time computable) function of its view.

We let $\textsc{Ideal}^{\mathsf{g.d.}}_{f,\mathcal{I},\mathcal{S}(z)}(\overrightarrow{x}, \lambda)$ be the random variable consisting of the output of the adversary and the output of the honest parties following an execution in the ideal model described above.

Execution in the real model. We next consider the real model in which an n-party protocol π is executed by $P_1, \ldots, P_n$ (and there is no trusted party). In this case, the adversary $\mathcal{A}$ gets the inputs of the corrupted party and sends all messages

on behalf of these parties, using an arbitrary polynomial-time strategy. The honest parties follow the instructions of π.

Let f be as above and let π be an n-party protocol computing f. Let $\mathcal{A}$ be a non-uniform probabilistic polynomial-time machine with auxiliary input z. We let $\text{REAL}_{\pi,\mathcal{I},\mathcal{A}(z)}(x_1,\ldots,x_n,\lambda)$ be the random variable consisting of the view of the adversary and the output of the honest parties following an execution of π where P_i begins by holding x_i for every $i \in [n]$.

Security as emulation of an ideal execution in the real model. Having defined the ideal and real models, we can now define security of a protocol. Loosely speaking, the definition asserts that a secure protocol (in the real model) emulates the ideal model (in which a trusted party exists). This is formulated as follows.

Definition 1. *Protocol π is said to securely compute f with guaranteed output delivery if for every non-uniform probabilistic polynomial-time adversary $\mathcal{A}$ in the real model, there exists a non-uniform probabilistic polynomial-time adversary $\mathcal{S}$ in the ideal model such that for every $\mathcal{I} \subseteq [n]$ with $|\mathcal{I}| \leq t$,*

$$\left\{\text{IDEAL}^{\text{g.d.}}_{f,\mathcal{I},\mathcal{S}(z)}(\overrightarrow{x},\lambda)\right\}_{\overrightarrow{x}\in\mathcal{X}^n,z\in\{0,1\}^*} \equiv \left\{\text{REAL}_{\pi,\mathcal{I},\mathcal{A}(z)}(\overrightarrow{x},\lambda)\right\}_{\overrightarrow{x}\in\mathcal{X}^n,z\in\{0,1\}^*}$$

We will sometimes relax security to statistical or computational definitions. A protocol is statistically secure if the random variables $\text{IDEAL}^{\text{g.d.}}_{f,\mathcal{I},\mathcal{S}(z)}(\overrightarrow{x},\lambda)$ and $\text{REAL}_{\pi,\mathcal{I},\mathcal{A}(z)}(\overrightarrow{x},\lambda)$ are statistically close, and computationally secure if they are computationally indistinguishable.

Security with Fairness

In this definition, the execution of the protocol can terminate in two possible ways: the first is when all parties receive their prescribed output (as in the case of guaranteed output delivery) and the second is when all parties (including the corrupted parties) abort without receiving output. The only change from the definition in Section 2.3.5 is with regard to the ideal model for computing f, which is now defined as follows:

Execution in the ideal model. The parties are $P_1, \ldots, P_n$, and there is an adversary $\mathcal{A}$ who has corrupted at most t parties, where $0 \leq t < n$. Denote by $\mathcal{I} \subseteq [n]$ the set of indices of the parties corrupted by $\mathcal{A}$. An ideal execution for the computation of f proceeds as follows:

- **Inputs:** $P_1, \ldots, P_n$ hold their private inputs $x_1, \ldots, x_n \in \mathcal{X}$; the adversary $\mathcal{A}$ receives an auxiliary input z.

- **Send inputs to trusted party:** The honest parties send their inputs to the trusted party. The corrupted parties controlled by $\mathcal{A}$ may send any values of their choice. In addition, there exists a special abort input. Denote the inputs sent to the trusted party by $x'_1, \ldots, x'_n$.

- **Trusted party sends outputs:** If $x'_i \notin \mathcal{X}$ for any $i \in [n]$, the trusted party sets x'_i to some default input in $\mathcal{X}$. If there exists an $i \in [n]$ such that $x'_i = \mathsf{abort}$, the trusted party sends $\perp$ to all the parties. Otherwise, the trusted party chooses r uniformly at random, computes $z_i = f^i(x'_1, \ldots, x'_n; r)$ for every $i \in [n]$ and sends z_i to P_i for every $i \in [n]$.

- **Outputs:** The honest parties output whatever was sent by the trusted party. The corrupted parties output nothing and $\mathcal{A}$ outputs an arbitrary (probabilistic polynomial-time computable) function of its view.

We let $\textsc{Ideal}^{\mathsf{fair}}_{f,\mathcal{I},\mathcal{S}(z)}(\overrightarrow{x}, \lambda)$ be the random variable consisting of the output of the adversary and the output of the honest parties following an execution in the ideal model described above.

Definition 2. *Protocol π is said to securely compute f with fairness if for every non-uniform probabilistic polynomial-time adversary $\mathcal{A}$ in the real model, there exists a non-uniform probabilistic polynomial-time adversary $\mathcal{S}$ in the ideal model such that for every $\mathcal{I} \subseteq [n]$ with $|\mathcal{I}| \leq t$,*

$$\left\{\textsc{Ideal}^{\mathsf{fair}}_{f,\mathcal{I},\mathcal{S}(z)}(\overrightarrow{x}, \lambda)\right\}_{\overrightarrow{x} \in \mathcal{X}^n, z \in \{0,1\}^*} \equiv \left\{\textsc{Real}_{\pi,\mathcal{I},\mathcal{A}(z)}(\overrightarrow{x}, \lambda)\right\}_{\overrightarrow{x} \in \mathcal{X}^n, z \in \{0,1\}^*}$$

We will sometimes relax security to statistical or computational definitions. A protocol is statistically secure if the random variables $\text{IDEAL}^{\mathsf{fair}}_{f,\mathcal{I},\mathcal{S}(z)}(\overrightarrow{x},\lambda)$ and $\text{REAL}_{\pi,\mathcal{I},\mathcal{A}(z)}(\overrightarrow{x},\lambda)$ are statistically close, and computationally secure if they are computationally indistinguishable.

Security with Fairness and Identifiable Abort

This definition is identical to the one for fairness, except that if the adversary aborts the computation, all honest parties learn the identity of one of the corrupted parties. The only change from the definition in Section 2.3.5 is with regard to the ideal model for computing f, which is now defined as follows:

Execution in the ideal model. The parties are $P_1, \ldots, P_n$, and there is an adversary $\mathcal{A}$ who has corrupted at most t parties, where $0 \leq t < n$. Denote by $\mathcal{I} \subseteq [n]$ the set of indices of the parties corrupted by $\mathcal{A}$. An ideal execution for the computation of f proceeds as follows:

- **Inputs:** $P_1, \ldots, P_n$ hold their private inputs $x_1, \ldots, x_n \in \mathcal{X}$; the adversary $\mathcal{A}$ receives an auxiliary input z.
- **Send inputs to trusted party:** The honest parties send their inputs to the trusted party. The corrupted parties controlled by $\mathcal{A}$ may send any values of their choice. In addition, there exists a special abort input. In case the adversary instructs P_i to send abort, it chooses an index of a corrupted party $i^* \in \mathcal{I}$ and sets $x'_i = (\mathsf{abort}, i^*)$. Denote the inputs sent to the trusted party by $x'_1, \ldots, x'_n$.
- **Trusted party sends outputs:** If $x'_i \notin \mathcal{X}$ for any $i \in [n]$, the trusted party sets x'_i to some default input in $\mathcal{X}$. If there exists an $i \in [n]$ such that $x'_i = (\mathsf{abort}, i^*)$ and $i^* \in \mathcal{I}$, the trusted party sends $(\perp, i^*)$ to all the parties. Otherwise, the trusted party chooses r uniformly at random, computes $z_i = f^i(x'_1, \ldots, x'_n; r)$ for every $i \in [n]$ and sends z_i to P_i for every $i \in [n]$.
- **Outputs:** The honest parties output whatever was sent by the trusted party.

The corrupted parties output nothing and $\mathcal{A}$ outputs an arbitrary (probabilistic polynomial-time computable) function of its view.

We let $\text{IDEAL}^{\text{id-fair}}_{f,\mathcal{I},\mathcal{S}(z)}(\overrightarrow{x},\lambda)$ be the random variable consisting of the output of the adversary and the output of the honest parties following an execution in the ideal model described above.

Definition 3. *Protocol π is said to securely compute f with fairness and identifiable abort if for every non-uniform probabilistic polynomial-time adversary $\mathcal{A}$ in the real model, there exists a non-uniform probabilistic polynomial-time adversary $\mathcal{S}$ in the ideal model such that for every $\mathcal{I} \subseteq [n]$ with $|\mathcal{I}| \leq t$,*

$$\left\{\text{IDEAL}^{\text{id-fair}}_{f,\mathcal{I},\mathcal{S}(z)}(\overrightarrow{x},\lambda)\right\}_{\overrightarrow{x}\in\mathcal{X}^n,z\in\{0,1\}^*} \equiv \left\{\text{REAL}_{\pi,\mathcal{I},\mathcal{A}(z)}(\overrightarrow{x},\lambda)\right\}_{\overrightarrow{x}\in\mathcal{X}^n,z\in\{0,1\}^*}$$

We will sometimes relax security to statistical or computational definitions. A protocol is statistically secure if the random variables $\text{IDEAL}^{\text{id-fair}}_{f,\mathcal{I},\mathcal{S}(z)}(\overrightarrow{x},\lambda)$ and $\text{REAL}_{\pi,\mathcal{I},\mathcal{A}(z)}(\overrightarrow{x},\lambda)$ are statistically close, and computationally secure if they are computationally indistinguishable.

Security with Abort

This definition is the standard one for secure computation [67] in that it allows *early abort*; that is, the adversary may receive its own output even though the honest party does not. However, if one honest party receives output, then so do all honest parties. Thus, this is the notion of *unanimous abort*. The only change from the definition in Section 2.3.5 is with regard to the ideal model for computing f, which is now defined as follows:

Execution in the ideal model. The parties are $P_1, \ldots, P_n$, and there is an adversary $\mathcal{A}$ who has corrupted at most t parties, where $0 \leq t < n$. Denote by $\mathcal{I} \subseteq [n]$ the set of indices of the parties corrupted by $\mathcal{A}$. An ideal execution for the computation of f proceeds as follows:

- **Inputs:** $P_1, \ldots, P_n$ hold their private inputs $x_1, \ldots, x_n \in \mathcal{X}$; the adversary $\mathcal{A}$ receives an auxiliary input z.

- **Send inputs to trusted party:** The honest parties send their inputs to the trusted party. The corrupted parties controlled by $\mathcal{A}$ may send any values of their choice. In addition, there exists a special abort input. Denote the inputs sent to the trusted party by $x'_1, \ldots, x'_n$.

- **Trusted party sends outputs to the adversary:** If $x'_i \notin \mathcal{X}$ for any $i \in [n]$, the trusted party sets x'_i to some default input in $\mathcal{X}$. If there exists an $i \in [n]$ such that $x'_i = \mathsf{abort}$, the trusted party sends $\perp$ to all the parties. Otherwise, the trusted party chooses r uniformly at random, computes $z_i = f^i(x'_1, \ldots, x'_n; r)$ for every $i \in [n]$ and sends z_i to P_i for every $i \in \mathcal{I}$ (that is, to the adversary $\mathcal{A}$).

- **Trusted party sends outputs to the honest parties:** After receiving its output (as described above), the adversary either sends abort or $\mathsf{continue}$ to the trusted party. In the former case the trusted party sends $\perp$ to the honest parties, and in the latter case the trusted party send z_j to P_j for every $j \in [n] \backslash \mathcal{I}$.

- **Outputs:** The honest parties output whatever was sent by the trusted party. The corrupted parties output nothing and $\mathcal{A}$ outputs an arbitrary (probabilistic polynomial-time computable) function of its view.

We let $\textsc{Ideal}^{\mathsf{abort}}_{f,\mathcal{I},\mathcal{S}(z)}(\overrightarrow{x}, \lambda)$ be the random variable consisting of the output of the adversary and the output of the honest parties following an execution in the ideal model described above.

Definition 4. *Protocol π is said to securely compute f with abort if for every non-uniform probabilistic polynomial-time adversary $\mathcal{A}$ in the real model, there exists a non-uniform probabilistic polynomial-time adversary $\mathcal{S}$ in the ideal model such that for every $\mathcal{I} \subseteq [n]$ with $|\mathcal{I}| \leq t$,*

$$\left\{\textsc{Ideal}^{\mathsf{abort}}_{f,\mathcal{I},\mathcal{S}(z)}(\overrightarrow{x}, \lambda)\right\}_{\overrightarrow{x} \in \mathcal{X}^n, z \in \{0,1\}^*} \equiv \left\{\textsc{Real}_{\pi,\mathcal{I},\mathcal{A}(z)}(\overrightarrow{x}, \lambda)\right\}_{\overrightarrow{x} \in \mathcal{X}^n, z \in \{0,1\}^*}$$

We will sometimes relax security to statistical or computational definitions. A protocol is statistically secure if the random variables $\text{IDEAL}^{\mathsf{abort}}_{f,\mathcal{I},\mathcal{S}(z)}(\overrightarrow{x},\lambda)$ and $\text{REAL}_{\pi,\mathcal{I},\mathcal{A}(z)}(\overrightarrow{x},\lambda)$ are statistically close, and computationally secure if they are computationally indistinguishable.

Security with Identifiable Abort

This definition is identical to the one for abort, except that if the adversary aborts the computation, all honest parties learn the identity of one of the corrupted parties. The only change from the definition in Section 2.3.5 is with regard to the ideal model for computing f, which is now defined as follows:

Execution in the ideal model. The parties are $P_1, \ldots, P_n$, and there is an adversary $\mathcal{A}$ who has corrupted at most t parties, where $0 \leq t < n$. Denote by $\mathcal{I} \subseteq [n]$ the set of indices of the parties corrupted by $\mathcal{A}$. An ideal execution for the computation of f proceeds as follows:

- **Inputs:** $P_1, \ldots, P_n$ hold their private inputs $x_1, \ldots, x_n \in \mathcal{X}$; the adversary $\mathcal{A}$ receives an auxiliary input z.

- **Send inputs to trusted party:** The honest parties send their inputs to the trusted party. The corrupted parties controlled by $\mathcal{A}$ may send any values of their choice. In addition, there exists a special abort input. In case the adversary instructs P_i to send abort, it chooses an index of a corrupted party $i^* \in \mathcal{I}$ and sets $x'_i = (\mathsf{abort}, i^*)$. Denote the inputs sent to the trusted party by $x'_1, \ldots, x'_n$.

- **Trusted party sends outputs to the adversary:** If $x'_i \notin \mathcal{X}$ for any $i \in [n]$, the trusted party sets x'_i to some default input in $\mathcal{X}$. If there exists an $i \in [n]$ such that $x'_i = (\mathsf{abort}, i^*)$ and $i^* \in \mathcal{I}$, the trusted party sends $(\perp, i^*)$ to all the parties. Otherwise, the trusted party chooses r uniformly at random, computes $z_i = f^i(x'_1, \ldots, x'_n; r)$ for every $i \in [n]$ and sends z_i to P_i for every $i \in \mathcal{I}$ (that is, to the adversary $\mathcal{A}$).

- **Trusted party sends outputs to the honest parties:** After receiving its output (as described above), the adversary either sends (abort, i^*) where $i^* \in \mathcal{I}$, or $\mathsf{continue}$ to the trusted party. In the former case the trusted party sends $(\perp, i^*)$ to the honest parties, and in the latter case the trusted party send z_j to P_j for every $j \in [n] \setminus \mathcal{I}$.

- **Outputs:** The honest parties output whatever was sent by the trusted party. The corrupted parties output nothing and $\mathcal{A}$ outputs an arbitrary (probabilistic polynomial-time computable) function of its view.

We let $\textsc{Ideal}^{\mathsf{id\text{-}abort}}_{f,\mathcal{I},\mathcal{S}(z)}(\overrightarrow{x}, \lambda)$ be the random variable consisting of the output of the adversary and the output of the honest parties following an execution in the ideal model described above.

Definition 5. *Protocol π is said to securely compute f with identifiable abort if for every non-uniform probabilistic polynomial-time adversary $\mathcal{A}$ in the real model, there exists a non-uniform probabilistic polynomial-time adversary $\mathcal{S}$ in the ideal model such that for every $\mathcal{I} \subseteq [n]$ with $|\mathcal{I}| \leq t$,*

$$\left\{\textsc{Ideal}^{\mathsf{id\text{-}abort}}_{f,\mathcal{I},\mathcal{S}(z)}(\overrightarrow{x}, \lambda)\right\}_{\overrightarrow{x} \in \mathcal{X}^n, z \in \{0,1\}^*} \equiv \left\{\textsc{Real}_{\pi,\mathcal{I},\mathcal{A}(z)}(\overrightarrow{x}, \lambda)\right\}_{\overrightarrow{x} \in \mathcal{X}^n, z \in \{0,1\}^*}$$

We will sometimes relax security to statistical or computational definitions. A protocol is statistically secure if the random variables $\textsc{Ideal}^{\mathsf{id\text{-}abort}}_{f,\mathcal{I},\mathcal{S}(z)}(\overrightarrow{x}, \lambda)$ and $\textsc{Real}_{\pi,\mathcal{I},\mathcal{A}(z)}(\overrightarrow{x}, \lambda)$ are statistically close, and computationally secure if they are computationally indistinguishable.

2.4 The Hybrid Model

We recall the definition of the hybrid model from [72] and [42]. The hybrid model combines both the real and ideal worlds. Specifically, an execution of a protocol π in the $\mathcal{G}$-hybrid model, for some functionality $\mathcal{G}$, involves parties sending normal messages to each other (as in the real model) and, in addition, having access to a

trusted party computing $\mathcal{G}$. The parties communicate with this trusted party in exactly the same way as in the ideal models described above; the question of which ideal model is taken (that with or without abort) must be specified. In this work, we always consider a hybrid model where the functionality $\mathcal{G}$ is computed according to the ideal model *with abort*. In all our protocols in the $\mathcal{G}$-hybrid model there will only be *sequential* calls to $\mathcal{G}$, that is, there is at most a single call to $\mathcal{G}$ per round, and no other messages are sent during any round in which $\mathcal{G}$ is called. This is especially important for reactive functionalities, where the calls to f are carried out in phases, and a new invocation of f cannot take place before all the phases of the previous invocation complete.

Let $\mathsf{type} \in \{\mathsf{g.d.}, \mathsf{fair}, \mathsf{id\text{-}fair}, \mathsf{abort}, \mathsf{id\text{-}abort}\}$. Let $\mathcal{G}$ be a functionality and let π be an n-party protocol for computing some functionality f, where π includes real messages between the parties as well as calls to $\mathcal{G}$. Let $\mathcal{A}$ be a non-uniform probabilistic polynomial-time machine with auxiliary input z. $\mathcal{A}$ corrupts at most t parties, where $0 \leq t < n$. Denote by $\mathcal{I} \subseteq [n]$ the set of indices of the parties corrupted by $\mathcal{A}$. Let $\text{HYBRID}^{\mathcal{G},\mathsf{type}}_{\pi,\mathcal{I},\mathcal{A}(z)}(\overrightarrow{x}, \lambda)$ be the random variable consisting of the view of the adversary and the output of the honest parties, following an execution of π with ideal calls to a trusted party computing $\mathcal{G}$ according to the ideal model "type" where P_i begins by holding x_i for every $i \in [n]$. Security in the model "type" can be defined via natural modifications of Definitions 1, 2, 3, 4 and 5. We call this the $(\mathcal{G}, \mathsf{type})$-hybrid model.

The hybrid model gives a powerful tool for proving the security of protocols. Specifically, we may design a real-world protocol for securely computing some functionality f by first constructing a protocol for computing f in the $\mathcal{G}$-hybrid model. Letting π denote the protocol thus constructed (in the $\mathcal{G}$-hybrid model), we denote by π^ρ the real-world protocol in which calls to $\mathcal{G}$ are replaced by sequential execution of a real-world protocol ρ that computes $\mathcal{G}$ in the ideal model "type". "Sequential" here implies that only one execution of ρ is carried out at any time, and no other π-protocol messages are sent during the execution of ρ. The results of [29] then imply that if π securely computes f in the $(\mathcal{G}, \mathsf{type})$-hybrid model, and ρ securely computes $\mathcal{G}$, then the composed protocol π^ρ securely computes f (in the real world). For completeness,

we state this result formally as we will use it in this work.

Lemma 1. *Let* $\mathsf{type}_1, \mathsf{type}_2 \in \{\mathsf{g.d.}, \mathsf{fair}, \mathsf{id\text{-}fair}, \mathsf{abort}, \mathsf{id\text{-}abort}\}$. *Let* $\mathcal{G}$ *be an* n*-party functionality. Let* ρ *be a protocol that securely computes* $\mathcal{G}$ *with* type_1*, and let* π *be a protocol that securely computes* f *with* type_2 *in the* $(\mathcal{G}, \mathsf{type}_1)$*-hybrid model. Then protocol* π^ρ *securely computes* f *with* type_2 *in the real model.*

Sometimes, while working in a hybrid model, say the $(\mathcal{G}, \mathsf{type})$-hybrid model, we will suppress type and simply state that we are working in the $\mathcal{G}$-hybrid model. This is because type is implied by the context, $\mathcal{G}$. For instance, unless specified otherwise:

- When $\mathcal{G} = \mathcal{F}_{\mathsf{bc}}$[6], $\mathsf{type} = \mathsf{g.d.}$.
- When $\mathcal{G} = \mathcal{F}_{\mathsf{OT}}$[7], $\mathsf{type} = \mathsf{abort}$.
- When $\mathcal{G} = \mathcal{F}_{\mathsf{2PC}}$[8], $\mathsf{type} = \mathsf{abort}$.
- When $\mathcal{G} = \mathcal{F}_{\mathsf{MPC}}$[9], $\mathsf{type} = \mathsf{abort}$.
- When $\mathcal{G} = \mathcal{F}_{\mathsf{SyX}}$[10], $\mathsf{type} = \mathsf{g.d.}$.

When working in a hybrid model that uses multiple ideal functionalities, $\mathcal{G}_1, \ldots, \mathcal{G}_k$ with associated types $\mathsf{type}_1, \ldots, \mathsf{type}_k$ for some $k \in \mathbb{N}$, we call it the $(\mathcal{G}_1, \mathsf{type}_1, \ldots, \mathcal{G}_k, \mathsf{type}_k)$-hybrid model. Furthermore, we will suppress type_j when type_j is implied by the context, $\mathcal{G}_j$ for $j \in [k]$.

2.5 Computing with an Honest Majority

We recall here some of the known results regarding feasibility of information-theoretic multiparty computation in the presence of an honest majority.

[6] See Section 2.7.
[7] See Section 2.6.
[8] See Section 5.2.
[9] See Section 2.6.
[10] See Section 5.2.

Preliminaries: $x_0, x_1 \in \{0,1\}^m$; $b \in \{0,1\}$. The functionality proceeds as follows:

- Upon receiving inputs (x_0, x_1) from the sender P_1 and b from the receiver P_2, send $\perp$ to P_1 and x_b to P_2.

Figure 2-1: The ideal functionality $\mathcal{F}_{\mathsf{OT}}$.

Lemma 2. *[68] Consider n parties $P_1, \ldots, P_n$ in the* point-to-point *model. Then, there exists a protocol π which securely computes $\mathcal{F}_{\mathsf{MPC}}$ with guaranteed output delivery in the presence of t-threshold adversaries for any $0 \leq t < n/3$.*

Lemma 3. *[58] Consider n parties $P_1, \ldots, P_n$ in the* point-to-point *model. Then, there exists a protocol π which securely computes $\mathcal{F}_{\mathsf{MPC}}$ with fairness in the presence of t-threshold adversaries for any $0 \leq t < n/2$.*

Lemma 4. *[68, 119] Consider n parties $P_1, \ldots, P_n$ in the* broadcast *model. Then, there exists a protocol π which securely computes $\mathcal{F}_{\mathsf{MPC}}$ with guaranteed output delivery in the presence of t-threshold adversaries for any $0 \leq t < n/2$.*

2.6 Oblivious Transfer

Oblivious transfer, or OT, refers to 1-out-of-2 oblivious transfer defined as in Figure 2-1. We note that in the definition of $\mathcal{F}_{\mathsf{OT}}$, one party, namely P_1, is seen as the sender, while the other, namely P_2, is seen as the receiver. However, from [123], OT is symmetric, which implies that the roles of the sender and the receiver can be reversed. Thus, if two parties P_1 and P_2 have access to the ideal functionality $\mathcal{F}_{\mathsf{OT}}$, they can perform 1-out-of-2 oblivious transfer with either party as a sender and the other as the receiver. It is known that OT is complete for secure multiparty computation with abort. We state this result formally below.

Lemma 5. *[89, 70, 86] Consider n parties $P_1, \ldots, P_n$ in the* OT-network *model. Then, there exists a protocol π which securely computes $\mathcal{F}_{\mathsf{MPC}}$ with abort in the presence of t-threshold adversaries for any $0 \leq t < n$.*

Preliminaries: $x_1, \ldots, x_n \in \{0,1\}^*$; $f_1, \ldots, f_n$ is an n-input, n-output functionalities. The functionality proceeds as follows:

- Upon receiving inputs (x_i, f_i) from P_i for all $i \in [n]$, check if $f = f_i$ for all $i \in [n]$. If not, abort. Else, send $f^i(x_1, \ldots, x_n)$ to P_i for all $i \in [n]$.

Figure 2-2: The ideal functionality $\mathcal{F}_{\mathsf{MPC}}$.

Preliminaries: $x \in \{0,1\}^*$. The functionality proceeds as follows:

- Upon receiving the input x from the sender P_1, send x to all parties $P_1, \ldots, P_n$.

Figure 2-3: The ideal functionality $\mathcal{F}_{\mathsf{bc}}$.

2.7 Broadcast

Broadcast is defined as in Figure 2-3. We recall that the ideal functionality for broadcast, namely $\mathcal{F}_{\mathsf{bc}}$, can be securely computed with guaranteed output delivery in the presence of t-threshold adversaries if and only if $0 \leq t < n/3$ [115, 100]. Furthermore, $\mathcal{F}_{\mathsf{bc}}$ can be securely computed with fairness in the presence of t-threshold adversaries for any $0 \leq t < n$ [59]. Furthermore, these results hold irrespective of the model we are working in so long as we do not have explicit access to $\mathcal{F}_{\mathsf{bc}}$.

2.8 Honest-Binding Commitment Schemes

We recall the notion of honest-binding commitments from [62]. Commitment schemes are a standard cryptographic tool. Roughly, a commitment scheme allows a sender S to generate a commitment c to a message m in such a way that (1) the sender can later open the commitment to the original value m (correctness); (2) the sender cannot generate a commitment that can be opened to two different values (binding); and (3) the commitment reveals nothing about the sender's value m until it is opened (hiding). For our application, we need a variant of standard commitments that guarantees binding when the sender is honest but ensures that binding can be violated if the sender is dishonest. (In the latter case, we need some additional properties as well;

these will become clear in what follows.) Looking ahead, we will use such commitment schemes to enable a simulator in security proofs to generate a commitment dishonestly. This will give the simulator the flexibility to break binding and open the commitment to any desired message (if needed), while also being able to ensure binding (when desired) by claiming that it generated the commitment honestly.

We consider only non-interactive commitment schemes. For simplicity, we define our schemes in such a way that the decommitment information consists of the sender's random coins ω that it used when generating the commitment.

Definition 6. *A (non-interactive) commitment scheme for message space $\mathcal{M}_\lambda$ is a pair of PPT algorithms $(\mathsf{Com}, \mathsf{Open})$ such that for all $\lambda \in \mathbb{N}$, all messages $m \in \mathcal{M}_\lambda$, and all random coins ω it holds that*

$$\mathsf{Open}(\mathsf{Com}(1^\lambda, m; \omega), \omega, m) = 1$$

A commitment scheme for message space $\mathcal{M}_\lambda$ is honest-binding if it satisfies the following:

Binding (for an honest sender). *For all PPT algorithms $\mathcal{A}$ (that maintain state throughout their execution), the following probability is negligible in λ:*

$$\Pr\left[\begin{array}{lll} m \xleftarrow{\$} \mathcal{A}(1^k); \omega \xleftarrow{\$} \{0,1\}^* & & \\ \quad c = \mathsf{Com}(1^\lambda, m; \omega) & : & \mathsf{Open}(c, m', \omega') = 1 \ \wedge \ m \neq m' \\ \quad (m', \omega') \xleftarrow{\$} \mathcal{A}(c, \omega) & & \end{array}\right]$$

Equivocation. *There is a pair of algorithms $\left(\widetilde{\mathsf{Com}}, \widetilde{\mathsf{Open}}\right)$ such that for all PPT algorithms $\mathcal{A}$ (that maintain state throughout their execution), the following quantity is negligible in λ:*

$$\left| \begin{array}{l} \Pr\left[m \xleftarrow{\$} \mathcal{A}(1^\lambda); \omega \xleftarrow{\$} \{0,1\}^*; c = \mathsf{Com}(1^\lambda, m; \omega) : \mathcal{A}(1^\lambda, c, \omega) = 1\right] \\ \qquad - \Pr\left[(c, \mathsf{state}) \xleftarrow{\$} \widetilde{\mathsf{Com}}(1^\lambda), m \xleftarrow{\$} \mathcal{A}(1^\lambda); \omega \xleftarrow{\$} \widetilde{\mathsf{Open}}(\mathsf{state}, m) : \mathcal{A}(1^\lambda, c, \omega) = 1\right] \end{array} \right|$$

Equivocation implies the standard hiding property, namely, that for all PPT algorithms $\mathcal{A}$ (that maintain state throughout their execution) the quantity is negligible in λ:

$$\left| \Pr\left[(m_0, m_1) \xleftarrow{\$} \mathcal{A}(1^\lambda); b \xleftarrow{\$} \{0,1\}; c \xleftarrow{\$} \mathsf{Com}(1^\lambda, m_b) : \mathcal{A}(c) = b\right] \right|$$

We also observe that if (c, ω) are generated by $\left(\widetilde{\mathsf{Com}}, \widetilde{\mathsf{Open}}\right)$ for some message m as in the definition above, then binding still holds: namely, no PPT adversary given (m, c, ω) can find (m', ω') with $m' \neq m$ such that $\mathsf{Open}(c, m', \omega') = 1$.

We will sometimes use the notation $(c, \omega) \xleftarrow{\$} \mathsf{Com}(m)$ to mean $c = \mathsf{Com}(1^\lambda, m; \omega)$, suppressing λ when it is clear from the context and having the committing algorithm Com return the commitment and the decommitment information or opening. [62] provides constructions of honest-binding commitments for bits assuming the existence of one-way functions.

2.9 Digital Signatures

Definition 7. *A (digital) signature scheme for message space $\mathcal{M}_\lambda$ is triple of PPT algorithms $\mathcal{V} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ such that for all $\lambda \in \mathbb{N}$ and all messages $m \in \mathcal{M}_\lambda$,*

$$\Pr\left[\begin{array}{c} (\mathsf{vk}, \mathsf{sk}) \xleftarrow{\$} \mathcal{V}.\mathsf{Gen}(1^\lambda) \\ \sigma \xleftarrow{\$} \mathcal{V}.\mathsf{Sign}(m; \mathsf{sk}) \end{array} : \mathcal{V}.\mathsf{Verify}(\sigma, m; \mathsf{vk}) = 1\right] = 1$$

A signature scheme for message space $\mathcal{M}_\lambda$ is existentially unforgeable if for any PPT adversary $\mathcal{A}$, the following probability is negligible in λ:

$$\Pr\left[\begin{array}{c} (\mathsf{vk}, \mathsf{sk}) \xleftarrow{\$} \mathcal{V}.\mathsf{Gen}(1^\lambda) \\ \mathcal{Q} = \emptyset \\ \left\{\begin{array}{c} m_i \xleftarrow{\$} \mathcal{A}(1^\lambda, \mathcal{Q}) \\ \sigma_i \xleftarrow{\$} \mathcal{V}.\mathsf{Sign}(m_i; \mathsf{sk}) \\ \mathcal{Q} = \mathcal{Q} \cup \{(m_i, \sigma_i)\} \end{array}\right\}_i \end{array} : \mathcal{V}.\mathsf{Verify}(\sigma, m; \mathsf{vk}) = 1 \ \wedge \ (m, \sigma) \notin \mathcal{Q}\right]$$

[120] provides constructions of existentially unforgeable signatures assuming the

existence of one-way functions.

2.10 Receiver Non-Committing Encryption

We recall the notion of receiver non-committing encryption from [32]. On a high level, a receiver non-committing encryption scheme is one in which a simulator can generate a single "fake ciphertext" and later "open" this ciphertext (by showing an appropriate secret key) as any given message. These "fake ciphertexts" should be indistinguishable from real ciphertexts, even when an adversary is given access to a decryption oracle before the fake ciphertext is known.

Formally, a receiver non-committing encryption scheme $\mathcal{E}$ consists of the following five PPT algorithms:

- $\mathcal{E}.\mathsf{Gen}(1^\lambda)$: Given the security parameter, λ, the key generation algorithm outputs a key-pair and some auxiliary information. This is denoted by: $(\mathsf{pk}, \mathsf{sk}, z) \xleftarrow{\$} \mathcal{E}.\mathsf{Gen}(1^\lambda)$. The public key pk defines a message space $\mathcal{M}_\lambda$.

- $\mathcal{E}.\mathsf{Enc}(m; \mathsf{pk})$: Given the public key pk and a message $m \in \mathcal{M}_\lambda$, the encryption algorithm returns a ciphertext $\mathsf{ct} \xleftarrow{\$} \mathcal{E}.\mathsf{Enc}(m; \mathsf{pk})$.

- $\mathcal{E}.\mathsf{Dec}(\mathsf{ct}; \mathsf{sk})$: Given the secret key sk and a ciphertext ct, the decryption algorithm returns a message $m \xleftarrow{\$} \mathcal{E}.\mathsf{Dec}(\mathsf{ct}; \mathsf{sk})$, where $m \in \mathcal{M}_\lambda \cup \{\perp\}$.

- $\mathcal{E}.\widetilde{\mathsf{Enc}}(\mathsf{pk}, \mathsf{sk}, z)$: Given the triple $(\mathsf{pk}, \mathsf{sk}, z)$ output by $\mathcal{E}.\mathsf{Gen}$, the fake encryption algorithm outputs a "fake ciphertext" $\widetilde{\mathsf{ct}} \xleftarrow{\$} \mathcal{E}.\widetilde{\mathsf{Enc}}(\mathsf{pk}, \mathsf{sk}, z)$.

- $\mathcal{E}.\widetilde{\mathsf{Dec}}(\mathsf{pk}, \mathsf{sk}, z, \widetilde{\mathsf{ct}}, m)$: Given the triple $(\mathsf{pk}, \mathsf{sk}, z)$ output by $\mathcal{E}.\mathsf{Gen}$, a "fake ciphertext" $\widetilde{\mathsf{ct}}$ output by $\mathcal{E}.\widetilde{\mathsf{Enc}}$ and a message $m \in \mathcal{M}_\lambda$, the "fake decryption" algorithm outputs a "fake secret key" $\widetilde{\mathsf{sk}} \xleftarrow{\$} \mathcal{E}.\widetilde{\mathsf{Dec}}(\mathsf{pk}, \mathsf{sk}, z, \widetilde{\mathsf{ct}}, m)$. (Intuitively, $\widetilde{\mathsf{sk}}$ is a valid-looking secret key for which $\widetilde{\mathsf{ct}}$ decrypts to m.)

We make the standard correctness requirement; namely, for any $(\mathsf{pk}, \mathsf{sk}, z)$ output by $\mathcal{E}.\mathsf{Gen}$ and any $m \in \mathcal{M}_\lambda$, we have $\mathcal{E}.\mathsf{Dec}(\mathcal{E}.\mathsf{Enc}(m; \mathsf{pk}); \mathsf{sk}) = m$. Our definition of security requires, informally, that for any message m an adversary cannot distinguish

whether it has been given a "real" encryption of m along with a "real" secret key, or a "fake" ciphertext along with a "fake" secret key under which the ciphertext decrypts to m. This should hold even when the adversary has non-adaptive access to a decryption oracle. We now give the formal definition.

Definition 8. *Let $\mathcal{E}$ be a receiver non-committing encryption scheme. We say that $\mathcal{E}$ is secure if the advantage of any PPT algorithm $\mathcal{A}$ in the game below is negligible in λ:*

1. *The key generation algorithm $\mathcal{E}.\mathsf{Gen}(1^\lambda)$ is run to get $(\mathsf{pk}, \mathsf{sk}, z)$.*
2. *The algorithm $\mathcal{A}$ is given 1^λ and pk as input, and is also given access to a decryption oracle $\mathcal{E}.\mathsf{Dec}(\cdot; \mathsf{sk})$. It then outputs a challenge message $m \in \mathcal{M}_\lambda$.*
3. *A bit b is chosen at random. If $b = 1$ then a ciphertext $\mathsf{ct} \xleftarrow{\$} \mathcal{E}.\mathsf{Enc}(m; \mathsf{pk})$ is computed, and $\mathcal{A}$ receives $(\mathsf{ct}, \mathsf{sk})$. Otherwise, a "fake" ciphertext $\widetilde{\mathsf{ct}} \xleftarrow{\$} \widetilde{\mathcal{E}.\mathsf{Enc}}(\mathsf{pk}, \mathsf{sk}, z)$ and a "fake" secret key $\widetilde{\mathsf{sk}} \xleftarrow{\$} \widetilde{\mathcal{E}.\mathsf{Dec}}(\mathsf{pk}, \mathsf{sk}, z, \widetilde{\mathsf{ct}}, m)$ are computed, and $\mathcal{A}$ receives $(\widetilde{\mathsf{ct}}, \widetilde{\mathsf{sk}})$. (After this point, $\mathcal{A}$ can no longer query its decryption oracle.) $\mathcal{A}$ outputs a bit $b \in \{0, 1\}$.*

The advantage of $\mathcal{A}$ is defined as $2 \cdot \left|\Pr[b = b'] - \frac{1}{2}\right|$.

2.11 Non-interactive Non-Committing Encryption

We recall the notion of non-interactive non-committing encryption from [112]. We do so in two ways. The first way of looking at non-interactive non-committing encryption is that it is the same as receiver non-committing encryption, except that it can equivocate multiple ciphertexts as opposed to one. On a high level, a non-interactive non-committing encryption scheme is one in which a simulator can generate multiple "fake ciphertexts" and later "open" them (by showing an appropriate secret key) as any given message vector. We first note that the receiver non-committing encryption scheme of [32] can be extended, as noted by them, to support equivocation of any bounded number of ciphertexts. However, the size of the key of the scheme would

grow linearly with the number of outstanding ciphertexts. Such schemes can be constructed based on standard assumptions such as the quadratic residuosity assumption. If no bound on the number of outstanding texts is known *apriori*, then as noted in [112], constructing such schemes is impossible in the standard model. The other way of looking at non-interactive non-committing encryption is that it is a realization of the ideal functionality for public key encryption, namely, $\mathcal{F}_{\mathsf{PKE}}$. We refer the reader to [33, 32] for further details.

For the sake of completeness and ease of later presentation, we recall the non-interactive non-committing encryption scheme of [112] in the random-oracle model. Let $\mathcal{F} = (\mathcal{K}, F)$ be a collection of trapdoor permutations, where $\mathcal{K}$ denotes an index set and $F = \{f_k\}_{k \in \mathcal{K}}$ is a set of permutations with efficiently samplable domains. For every $k \in \mathcal{K}$, we denote by t_k the trapdoor associated with k which enables inversion of f_k. We assume the existence of a generation algorithm $\mathcal{G}$ which on input the security parameter λ outputs a key-trapdoor pair (k, t_k) uniformly at random. Let $H : \{0,1\}^* \to \{0,1\}^{\ell(\lambda)}$ be a random oracle (instantiated by an appropriate hash function). The non-interactive non-committing encryption scheme $\mathcal{E}$ consists of the following algorithms:

- $\mathcal{E}.\mathsf{Gen}(1^\lambda)$: Given the security parameter, λ, the key generation algorithm obtains (k, t_k) by executing $\mathcal{G}$ with the security parameter λ as input. It then outputs the public and private keys $\mathsf{pk} = (k, f_k, H)$ and $\mathsf{sk} = t_k$. The message space is defined to be $\mathcal{M}_\lambda = \{0,1\}^{\ell(\lambda)}$.

- $\mathcal{E}.\mathsf{Enc}(m; \mathsf{pk})$: Given the public key pk and a message $m \in \mathcal{M}_\lambda$, the encryption algorithm samples x from the domain of f_k and returns a ciphertext $\mathsf{ct} = (f_k(x), H(x) \oplus m)$.

- $\mathcal{E}.\mathsf{Dec}(\mathsf{ct}; \mathsf{sk})$: Given the secret key sk and a ciphertext $\mathsf{ct} = (\mathsf{ct}^1, \mathsf{ct}^2)$, the decryption algorithm computes x by inverting ct^1 using t_k and returns the message $m = H(x) \oplus \mathsf{ct}^2$.

We refer the reader to [112] for a complete proof that the scheme defined above is a non-interactive non-committing encryption scheme. The sketch the proof here.

The scheme is clearly non-interactive. We now need to design a simulator $\mathcal{S}$ which can generate multiple "fake ciphertexts" and later "open" them to an arbitrary sequence of messages. Note that this is easy to do. To generate n "fake ciphertexts", $\mathcal{S}$ samples $x_1, \ldots, x_n$ independently at random from the domain of f_k. It then samples $y_1, \ldots, y_n \xleftarrow{\$} \{0,1\}^{\ell(\lambda)}$. The m ciphertexts are defined to be $\{\mathsf{ct}_i\}_{i\in[n]}$ where $\mathsf{ct}_i = (f_k(x_i), y_i)$. Then, in order to open the n ciphertexts to a message vector $\overrightarrow{m} = (m_1, \ldots, m_n) \in \mathcal{M}_\lambda^n$, $\mathcal{S}$ would program the random oracle H such that $H(x_i) = m_i \oplus y_i$. Note that this ensures that the "fake ciphertexts" do in fact "open" to the message vector $\overrightarrow{m}$. We also stress, as this will be required for us later, that the simulator need not know n in advance, that is, it can produce any (polynomially bounded) number of "fake ciphertexts" and later "open" them as required. This is also precisely the difference from receiver non-committing encryption as described earlier which necessitates the use of random oracles as noted in [112].

Chapter 3

Infrastructure for Reliable Communication

We study the problem of almost-everywhere reliable message transmission*; a key component in designing efficient and secure MPC protocols for sparsely connected networks. The goal is to design low-degree networks which allow a large fraction of honest nodes to communicate reliably even while linearly many nodes can experience byzantine corruption and deviate arbitrarily from the assigned protocol.*

In this work, we achieve a log*-degree network with a polylogarithmic work complexity protocol, thereby improving over the state-of-the-art result of Chandran* et al. *(ICALP 2010) who required a polylogarithmic-degree network and had a linear work complexity. In addition, we also achieve:*

- *A work efficient version of Dwork et. al.'s (STOC 1986) butterfly network.*
- *An improvement upon the state of the art protocol of Ben-or and Ron (Information Processing Letters 1996) in the randomized corruption model—both in work-efficiency and in resilience.*

The results presented in this chapter also appear in a joint work with Siddhartha Jayanti and Nikhil Vyas [87].

3.1 Preliminaries

3.1.1 Graphs: Expanders

Definition 9. *A graph $G = (V, E)$ is an expander if there exists a constant $\theta < 1$ such that for every subset $U \subset V$ of vertices of size $|U| \leq \frac{|V|}{2}$, the set of vertices outside U that have at least one neighbor in U is at least $\theta|U|$.*

Constructions of expanders of constant degree are known [107].

3.1.2 Adversaries

We consider security against *malicious adaptive t-threshold adversaries*, that is, adversaries that adaptively corrupt a set of at most t parties, where $0 \leq t < n$.[1] We will be working with both *worst-case* and *random* adversaries. The randomized adversary model assumes that the t corrupted nodes are chosen uniformly at random from the set of n nodes. We call this model of picking a random subset of size t the Hamming Random Model or corruption. Alternately, a randomized adversary may make each node corrupt with probability t/n; we call this the Shannon model. Basic Chernoff bounds show that the Shannon and Hamming models are equivalent up to a constant factor difference in t with all but exponentially small probability. Thus, we freely switch between the two models in our exposition. While this model of corruption is primarily good for simulating phishing and password guessing attacks, our probabilistic approaches show that it can be the starting point for state of the art protocols against corporation and worst-case adversaries.

3.1.3 Protocoled-Network

Given a graph $G = (V, E)$, a *message transmission protocol* or simply *protocol* Π on the graph, is a specification for how messages are routed between every pair of nodes. In particular, $\Pi(u, v)$ is the protocol for node $u \in V$ to *transmit* to node $v \in V$. A protocol is comprised of discrete synchronous *rounds*. In each round, we allow each

[1] Note that when $t = n$, there is nothing to prove.

node $w \in V$ to perform local computations and pass a different one bit message to each of its neighbors in G.

We call a pair $N = (G, \Pi)$ a *protocoled-network* if Π is a protocol for graph G. We define the following complexity measures of a protocoled-network, where u and v are two different nodes in G:

1. **Work complexity**, or, **Total work:** The total work of $\Pi(u, v)$ is the number computations, $W(u, v)$, performed across all processors in the network in a transmission from u to v. The *total work* of Π is $W = \max_{u,v \in V} W(u, v)$.

2. **Graph degree:** The degree of u is the number of neighbors, $d(u)$, that u has in G. The *degree* of G is $d = \max_{u \in V} d(u)$.

3. **Resilience:** We say a network (G, Π) is *resilient* to a set of nodes T, of size $t = |T|$, being *corrupted* while dooming only x nodes if there is a subset $S \subseteq V$ of $n - t - x$ *privileged* nodes that can reliably transmit messages between each other, after the nodes in T experience byzantine failure. Nodes in set S are called *privileged*, nodes in $X = V - (S \cup T)$ are called *doomed*, and nodes in $X \cup T$ are called *unprivileged*. We say a network is $(f(n), g(t))$-resilient if it can sustain an arbitrary set of up to $t \leq f(n)$ corruptions while dooming at most $x = g(t)$ nodes. When corruptions are randomized (see Section 3.1.2), we say that a network is $(f(n), g(t))$-resilient with probability p, if it can sustain a random subset of up to $t \leq f(n)$ corruptions, and at most $x = g(t)$ nodes get doomed with probability at least p. Informally speaking, a network is highly resilient if $f(n)$ is large while $g(t)$ is not too large, and thus the set of privileged nodes is large.

Our goal is to design a highly resilient low degree network of low work complexity.

3.1.4 Almost-everywhere Security

The notion of almost-everywhere secure primitives was introduced by Dwork, Peleg, Pippenger, Upfal [54]. In this setting, we consider a sparse communication network on

the nodes. We assume a synchronous network and that the communication is divided into rounds. In every round, each node can send (possibly different) messages on its incident edges; these messages are delivered before the next round. Suppose a certain subset of the nodes may be adversarially corrupt, in particular adaptive, rushing and computationally unbounded. This implies that a protocol for any task on this network must "give up" a certain number of honest nodes on account of their poor connectivity to other honest nodes. We set up the following notation. Consider a network of n nodes connected by a communication network $G = (V, E)$ of degree d. On executing a protocol Π on this network in the presence of a subset $T \subset V$ of adversarial or *corrupt* nodes, let $X \subset V$ be the set of honest nodes that are given up, or *doomed*, and let $P \subset V$ be the set of honest nodes for whom the protocol requirements of correctness and security hold, or *privileged* nodes. The nodes that are not privileged are *unprivileged* nodes. Let $|T| = t$, $|X| = x$ and $|S| = s$. We have $t + x + s = n$.

3.1.5 Almost-everywhere Reliable Message Transmission

We present some prior networks for almost-everywhere reliable message transmission that will be useful in our constructions.

Protocoled-Network Construction 1 ([54]). *The butterfly protocoled-network* (G_{But}, Π_{But}) *is as follows.*

> ***Graph****:* $G_{But} = (V_{But}, E_{But})$ *where* $V_{But} = \{(i, j)\}$ *where* $0 \leq i \leq m - 1$ *and* $j \in \{0, 1\}^m$ *is a set of* $n = m2^m$ *nodes, and* $E_{But} = \{(i, j), (i', j')\}$ *is the set of edges where* $i' = (i + 1) \mod m$ *and* j *and* j' *only possibly differ in the* i*th bit.*
>
> ***Protocol****: Let* u *and* v *be distinct vertices in* V_{But}*. There exists as set of paths* $P_{u,v}$ *from* u *to* v *such that* $|P_{u,v}| = 2^m = \Theta(n/\log n)$*. The message transmission protocol* Π *from* u *to* v *in* G_{But} *is as follows:* u *sends the message along all paths* $P_{u,v}$*,* v *receives all the messages and takes majority.*

Theorem 1 ([54]). *For the* $n = m2^m$*-node network* $G_{But} = (V, E)$ *and the protocol* Π_{But} *for message transmission on it, there exists constants* α_1 *and* β_1*, such that:*

1. *The network G_{But} is of constant degree, namely 11.*

2. *The work complexity is $\tilde{O}(n)$.*

3. *If a subset of nodes $T \subset V$ is corrupt, where $|T| \leq \alpha_1 n/\log n$, there exists a set of nodes $S \subset V$ where $|S| \geq n - \beta_1 |T| \log |T|$ such that for every pair of nodes (u, v) in S, $(2/3)^{rd}$ of the paths in $P_{u,v}$ have no corrupted nodes in them which implies that all pairs of nodes in S can communicate reliably with each other by invoking Π_{But}.*

Theorem 2 ([122]). *For sufficiently large n, there exists an n-node network $G_{Upfal} = (V, E)$, a protocol Π_{Upfal} for message transmission on it, and constants α_2 and β_2, such that:*

1. *The network G_{Upfal} is of constant degree*[2].

2. *The work complexity is Π_{Upfal} is $O(2^n)$.*

3. *Π_{Upfal} is $(\alpha_2 n, \beta_2 t)$-resilient.*

[2] G_{Upfal} is an n node Ramanjuan graph, and we know such graphs with large enough constant degree.

3.2 Our contributions

We introduce our main result:

Theorem (**Main Theorem for Communication Infrastructure**)**.** *For sufficiently large n, there exists an n-node network $G_{wc} = (V, E)$, a protocol $\Pi_{wc,eff}$ for message transmission on it, and constants α and β, such that:*

1. *The network G_{wc} is of degree $O(\log n)$.*
2. *The Work complexity of $\Pi_{wc,eff}$ is $O(\text{polylog}(n))$.*
3. *$\Pi_{wc,eff}$ is $(\alpha n, \beta t/\log n)$-resilient.*

Organization: Roadmap to our main result

The remainder of this chapter is organized as follows:

- We describe our construction for the randomized adversary model in Section 3.3.
- We describe our construction for the worst-case adversary model in Section 3.4.
- Our polylogarithmic work-efficiency protocol on the Butterfly Network is specified in Section 3.5.
- Our main result which combines resiliency in the face of worst-case corruptions with work-efficiency is described in Section 3.5.

3.3 Constant-degree Networks in the Random Model

In this section we will build a network that is resistant to linearly many random corruptions with an improved success probability as compared to Ben-or and Ron's work [18].

We turn our attention to the protocol of Chandran, Garay and Ostrovsky [35]. Their protocol builds on the following observation. Consider the protocols of [54] and [122] where if node A wishes to communicate with node B, A floods all paths from A to B (possibly of a bounded length) with the message. In [54], the parameters are set to ensure that a majority of such paths contain no corrupt nodes (for most pairs of nodes A, B) while [122] employs an exhaustive search to determine which paths may have contained corrupt nodes. These protocols face the disadvantage that paths that pass through even one corrupt node are lost. [35] introduced the idea of local correction through the use of Bracha committees. If we were able to create committees that had the ability to locally correct the message transmission, we can potentially tolerate a lot more corruptions than in [54] and perform the final decoding more efficiently than in [122]. [35] however considers many overlapping committees in order to ensure that even if a constant fraction of the nodes are corrupt, a sub-constant fraction of the committees are corrupt, where a committee is considered corrupt if a certain fraction of its nodes is corrupt. This calls for a larger degree. We show in this section that in our model of random corruptions, it suffices to construct fewer committees to achieve the same goal. Going forward, we refer to the networks (protocol, resp.) of [122] by G_{Upfal} (Π_{Upfal} resp.) respectively.

Let the set of nodes that wish to communicate be $V = [n]$ for $n \in \mathbb{N}$. We arbitrarily divide the nodes of V into n/s committees of size $s = (2/\alpha_2) \log \log n$ where α_2 is from Theorem 2. Within each committee, we instantiate G_{Upfal}, which is an expander of constant degree $d = O(1)$. We then connect the n/s committees using the network G_{But} from Theorem 1, where in order to connect two committees, we connect them by means of a perfect matching between the two sets of s nodes.

Protocoled-Network Construction 2 (protocoled-network for random corruption model).

Graph: *Our graph that is resistant to random errors is* $G_{rand} = (V, E)$*, where* $V = [n]$*. The edge set is as follows. Arbitrarily partition the nodes of* V *into* n/s *committees of size* $s = (2/\alpha_2) \log \log n$*. We let* C_v *denote the committee containing node* v*, where* $C_u = C_v$ *if* u *and* v *are in the same committee. Within each committee, we instantiate* G_{Upfal}*, which is an expander of constant degree* $d = O(1)$*. We then connect the* n/s *committees using the network* G_{But}*, where in order to connect two committees, we connect them by means of a perfect matching between the two sets of* s *nodes.*

Protocol: *We now describe the communication protocol* Π_{rand} *over this network. To this end, we first describe two building block protocols* Π_{edge} *and* Π_{maj}*.*

- Π_{edge} *is the protocol that is invoked when we wish to send a message from one committee,* C *to another* C' *that are connected in the* G_{But} *network (connected by means of a perfect matching). We will assume that each node in* C *is initialized with some message. In the protocol* Π_{edge}*, each node in* C *sends its message to the node it is matched to in* C'*.*
- Π_{maj} *is a majority protocol invoked within a committee* C*. We will assume that each node* i *in* C *is initialized with some message* m_i*. The goal of the* Π_{maj} *protocol is for each node in* C *to compute the majority function* $m = \text{maj}\{m_i\}_i$*. The protocol proceeds as follows: every node in* C *invokes* Π_{Upfal} *to send its message to every other node in* C*. Each node then simply computes (locally) the majority of the messages it received.*

Now, if a node A *wishes to send a message* m *to node* B*:*

(a) If A *and* B *are in the same committee* C*, then* A *simply sends the message to* B *by invoking* Π_{Upfal} *within the committee* C*.*

(b) If A *and* B *are in different committees,* C_A *and* C_B *respectively, then:*

i. A invokes Π_{Upfal} to send m to every other node in its committee C_A.

ii. The committee C_A then invokes Π_{But} to send a message to the committee C_B. In the invocation of Π_{But}, whenever two committees C and C' connected by G_{But} wish to communicate with each other, they invoke Π_{edge} and then C' invokes Π_{maj}.

iii. Finally, every node other than B in committee C_B invokes Π_{Upfal} to send the message they received to B. B computes (locally) the majority of the messages it received.

We set out to prove the following theorem.

Theorem 3. *For sufficiently large n, there exists an n-node network $G_{rand} = (V, E)$, a protocol Π_{rand} for message transmission on it, and constants α_3 and β_3, such that:*

1. *The network G_{rand} is of constant degree.*

2. *If a subset of nodes $T \subset V$ is randomly corrupt, where $|T| \leq \alpha_3 n$, with probability $1 - (t/n)^{\alpha_2 t/(4\log(n))}$, there exists a set of nodes $S \subset V$ where $|S| \geq n - \beta_3|T|$ such that every pair of nodes in S can communicate reliably with each other by invoking Π_{rand}.*

Note that at $t = \Theta(n)$ we get that the protocol works with probability $1 - 2^{-\Omega\left(\frac{n}{\log(n)}\right)}$ which improves upon [18] which achieved $1 - 2^{-\Omega\left(\frac{n}{\log^2(n)}\right)}$.

We now prove Theorem 3.

Part 1 of Theorem 3: Degree. The network constructed is of constant degree, namely $D = d + 11$.

Part 2 of Theorem 3: Resilience. We now wish to argue that in the presence of a set $T \subset V$ of randomly corrupt nodes with $|T| \leq \alpha_3 n$, there exists a set $S \subset V$ with $|S| \geq n - \beta_3|T|$ such that every pair of nodes in S can communicate reliably

with each other, for appropriately chosen universal constants α_3, β_3 to be determined later. The proof proceeds as follows. Under these choices of α_3, β_3, we first show that most committees must in fact contain less than an α_2-fraction of corrupt nodes. In such committees, Π_{Upfal} works successfully for all but an $\epsilon = O(\alpha_2)$-fraction of nodes in that committee by Theorem 2. Call such committees as *good* committees. From Theorem 2, in good committees there exists a set of *privileged* nodes of size at least $s - O(\alpha_2 s)$ that can communicate reliably with each other.

We now consider nodes A, B that wish to communicate with each other, and are *privileged* nodes in *good* committees. Hence, all but an ϵ-fraction of the nodes in C_A (the committee containing A) receive A's message correctly on executing Π_{Upfal}. On any execution of Π_{edge} between C_A and another committee C', all but at most an ϵ-fraction of the nodes in C' receive the correct value. Now, if C' is *good*, in the execution of the Π_{maj} protocol in C', all but at most a $\epsilon + \alpha_2 = O(\alpha_2)$-fraction of the nodes begin with the correct value and Π_{Upfal} works successfully for all but an ϵ-fraction of nodes. This ensures that as long as $\epsilon + \alpha_2 < 1/2$, all but at most an ϵ-fraction of the nodes compute the majority of the incoming messages correctly. Inductively, this would show that at the end of the emulation of the Π_{But} protocol, all but an ϵ-fraction of the nodes in the committee containing B receive A's message correctly and since C_B is a good committee and $\epsilon + \alpha_2 < 1/2$, B receives A's message correctly as B is *privileged*.

We now formalize this argument. We call a committee *good* if the fraction of corrupt nodes in it is at most α_2 and *bad* otherwise. Let $T \subset V$ be a set of randomly corrupt nodes with $|T| = t = \alpha_3 n$ where $\alpha_3 \leq \min\{\alpha_1, (\alpha_2/e)^2\}$ where the constant α_2 is from Theorem 2.

Lemma 6. *The probability that a committee is* good *is at least* $1 - (t/n)^{\log \log n}$.

Proof. The probability that a committee is bad is

$$\begin{aligned}\Pr[\text{A committee is bad}] &\leq \binom{s}{\alpha_2 s}\left(\frac{t}{n}\right)^{\alpha_2 s} \\ &\leq \left(\frac{es}{\alpha_2 s}\right)^{\alpha_2 s}\left(\frac{t}{n}\right)^{\alpha_2 s} \\ &\leq \left(\frac{et}{\alpha_2 n}\right)^{\alpha_2 s}\end{aligned}$$

Taking $\alpha_3 \leq (\alpha_2/e)^2$ which implies $\frac{e\alpha_3}{\alpha_2} \leq \sqrt{\alpha_3}$, i.e., $\frac{et}{\alpha_2 n} \leq \sqrt{\frac{t}{n}}$, we get

$$\begin{aligned}\Pr[\text{A committee is bad}] &\leq \left(\frac{t}{n}\right)^{\frac{\alpha_2 s}{2}} \\ &\leq \left(\frac{t}{n}\right)^{\log\log(n)}\end{aligned}$$

as $s = (2/\alpha_2)\log\log(n)$. □

Lemma 7. *The number of* bad *committees is at most* $\frac{t/s}{\log(n)}$ *with probability at least* $1 - (t/n)^{\alpha_2 t/(4\log(n))}$.

Proof. Let $\zeta = (t/n)^{\log\log n}$. Note that

$$\begin{aligned}1/\sqrt{\zeta} &= (t/n)^{-\log\log n/2} \\ &= (1/\alpha_3)^{\log\log n/2} \\ &= (1/\alpha_3)(1/\alpha_3)^{\log\log n/2-1} \\ &\geq (1/\alpha_3)\cdot(8)^{\log\log n/2-1} \\ &= (1/\alpha_3)\cdot 2^{1.5\log\log n-3} \\ &\gg (1/\alpha_3)\cdot 2^{\log\log n+\log_2(e)} \\ &= (1/\alpha_3)\cdot e\log(n) = en\log(n)/t \qquad (3.1)\end{aligned}$$

The probability that the number of bad committees is more than $\frac{t/s}{\log(n)}$ is

$$
\begin{aligned}
&\leq \binom{n/s}{t/(s\log(n))} \zeta^{t/(s\log(n))} \\
&\leq \left(\frac{en/s}{t/(s\log(n))}\right)^{t/(s\log(n))} \zeta^{t/(s\log(n))} \\
&= \left(\zeta \cdot \frac{en\log(n)}{t}\right)^{t/(s\log(n))} \\
&\leq \left(\sqrt{\zeta}\right)^{t/(s\log(n))} \qquad \text{from (3.1)} \\
&= \left(\frac{t}{n}\right)^{\frac{\log\log(n)t}{2s\log(n)}} \\
&= \left(\frac{t}{n}\right)^{\frac{\alpha_2 t}{4\log(n)}} \\
&= (t/n)^{\alpha_2 t/(4\log(n))}
\end{aligned}
$$

□

We have that if C is a *good* committee with $t' \leq \alpha_2 s$ corrupt nodes, from Theorem 2, there exists a set S_C (privileged nodes) of at least $s - \beta_2 t'$ nodes in C that can communicate reliably with each other. We say that a committee holds value v if all the privileged nodes in the committee hold value v.

Lemma 8. *If C and C' are* good *committees connected by an edge in G_{But} and if C holds value v, after invoking Π_{edge} and Π_{maj}, C' holds value v.*

Proof. Since C holds value v, at least $s - \beta_2\alpha_2 s$ nodes in C' receive the value v after invoking Π_{edge}. Since C' is *good* at most $\alpha_2 s$ nodes in C' are corrupt. Hence, at least $s - (\beta_2 + 1)\alpha_2 s$ nodes in C' begin with the value v while invoking Π_{maj} in C'. Consider a node Z in the set $S_{C'}$ of privileged nodes in C'. As C' is good, we have $|S_{C'}| \geq s - \beta_2\alpha_2 s$. Nodes in $S_{C'}$ receive messages reliably from each other. Out of the messages received by Z from nodes in $S_{C'}$ during the execution of Π_{maj}, at most $(\beta_2 + 1)\alpha_2 s$ may be unequal to v. The messages received by Z from the $\beta_2\alpha_2 s$ non-privileged nodes may not be equal to v. Still each node in $S_{C'}$ will receive at least $s - (2\beta_2 + 1)\alpha_2 s$ copies of v. Hence, if $(2\beta_2 + 1)\alpha_2 < 1/2$, the claim follows.

We note from [122] that it is possible to take $\alpha_2 = 1/72$ and $\beta_2 = 6$ which satisfies $(2\beta_2 + 1)\alpha_2 < 1/2$. □

Considering the *bad* committees as corrupt nodes in G_{But}, there are at most $\frac{t/s}{\log n}$ of them with overwhelming probability by Lemma 7. From Theorem 1, there exists a set of committees P (privileged committees) that can communicate with each other reliably.

Lemma 9. *Let A and B be two nodes in privileged (good) committees $C_A \in P$ and $C_B \in P$ respectively. If $A \in S_{C_A}$ and $B \in S_{C_B}$, then the above protocol guarantees reliable message transmission from A to B.*

Proof. Note that if $C_A = C_B$, we are done by Theorem 2 as A and B are privileged. We consider the case $C_A \neq C_B$. Since $A \in S_{C_A}$, all nodes in S_{C_A} receive A's message, m, correctly and C_A holds m. Since $C_A, C_B \in P$, after the invocation of Π_{But}, C_B holds m. Since $B \in S_{C_B}$, it receives m from each node in $S_{C'}$. Hence B will receive at least $s - \beta_2\alpha_2 s$ copies of v. If $\beta_2\alpha_2 < 1/2$, the claim follows. We note from [122] that it is possible to take $\alpha_2 = 1/72$ and $\beta_2 = 6$ which satisfies $\beta_2\alpha_2 < 1/2$. □

Lemma 10. *With probability $1 - (t/n)^{\alpha_2 t/(4\log(n))}$, there exists a set of nodes $S \subset V$ where $|S| \geq n - \beta_3|T|$ such that every pair of nodes in S can communicate reliably with each other.*

Proof. The set S consists of nodes that are privileged nodes in privileged committees. We have that the total number of committees is $N_C = n/s$. Let t_C denote the number of bad committees. Note that with probability at least $1-(t/n)^{\alpha_2 t/(4\log(n))}$, $t_C \leq \frac{t/s}{\log n}$. Furthermore, since $t = \alpha_3 n \leq \alpha_1 n$ (by the choice of $\alpha_3 \leq \min\{\alpha_1, (\alpha_2/e)^2\}$), $t_C \leq \frac{t/s}{\log n} \leq \alpha_1 \cdot \frac{n/s}{\log n} \leq \alpha_1 \cdot \frac{N_C}{\log N_C}$. This implies that Theorem 1 is now applicable. From Theorem 1, the number of unprivileged committees is bounded by $O(t_C \log t_C) = O(t/s)$. Thus, the number of nodes in unprivileged committees is bounded by $s \cdot O(t/s) = O(t)$. Finally, we consider the unprivileged nodes in privileged committees. Let t_i denote the number of corrupt nodes in committee C_i for $i \in [n/s]$. The number

of unprivileged nodes in privileged committees is upper bounded by

$$\sum_i O(t_i) = O\left(\sum_i t_i\right) = O(t)$$

from Theorem 2. Thus, $|S| \geq n - \beta_3 t$ for some constant β_3. □

This completes the proof of Theorem 3. We end this section with the following remark. Let $|T| = t$. Note that in [35], the number of nodes that can communicate with each other reliably is $n - t - O(t/\log n)$, that is, we give up at most $O(t/\log n) = o(t)$ nodes. We remark that this is not achievable in networks of constant degree even in the random model. In an adversarial corruption setting, one can corrupt the neighbors of $O(t/d)$ nodes, and hence if $d = O(1)$, any protocol must give up $O(t)$ nodes. This is true even in the random corruption model: a node has corrupt neighbors with some constant probability if $t = O(n)$ and hence any protocol must give up $O(t)$ nodes. Similarly, in networks of $\log \log n$ degree, any protocol must give up $O(t/(\log n)^{\Theta(1)})$ nodes.

3.4 Logarithmic degree Networks in the Worst-case Model

In the worst-case model, the current best networks are those constructed by [35]. They construct a graph with degree $d = \log^q n$ for some fixed constant $q > 1$, that is resilient to $t = O(n)$ adversarial corruptions. We show using a probabilistic argument the existence of a network of degree $O(\log n)$ that is resilient to $t = O(n)$ adversarial corruptions. Furthermore, the probabilistic construction works with all but negligibly small probability.

Our construction is also rather simple. and uses our network that is resilient to random errors as a black box. This style of our argument provides further motivation for studying the random corruption model, even if the ultimate goal is to be resilient to adversarial corruptions.

Protocoled-Network Construction 3 (protocoled-network for worst-case corruption model).

> **Graph**: *Our graph that is resistant to worst-case errors is $G_{wc} = (V, E)$, where $V = [n]$. The edge set is as follows. Let $\{G^i_{rand}\}_i = \{(V_i^R, E_i)\}_i$ be our network, G_{rand}, resilient to random corruptions on a randomly permutation V_i^R of the vertex set V, for $1 \leq i \leq z \triangleq k \cdot \log n$ for $k = 40/\alpha_2$, where α_2 is the constant from Theorem 2. Define $E \triangleq \bigcup_{i=1}^{z} E_i$.*
>
> **Protocol**: *We now describe the communication protocol Π_{wc} over this network. Let Π^i_{rand} be the reliable transmission protocol associated with the network G^i_{rand} as described in Definition 2, for each $1 \leq i \leq z$. Now, if a node A wishes to send a message m to node B:*
>
> *(a) A will invoke the protocol Π^i_{rand} to transmit the message m to B over the network G^i_{rand}.*
>
> *(b) B receives z messages, corresponding to the z executions of Π^i_{rand} for $1 \leq i \leq z$. B takes the majority of all these messages.*

We set out to prove the following theorem.

Theorem 4. *For sufficiently large n, there exists an n-node network $G_{wc} = (V, E)$, a protocol Π_{wc} for message transmission on it, and constants α_4 and β_4, such that:*

1. *The network G_{wc} is of degree $O(\log n)$.*

2. *Π_{wc} is $(\alpha_4 n, \beta_4 t/\log n)$-resilient.*

Part 1 of Theorem 4: Degree. The network constructed is of degree $O(\log n)$, since the network is constructed using $z = O(\log n)$ copies of the constant degree network G_{rand} from Definition 2.

Part 2 of Theorem 4: Resilience. We proceed to prove resiliency of the protocol. We will first consider an arbitrary fixed adversary $T \subset V$, estimate the probability of resilience against it and finally perform a union bound over all adversaries. Consider an arbitrary fixed adversary. We will say that the i^{th} layer is bad for this fixed adversary if the conditions in Theorem 3 do not hold for G^i_{rand}. Correspondingly we call a layer good for this adversary if the conditions in Theorem 3 hold. In Lemma 11, we prove that with high probability only at most (1/5)th of the layers are bad.

Consider a good layer i, for some $1 \leq i \leq z$. We define D_i to be set of doomed nodes in protocol Π^i_{rand}. By Theorem 3, $|D_i| \leq \beta_3 |T|$. For an arbitrary fixed adversary, we will show that the set D_i behaves as a small random set as a result of permuting the vertex set V to obtain V^R_i over which G^i_{rand} is constructed. For any honest node $v \in V$, let L^D_v denote the set of all good layers i such that $v \in D_i$, that is, v is doomed in layer i. We will finally show that, with high probability, for most nodes v, $|L^D_v|$ is small.

To wrap up the proof, we designate a node $v \in V$ as doomed for Π_{wc} with respect to this fixed adversary if $|L^D_v| > (1/10)z$. Consider a pair of privileged nodes (nodes that are honest and not designated as doomed for Π_{wc}) $A, B \in V$. Since, with high probability, at most (1/5)th of the layers are bad and, by definition, A, B are doomed

in at most (1/10)th of the good layers, A, B are both privileged in at least (3/5)th of the good layers with respect to this adversary. Hence a majority of the messages sent by A in Π_{wc} reach B correctly and B's majority is computed correctly. By our earlier claim, with high probability, the number of doomed nodes is small, that is, most nodes are privileged and can hence communicate reliably in the presence of this fixed adversary with high probability. Performing a union bound over all possible adversaries, we get our final result.

We now formalize this argument. Let $T \subset V$ be an arbitrary set of corrupt nodes with $|T| = t = \alpha_4 n$ where $\alpha_4 \leq \min\{\alpha_3, 1/10, \frac{1}{11^{4/3} e^{3/2} \beta_3^{4/3}} \approx \frac{0.01}{\beta_3^{4/3}}\}$ where the constant α_3 is from Theorem 3.

Lemma 11. *For a fixed adversary, with probability at least* $1 - n^{\frac{32}{\alpha_2}} \cdot \left(\frac{n}{t}\right)^{-2t}$*, at most* $\delta = \frac{1}{5}$ *fraction of the layers are bad.*

Proof. Note that the ith layer is constructing by randomly and independently permuting the vertex set V to obtain V_i^R over which G_{rand}^i is constructed. This is equivalent to constructing G_{rand}^i over V and thinking of the adversary as being a random subset of V of size $|T|$. This enables to apply Theorem 3. By Theorem 3, for a fixed adversary, the ith layer is bad independently with probability $\leq (t/n)^{\alpha_2 t/(4 \log n)}$. So the

probability that δz out of the z layers are bad is

$$\begin{aligned}
\Pr[\delta z \text{ out of the } z \text{ layers are bad}] &\leq \binom{z}{\delta z}\left(\left(\frac{t}{n}\right)^{\alpha_2 t/(4\log n)}\right)^{\delta z} \\
&\leq \left(\frac{ez}{\delta z}\right)^{\delta z}\left(\left(\frac{t}{n}\right)^{\alpha_2 t/(4\log n)}\right)^{\delta k \log n} \\
&= \left(\frac{e}{\delta}\right)^{\delta k \log n}\left(\left(\frac{t}{n}\right)^{\alpha_2 t/4}\right)^{\frac{8}{\alpha_2}} \\
&= (5e)^{(8/\alpha_2)\cdot\log n}\left(\frac{t}{n}\right)^{2t} \\
&= (5e)^{(8/\alpha_2)\cdot\log n}\left(\frac{n}{t}\right)^{-2t} \\
&= n^{(8/\alpha_2)\cdot\log(5e)}\left(\frac{n}{t}\right)^{-2t} \\
&\leq n^{32/\alpha_2}\cdot\left(\frac{n}{t}\right)^{-2t}
\end{aligned}$$

□

Lemma 12. *For a fixed adversary and a fixed layer i, the probability that $D_i = S \subset V \setminus T$ only depends on $|S|$.*

Proof. Consider a fixed adversary and a fixed layer i. Let π_i be a permutation of V and let $\pi_i(V) = V_i^R$. Also, let $\pi_i(T) = T_i^R$. Let $D_i^R \subset V_i^R \setminus T_i^R$ be the doomed nodes in V_i^R with respect to this adversary. Note that D_i^R is fixed by the choice of T_i^R, or equivalently, by the choice of $\pi_i(T)$. Let $D_i \subset V \setminus T$ be the set of doomed nodes in V. Note that $\pi_i(D_i) = D_i^R$. By symmetry, for any two subsets $S_1, S_2 \subset V \setminus T$ with $|S_1| = |S_2| = |D_i^R|$, the number of permutations π such that:

- $\pi(T) = T_i^R$ and $\pi(S_1) = D_i^R$
- $\pi(T) = T_i^R$ and $\pi(S_2) = D_i^R$

is the same, and is equal to the number of permutations of the remaining $|V| - |T| - |D_i^R|$ nodes. Hence, the probability that $D_i = S \subset V \setminus T$ only depends on $|S|$. □

For a fixed adversary and a fixed honest node $v \in V \setminus T$, let L_v^D denote the set of all good layers i such that $v \in D_i$, that is, v is doomed in layer i.

Lemma 13. *For a fixed adversary, with probability* $1-\left(\frac{11e\beta_3 t}{n}\right)^{8t}$*, the number of honest nodes* v *such that* $|L_v^D| \geq z/10$ *is at most* $\beta_4 t/\log n$ *(*$\beta_4 = 2\alpha_2$*).*

Proof. Let layer i be good. This implies that $|D_i| \leq \beta_3 t$ by Theorem 3. Without loss of generality we can assume that $|D_i| = \beta_3 t$ as more doomed nodes is worse for us. Let v be an arbitrary honest node. By Lemma 12,

$$\Pr[v \in D_i] = \frac{\beta_3 t}{n-t}$$

as all subsets of honest nodes of size $\beta_3 t$ are equally likely and the number of honest nodes is $n-t$.

As all layers are sampled independently, we have

$$\begin{aligned}
\Pr[|L_v^D| \geq z/10] &\leq \binom{z}{z/10}\left(\frac{\beta_3 t}{n-t}\right)^{z/10} \\
&\leq \left(\frac{ez}{z/10}\right)^{z/10}\left(\frac{\beta_3 t}{n-t}\right)^{z/10} \\
&\leq \left(\frac{10e\beta_3 t}{n-t}\right)^{z/10} \\
&\leq \left(\frac{11e\beta_3 t}{n}\right)^{z/10}
\end{aligned}$$

where the last inequality follows from $t \leq n/10$.

Let u, v be two honest nodes. We have that $\Pr[v \in D_i] = \frac{\beta_3 t}{n-t}$, while $\Pr[u \in D_i | v \in D_i] = \frac{\beta_3 t - 1}{n-t-1} < \frac{\beta_3 t}{n-t}$. Hence the events $u \in D_i$ and $v \in D_i$ are anti-correlated. This implies that $|L_v^D| \geq z/10$ and $|L_u^D| \geq z/10$ are also anti-correlated. As we want to upper bound the number of nodes A which satisfy $|L_A^D| \geq z/10$, we can assume that the events $|L_v^D| \geq z/10$ and $|L_u^D| \geq z/10$ are independent.

The probability that for more than $\beta_4 t/\log n$ honest nodes $|L_v^D| \geq z/10$ with $\beta_4 =$

$\alpha_2/2$ is

$$\begin{aligned}\Pr\left[\text{For more than } t/\log n \text{ honest nodes, } |L_v^D| \geq z/10\right] &\leq \left(\left(\frac{11e\beta_3 t}{n}\right)^{z/10}\right)^{\beta_4 t/\log n} \\ &= \left(\frac{11e\beta_3 t}{n}\right)^{k\beta_4 t/10} \\ &= \left(\frac{11e\beta_3 t}{n}\right)^{4\beta_4 t/\alpha_2} \\ &= \left(\frac{11e\beta_3 t}{n}\right)^{8t}\end{aligned}$$

□

We now prove Theorem 4.

Proof of Theorem 4. For a fixed adversary $\mathcal{A}$, let $E_1^{\mathcal{A}}$ be the event that less than $z/5$ of the layers are bad. Then by Lemma 11, $\Pr[E_1^{\mathcal{A}}] \geq 1 - n^{\frac{32}{\alpha_2}} \cdot \left(\frac{n}{t}\right)^{-2t}$. Let E_1 be the event that $E_1^{\mathcal{A}}$ holds for all adversaries $\mathcal{A}$ with t corruptions. By a union bound over all such adversaries,

$$\begin{aligned}\Pr[E_1] &\geq 1 - \binom{n}{t} \cdot n^{\frac{32}{\alpha_2}} \cdot \left(\frac{n}{t}\right)^{-2t} \\ &\geq 1 - \left(\frac{en}{t}\right)^t \cdot \left(\frac{n}{t}\right)^{-2t} \cdot n^{\frac{32}{\alpha_2}} \\ &= 1 - \left(\frac{et}{n}\right)^t \cdot n^{\frac{32}{\alpha_2}} \\ &\geq 1 - \left(\frac{t}{n}\right)^{.5t} \cdot n^{\frac{32}{\alpha_2}} \qquad \text{As } t = n/10 \text{ which implies } e \leq \sqrt{\frac{n}{t}} \\ &\geq 1 - 1/n^{\omega(1)} \qquad [\text{For } t = \omega(1)]^3\end{aligned}$$

Let $E_2^{\mathcal{A}}$ be the event that the number of honest nodes v such that $|L_v^D| \geq z/10$ is at most $\beta_4 t/\log n$. Then by Lemma 13 for a fixed adversary$\Pr[E_2^{\mathcal{A}}] \geq 1 - \left(\frac{11e\beta_3 t}{n}\right)^{8t}$. Let E_2 be the event that $E_2^{\mathcal{A}}$ holds for all adversaries $\mathcal{A}$ with t corruptions. By a

union bound over all such adversaries,

$$\begin{aligned}
\Pr[E_2] &\geq 1 - \binom{n}{t}\left(\frac{11e\beta_3 t}{n}\right)^{8t} \\
&\geq 1 - \left(\frac{en}{t}\right)^{t} \cdot \left(\frac{11e\beta_3 t}{n}\right)^{8t} \\
&\geq 1 - \left(\frac{11^8 \cdot e^9 \cdot \beta_3^8 \cdot t^7}{n^7}\right)^{t} \\
&\geq 1 - \left(\frac{t}{n}\right)^{t} \qquad \text{As } t \leq \frac{n}{11^{4/3} e^{3/2} \beta_3^{4/3}} \\
&\geq 1 - 1/n^{\omega(1)} \qquad [\text{For } t = \omega(1)]
\end{aligned}$$

Hence by union bound $\Pr[E_1 \wedge E_2] \geq 1 - 1/n^{\omega(1)} - 1/n^{\omega(1)} = 1 - 1/n^{\omega(1)}$.

E_1 implies that for any adversary $\leq 1/5$ fraction of the layers are bad. E_2 implies that for any adversary there exists a set of honest nodes $S, |S| \geq n - t - \beta_4 t \log n$ such that for all $v \in S$, $|L_v^D| \leq z/10$. Hence for any two nodes $A, B \in S$ they are both privileged in at least $1 - 1/5 - 1/10 - 1/10 > 1/2$ fraction of the layers. Hence the message from A to B will be correctly delivered on $> 1/2$ fraction of the layers hence B will find the correct message after taking majority. The set S behaves as the privileged set for the network G_{wc}, Π_{wc}. □

3.5 Low-work Protocols in the Worst-case Model

In this section, we will design low degree graphs with efficient communication protocols for AE reliable message transmission. Our final networks are constructed by composing several simpler graph structures. An important graph that our work builds on is [54]'s *butterfly* network. The diameter of a graph is a fundamental lower bound on the amount of work required for message transmission. Any graph with constant degree will necessarily have work complexity $\Omega(\log n)$. Thus, the logarithmic diameter of the butterfly network is optimal up to constant factors. Since the diameter is a fundamental lower bound on the work complexity of point to point transmissions in a network, we think of a polynomial work complexity in the diameter—polylogarithmic work complexity—as a reasonable definition for *work-efficient* in this context. [54]'s protocol which requires $\Omega(n)$ work complexity for a single point to point message transmission is thereby work-inefficient. Another weakness of [54]'s protocol, is that it floods the network, and thus nearly every node in the network is necessarily involved in every point to point message transmission. It would aid both efficiency and parallelizability of higher level protocols to significanlty limit the number of nodes used for a point to point transmission.

We make simple modifications to [54]'s ideas to achieve a work-efficient protocol that requires only polylogarithmically many nodes to be active in any point to point communication in this section. Our main observation is that a u to v transmission over the Butterfly network need not flood all $\Theta(n/\log n)$ paths in the network to ensure reliable transmission. In fact, we show that picking a set of just $\Theta(\log n)$ paths between every pair of vertices, and sending the message only over those paths suffices. This reduces both the number of nodes used per point to point transmission and total work to $O(\log^2 n)$.

Protocoled-Network Construction 4 (work-efficient Butterfly protocoled-network). *The* efficient Butterfly *protocoled-network* $N_{Eff} = (G_{But}, \Pi_{Eff})$ *is as follows:*

> **Graph***: We use the Butterfly graph* $G_{But} = (V, E)$ *as defined in Definition 1 such that* $|V| = n = m2^m$*. For every pair* u, v *of distinct vertices in* V*, there*

exists a set of paths $P_{u,v}$ as defined in Definition 1 between u and v. Let $Q_{u,v}$ be a random subset of $P_{u,v}$ of size $\Theta(\log n)$. The subset $Q_{u,v}$ is sampled before the protocol and is fixed, in particular it is known to all the nodes as well as the adversary.

Protocol*: The message transmission protocol Π_{Eff} from u to v in G_{Eff} is as follows: u sends the message along all paths in $Q_{u,v}$, v receives all the messages and takes majority.*

We set out to prove the following theorem.

Theorem 5. *For the $n = m2^m$-node network $G_{But} = (V, E)$ there is a protocol Π^*_{Eff} for message transmission on it such that the following holds:*

1. *The network G_{But} has degree 11.*
2. *The total work of the protocol is $O(\text{polylog}(n))$.*
3. *There is a constant $\varepsilon \in (0, 1)$ such that Π^*_{Eff} is $(\varepsilon n/\log n, O(t \log t))$-resilient.*

In proving Theorem 5, we will need the following lemma.

Lemma 14. *For the $n = m2^m$-node network $G_{But} = (V, E)$ and the protocol Π_{Eff} for message transmission on it the following statements hold:*

1. *The network G_{But} has degree 11.*
2. *The total work is $O(\text{polylog}(n))$.*
3. *There is a constant $\varepsilon \in (0, 1)$ such that Π_{Eff} is $(\varepsilon n/\log n, O(t \log t))$-resilient with probability $1 - o(1)$.*

Proof. It is clear that the degree of the network is 11 and that the work complexity in the protocol are $O(\text{polylog}(n))$ as we send $\Theta(\log n)$ messages on paths of length $\Theta(\log n)$.

We now prove the resilience guarantee. Consider any fixed subset $T \subset V$ with $t = |T| \leq \alpha_1 n/\log n$, where α_1 is that of Theorem 1. By Theorem 1, we know that

there is a set V' of size $n - \beta_1 t \log t$ that can communicate reliably with each other by invoking Π_{But}. For any pair of vertices $u, v \in V'$, we let $P_{u,v}$ be the set of paths used in message transmissions from u to v by protocol Π_{But}. By Theorem 1 property (3) we know that at least a 2/3 fraction of the paths in each $P_{u,v}$ contain no corrupt node. We will assume that exactly 2/3 fraction of the paths in each $P_{u,v}$ contain no corrupt node as that is only worse for us. If a message is sent through a path with no corrupt nodes, the correct message reaches v. Let $Q_{u,v}$ be a random sample of $h = 144 \log_e(n) \approx 100 \log n$ paths from $P_{u,v}$. The protocol Π_{Eff} sends a message from u to v as follows:

1. u sends the message along all the paths $Q_{u,v}$,

2. v receives all h messages that were sent along the paths in $Q_{u,v}$ and takes the majority.

We now argue when this majority will be the correct message with high probability. Fix two nodes $u, v \in V'$ and fix an adversary (the subset of corrupted nodes). We look at the paths $Q_{u,v}$ in a communication from u to v. The expected number of paths, μ, in $Q_{u,v}$ with a corrupted node is $\mu = h/3$. So, we define $\delta = \frac{1/2}{1/3} - 1$ and by the Chernoff bound from Equation 2.1, the probability that a majority of the paths $Q_{u,v}$ contain a corrupt node is:

$$\begin{aligned}
\Pr[\text{majority of paths } Q_{u,v} \text{ are incorrect}] &\leq e^{-\delta^2 \mu/3} \\
&\leq e^{-(1/2)^2 \cdot (h/9)} \\
&\qquad [\text{As } \delta = \frac{1/2}{1/3} - 1 \text{ and } \mu = h/3] \\
&= e^{-h/36} \\
&= e^{-4 \log_e(n)} \\
&= 1/n^4
\end{aligned}$$

We call a pair of vertices $\{u, v\}$ a *doomed-pair* if a majority of paths between them contain a corrupt node. For a fixed adversary, the probability that there are more

than g doomed-pairs is bounded above by

$$\binom{n^2}{g}\left(\frac{1}{n^4}\right)^g \leq \left(\frac{en^2}{g}\right)^g \left(\frac{1}{n^4}\right)^g < \left(\frac{e}{n^2}\right)^g$$

since the probability of pair corruptions is independent conditioned on the adversary. To show that the construction works for an adversarially chosen set of corruptions, we take a union bound over all adversaries. The probability that there is an adversary with t corruptions for which the number of doomed-pairs is at least g is bounded above by:

$$\binom{n}{t}\left(\frac{e}{n^2}\right)^g$$

Setting $g = t$ we get $\binom{n}{t}\left(\frac{e}{n^2}\right)^g \leq \left(\frac{en}{t}\right)^t \left(\frac{e}{n^2}\right)^t \leq \left(\frac{e^2}{n}\right)^t = o(1)$. Hence the number of doomed-pairs $u, v \in V'$ is $\leq t$ with probability $1 - o(1)$. Let S be the set of vertices $v \in V'$ which are not in any doomed-pair. The set S is privileged for N_{eff} as for any $A, B \in S$ the majority of paths $Q_{A,B}$ have no corrupt nodes and hence B decodes the correct message by taking majority. As with probability $1 - o(1)$ the number of doomed-pairs is $\leq t$ which implies that number of nodes in any doomed-pair is $\leq 2t$. Hence $|S| \geq |V'| - 2t$ with probability $1 - o(1)$. By Theorem 1, $|V'| \geq n - \beta_1 t \log t$ hence $|S| \geq n - (\beta_1 + 2)t \log t$ with probability $1 - o(1)$ which implies that the number of doomed nodes is $O(t \log t)$. □

Lemma 14 shows that the network $N_{Eff} = (G_{But}, \Pi_{Eff})$ satisfies resilience only with high probability - not deterministically. We now show, via the probabilistic method, that the resilience guarantee can be made deterministic; yet we state it explicitly, because this is the protocol that we use to enable our main theorem.

Proof of Theorem 5. Since Lemma 14 holds with probability greater than 0, and the randomness is just over the protocol Π_{Eff}, there is some specific protocol Π^*_{Eff} in the support of Π_{Eff} that has properties (1-3). □

3.5.1 Resilient and Efficient Networks

We now show how to modularly substitute-in our work-efficient protocol on the Butterfly network, in order to get work-efficient versions of Theorems 3 and 4. The high level idea is simple. The protocol Π_{rand} uses Π_{But} as a blackbox, and the protocol Π_{wc} uses a Π_{rand} as a blackbox. Substituting the efficient protocol Π^*_{Eff} in for Π_{But} into Π_{rand} creates an efficient version of Π_{rand}, which we call $\Pi_{rand,eff}$ below; and substituting $\Pi_{rand,eff}$ into Π_{wc} creates an efficient version of Π_{wc}, which we call $\Pi_{wc,eff}$ below. We describe the details of these subsitutions in the following theorems.

We substitute Π^*_{Eff} in for Π_{But} in Π_{rand} to strengthen Theorem 3 to the following:

Theorem 6. *For sufficiently large n, there exists an n-node network $G_{rand} = (V, E)$, a protocol $\Pi_{rand,eff}$ for message transmission on it, and constants α_3 and β_3, such that:*

1. *The network $G_{rand,eff}$ is of constant degree.*
2. *The Work complexity of $\Pi_{rand,eff}$ is $O(\text{polylog}(n))$.*
3. *If a subset of nodes $T \subset V$ is randomly corrupt, where $|T| \leq \alpha_3 n$, with probability $1 - (t/n)^{\alpha_2 t/(4\log(n))}$, there exists a set of nodes $S \subset V$ where $|S| \geq n - \beta_3|T|$ such that every pair of nodes in S can communicate reliably with each other by invoking $\Pi_{rand,eff}$.*

Proof. In Theorem 3 we note that work done inside a single committee was exponential in the size of the committee as we instantiate G_{Upfal} (from Theorem 2) inside every committee. But as the size of the committee is $s = O(\log\log(n))$ this is only $O(\text{polylog}(n))$. Thinking of committees as super-nodes we had instantiated G_{But} over super-nodes. The total number of super-nodes used in a single message transmission was $\Omega(n/s)$ as we have n/s super-nodes. By using Π_{eff} instead of Π_{But} we can bring this down to $\text{polylog}(n/s) \leq \text{polylog}(n)$, Sending a single message from a super-node to its neighbor requires running G_{Upfal} inside the committee which takes $O(\text{polylog}(n))$ work. Thus the total work is $O(\text{polylog}(n) \cdot \text{polylog}(n)) = O(\text{polylog}(n))$. □

Finally, we substitute $\Pi^*_{rand,eff}$ in for Π_{rand} in Π_{wc} to strengthen Theorem 4 to our main theorem:

Theorem 7. *For sufficiently large n, there exists an n-node network $G_{wc} = (V, E)$, a protocol $\Pi_{wc,eff}$ for message transmission on it, and constants α and β, such that:*

1. *The network G_{wc} is of degree $O(\log n)$.*
2. *The Work complexity of $\Pi_{wc,eff}$ is $O(\text{polylog}(n))$.*
3. *$\Pi_{wc,eff}$ is $(\alpha n, \beta t/\log n)$-resilient.*

Proof. The protocol Π_{wc} uses Π_{rand} as a blackbox $O(\log n)$ times, one for each layer. We substitute Π_{rand} with $\Pi_{rand,eff}$. This brings down the work complexity to $O(\log n \cdot \text{polylog}n) = \text{polylog}n$. □

Chapter 4

Infrastructure for Secure Computation

We propose an infrastructure for secure multiparty computation (MPC) based on oblivious transfer (OT), which would consist of OT channels between some pairs of parties in the network. We devise information-theoretically secure protocols that allow additional pairs of parties to establish secure OT correlations using the help of other parties in the network in the presence of a dishonest majority. Our main technical contribution is an upper bound that matches a lower bound of Harnik, Ishai, and Kushilevitz (Crypto 2007), who studied the number of OT channels necessary and sufficient for MPC. In particular, we characterize which n-party OT graphs G allow t-secure computation of OT correlations between all pairs of parties, showing that this is possible if and only if the complement of G does not contain the complete bipartite graph $K_{n-t,n-t}$ as a subgraph.

The results presented in this chapter also appear in a joint work with Ranjit Kumaresan and Adam Sealfon [96].

4.1 Preliminaries

4.1.1 Graphs: Unsplittability

All graphs addressed in this chapter are undirected. We denote a graph as $G = (V, E)$ where V is a set of vertices and E is a set of edges. We denote an edge e as $e = \{v_1, v_2\}$, where $v_1, v_2 \in V$.

For $n \in \mathbb{N}$, let K_n denote the complete graph on n vertices. Let Λ_a^s denote the graph $G = (V, E)$ on $2a + s$ vertices with $V = V_A \dot{\bigcup} V_S \dot{\bigcup} V_B$, where $|V_A| = |V_B| = a$ and $|V_S| = s$, and

$$E = \{\{v_1, v_2\} : v_1 \notin V_A \vee v_2 \notin V_B\}$$

We will sometimes consider subgraphs of Λ_a^s which preserve labels of vertices. In this case we will always label the vertices so that vertex $A \in V_A$ and vertex $B \in V_B$.

For two graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ with the same vertex set V, we say that G_1 and G_2 are $(v_1, \ldots, v_\ell)$-*isomorphic*, denoted by $G_1 \simeq_{v_1,\ldots,v_\ell} G_2$, if the two graphs are isomorphic to one another while fixing the labelings of vertices $v_1, \ldots, v_\ell \in V$, that is, there exists an map σ such that $\sigma(v_i) = v_i$ for all $i \in [\ell]$.

Similarly, given graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with $V_1 \subseteq V_2$ and $v_1, \ldots, v_\ell \in V_1$, we say that G_1 is a $(v_1, \ldots, v_\ell)$-*subgraph* of G_2, denoted $G_1 \subseteq_{v_1,\ldots,v_\ell} G_2$, if G_1 is $(v_1, \ldots, v_\ell)$-isomorphic to some subgraph of G_2.

In particular, in the special case that graph $G = (V, E)$ contains vertices $A, B \in V$, we say that G is an (A, B)-*subgraph* of Λ_a^s (or that $G \subseteq_{A,B} \Lambda_a^s$) if there is an isomorphism σ between G and a subgraph of Λ_a^s such that A is mapped into set V_A and B is mapped into set V_B (that is, $\sigma(A) \in V_A$ and $\sigma(B) \in V_B$).

Call an n-vertex graph $G = (V, E)$ k-*unsplittable* for $k \leq n/2$ if any two disjoint sets of k vertices have some edge between them. That is, G is k-unsplittable if for all partitions of the vertices V into three disjoint sets V_1, V_2, V_3 of sizes $|V_1| = |V_2| = k$ and $|V_3| = n - 2k$, there exists some edge $(u, v) \in E$ with $u \in V_1, v \in V_2$. It is immediate from this definition that G is k-unsplittable if and only if $G \not\subseteq \Lambda_k^{n-2k}$.

Similarly, call G (k, A, B)-*unsplittable* for $k \leq n/2$ and $A, B \in V$ if any two disjoint

sets of k vertices containing A and B, respectively, have some edge between them. That is, G is (k, A, B)-unsplittable if for all partitions of the vertices of V into three disjoint sets V_1, V_2, V_3 of sizes $|V_1| = |V_2| = k$ and $|V_3| = n - 2k$ such that $A \in V_1$ and $B \in V_2$, there exists some edge $(u, v) \in E$ with $u \in V_1, v \in V_2$. From this definition we have immediately that G is (k, A, B)-unsplittable if and only if $G \not\subseteq_{A,B} \Lambda_k^{n-2k}$.

4.1.2 Adversaries

We consider security against *semi-honest adaptive t-threshold adversaries*, that is, adversaries that adaptively corrupt a set of at most t parties, where $0 \leq t < n$.[1] We assume the adversary to be *semi-honest* (i.e., *honest-but-curious*). That is, the corrupted parties follow the prescribed protocol, but the adversary may try to infer additional information about the inputs of the honest parties. As noted in [77], in the computational setting, using zero-knowledge proofs, it is possible to generically compile a protocol which is secure against semi-honest adversaries into another protocol which is secure against adaptive malicious adversaries [69].[2] This justifies our focus on the semi-honest setting here.

We will consider secure computation with perfect information-theoretic security and a non-adaptive adversary. By [30], in the semi-honest setting with information-theoretic security, any protocol which is non-adaptively secure is also adaptively secure. Consequently, satisfying this definition suffices to achieve adaptive security.

4.1.3 Oblivious Transfer

In this chapter OT refers to 1-out-of-2 oblivious transfer defined as follows.

Definition 10. *We define 1-out-of-2 oblivious transfer f_{OT} for a sender $A = P_1$ with inputs $x_0, x_1 \in \{0,1\}^m$, a receiver $B = P_2$ with input $b \in \{0,1\}$ and $n-2$ parties*

[1] Note that when $t = n$, there is nothing to prove.

[2] We note that in the computational setting, it is also possible to transform, in a *black-box* way, a protocol which is secure against semi-honest adversaries into another protocol which is secure against static malicious adversaries [76].

$P_3, \ldots, P_n$ *with input* $\perp$ *as*

$$f_{\mathrm{OT}}((x_0, x_1), b, \perp, \ldots, \perp) = (\perp, x_b, \perp, \ldots, \perp)$$

Note that while OT is typically defined as a 2-party functionality, the definition above adapts it our setting and formulates OT as an n-party functionality where only two parties supply non-$\perp$ inputs.

Definition 11. *Let G be a network consisting of n parties $A = P_1, B = P_2, P_3, \ldots, P_n$. Then a t-secure OT protocol $\Pi^{G,t}_{A \to B}$ is a protocol that t-securely computes the function f_{OT} on the inputs of the parties with A as the sender and B as the receiver.*

We note that OT is symmetric, in the following sense.

Lemma 15. *[123] If there exists a t-secure OT protocol $\Pi^{G,t}_{A \to B}$ for an n-party network G with n parties $A = P_1, B = P_2, P_3, \ldots, P_n$ with A as the sender and B as the receiver, then there exists a t-secure OT protocol $\widehat{\Pi}^{G,t}_{B \to A}$ for the same n parties with B as the sender and A as the receiver.*

We represent parties as nodes of a graph G where an edge $\{A, B\}$ indicates that parties A and B may run a 1-secure OT protocol with A as the sender and B as the receiver. By Lemma 15, the roles of the sender and receiver may be reversed, so it makes sense to define G as an undirected graph.

We note the following result regarding the completeness of OT for achieving arbitrary secure multiparty computation.

Lemma 16. *[89, 70, 86] Consider the complete network $G \simeq K_n$ on n vertices. Then, for any function $f : \mathcal{D}^n \to \mathcal{R}^n$, there exists a protocol Π which $(n-1)$-securely computes f, where party i receives the ith input $x_i \in \mathcal{D}$ and produces the ith output $(f(x))_i \in \mathcal{R}$.*

4.2 Our contributions

We introduce our main result:

Theorem (**Main Theorem for OT Infrastructure**). *Let $G = (V, E)$ be an OT graph on n parties $P_1, \ldots, P_n$, so that any pair of parties P_i, P_j which are connected by an edge may make an unbounded number of calls to an OT oracle. Let $\mathbb{A}$ be the class of semi-honest t-threshold adversaries which may adaptively corrupt at most t parties.*[3] *Then two parties A and B in $\{P_1, \ldots, P_n\}$ can information-theoretically emulate an OT oracle while being secure against all adversaries $\mathcal{A} \in \mathbb{A}$ if and only if*

1. *(honest majority) it holds that $t < n/2$; or*
2. *(trivial) A and B are connected by an edge in G; or*
3. *(partition) there exists no partition V_1, V_2, V_3 of G such that all of the following conditions are satisfied: (a) $|V_1| = |V_2| = n - t$ and $|V_3| = 2t - n$; (b) $A \in V_1$ and $B \in V_2$; and (c) for every $A' \in V_1$ and $B' \in V_2$ it holds that $(A', B') \notin E$.*

[3] Combining our work with results from [76, 69], we can also obtain computational security against malicious adversaries in both the nonadaptive and adaptive settings.

Organization: Roadmap to our main result

The remainder of this chapter is organized as follows:

- We give an overview of some of our techniques in Section 4.3, where we show solutions for the simple case when $n = 4$ and $t = 2$.
- Following this we briefly sketch the lower bound in Section 4.4 and describe the building blocks required for our upper bounds in Section 4.5.
- We prove our main result for the specific case of $t = n/2$ in Section 4.6.
- We then prove our main result for the specific case of $t = n - 2$ (Section 4.7).
- We then use each of the protocols in two different solutions to prove our main result the general case of $t \geq n/2$ in Section 4.8.
- Finally, in Section 4.9, we discuss minimal networks for $t \approx \frac{n}{2}$.

(a) G_{CK} (b) G_{HIK}

Figure 4-1: Known impossibility results. Securely computing f_{OT} between A' and B' is impossible for $t = 1$ in G_{CK} and is impossible for $t = 2$ in G_{HIK}.

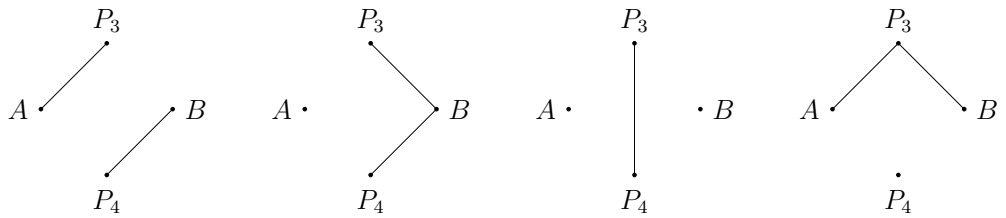
4.3 Constructions for the simple case of $n = 4$, $t = 2$

Let $G = (V, E)$ be an n-vertex graph representing a network with n parties, where an edge $\{P_i, P_j\} \in E$ indicates that parties P_i and P_j may run a 1-secure 2-party OT protocol with P_i as the sender and P_j as the receiver. Let $t < n$ be an upper bound on the number of corruptions made by the adversary. The central question considered in this work is the following. For which graphs G and which pairs of parties $A, B \in V$ does there exist a t-secure OT protocol with A as the sender and B as the receiver?

We begin by discussing some simple special cases of small networks. These will provide useful intuition for our main results. For $t < n/2$, it is possible to obtain a t-secure OT protocol for any n-vertex graph $G = (V, E)$ between any $A, B \in V$, since we can perform secure multiparty computation without any pre-existing OT channels if there is an honest majority [119]. It remains to consider the setting where $t \geq n/2$.

A few small cases have been resolved in prior work. For $n = 2$, $t = 1$, a 1-secure OT protocol (with perfect security) between the vertices of the two-vertex graph G does not exist unless the parties were already connected by an OT channel [38, 99]. This result is illustrated in Figure 4-1(a).

For $n = 3$, $t = 2$, it is known that we can obtain a 2-secure OT protocol between a pair of vertices A, B only if those vertices are already connected by an OT channel, even if there are OT channels from both A and B to the third vertex C as depicted in Figure 4-1(b). More generally, for any $n \geq 2$ and $t = n-1$, there exists a t-secure OT protocol with sender A and receiver B only if those vertices are already connected by an OT channel, even if all other $\binom{n}{2} - 1$ pairs of vertices are connected by OT channels [77]. This also resolves the question for $n = 4, t = 3$.



(a) G_1 (b) G_2 (c) G_3 (d) G_4

Figure 4-2: Cases for $n = 4$ parties with $t = 2$ corruptions.

The remainder of this section is devoted to exploring the setting $n = 4, t = 2$. This is the smallest case not resolved by prior techniques, and will illustrate many of the tools used in subsequent sections to obtain our general protocols. The key cases for $n = 4, t = 2$ are shown in Figure 4-2. As discussed below, these cases are sufficient to completely resolve the four-party setting.

4.3.1 Case 1 : Figure 4-2(a)

We first show that if $G \simeq_{A,B} G_1$ then there does not exist a 2-secure OT protocol for G with A as the sender and B as the receiver.[4] This is a consequence of the impossibility result of [38, 99]. An outline of the argument is as follows.

Consider components $\mathcal{C}_1 = \{A, P_3\}$ and $\mathcal{C}_2 = \{B, P_4\}$ of G, and let Π be a 2-secure protocol computing f_{OT} in G with A as the sender and B as the receiver. Then we can use Π to construct a 2-secure protocol Π' for the 2-party network in Figure 4-1(a) with A' as the sender and B' as the receiver. In protocol Π', party A' runs Π for both parties of component $\mathcal{C}_1$ of G, and B' runs Π for both parties of component $\mathcal{C}_2$. OT channel invocations can be handled locally, since all OT channels in G are between parties in the same component. Since protocol Π is 2-secure, in particular it is secure against corruptions of parties in $\mathcal{C}_1$ or the parties in $\mathcal{C}_2$. Consequently Π' is a 1-secure OT protocol for a network $G' \simeq_{A',B'} G_{\text{CK}}$ with A' as the sender and B' as the receiver. However, from [38, 99], we know that no such protocol exists with perfect security. Consequently there is no 2-secure protocol Π for a network $G \simeq_{A,B} G_1$.

[4] Recall that $H \simeq_{A,B} H'$ for two graphs H, H' if there exists an isomorphism between H and H' preserving the labels of vertices A and B.

Note that this impossibility holds not only for $G \simeq_{A,B} G_1$ but for any (A, B)-subgraph of G_1. In particular, any four-vertex graph $G = (V, E)$ with $|E| \leq 1$ cannot have a 2-secure protocol computing f_{OT} between vertices A and B except in the trivial case when there is already an edge $\{A, B\} \in E$. This technique of reducing to the known impossibility results of [38, 99, 77] to obtain lower bounds is described formally in Chapter 4.4.

4.3.2 Case 2 : Figure 4-2(b)

In this example we obtain a positive result, showing that there exists a 2-secure OT protocol with A as the sender and B as the receiver. Let the *degree* of party P denote the degree of the corresponding vertex in the OT network. Since B has degree 2 in G_2, we have that either B or at least one of its OT neighbors must be honest, and so one of the two OT channels must contain an honest party. This suggests the idea of using secret-sharing to ensure security against 2 corruptions.

Consider the following OT protocol where sender A has inputs $x_0, x_1 \in \{0,1\}^m$ and receiver B has input $b \in \{0,1\}$. A computes 2-out-of-2 shares (x_0^1, x_0^2) and (x_1^1, x_1^2) of its inputs x_0, x_1, respectively. A then sends shares x_0^1 and x_1^1 to party P_3 and x_0^2 and x_1^2 to party P_4. Parties P_3 and B invoke their secure OT channel with inputs (x_0^1, x_1^1) and b, and parties P_4 and B invoke their secure OT channel with inputs (x_0^2, x_1^2) and b respectively. B uses the obtained shares x_b^1, x_b^2 to reconstruct x_b.

We informally argue the 2-security of this protocol assuming that exactly one of A and B is corrupt.[5] Consider the case where A is corrupt and B is honest. The input of B is only used over secure OT channels, so by the 1-security of the OT channels with P_3 and P_4, the corrupt parties can learn nothing about B's input bit b. Now consider the case where B is corrupt and A is honest. Either P_3 or P_4 must be honest. If P_3 is honest then the security of OT channel $\{P_3, B\}$ implies that B learns nothing about share x^1_{1-b}, so the security of the secret sharing scheme implies that the corrupt parties do not use x_{1-b}. By symmetry, the same argument applies if P_4 is honest. This completes the argument.

Note that by Lemma 15, we can also obtain a 2-secure OT protocol from A to B whenever A has degree 2 in OT network. Furthermore, we can extend this idea to construct a t-secure OT protocol whenever either the sender or the receiver has degree at least t. We call this protocol the t-claw protocol and describe it in detail in Chapter 4.5.1.

4.3.3 Case 3 : Figure 4-2(c)

Somewhat surprisingly, we can also show a positive result for graphs $G \simeq_{A,B} G_3$ even though the OT network has no edges involving either the sender A or the receiver B. The protocol is as follows. Since parties P_3 and P_4 have an OT channel between them, by Lemma 16, they can perform 1-secure MPC between them. P_3 and P_4 use an MPC to compute 2-out-of-2 shares of OT correlations with uniformly random inputs and send corresponding shares to A and B who can then reconstruct the correlations. More concretely, the MPC protocol computes 2-out-of-2 shares (r^1_0, r^2_0), (r^1_1, r^2_1) of two randomly sampled m-bit strings r_0, r_1, 2-out-of-2 shares (c^1, c^2) of a random bit $c \in \{0, 1\}$, and independent 2-out-of-2 shares (s^1, s^2) of the string r_c. Party P_3 receives the first share of each secret, and party P_4 receives the second share. Party P_3 then

[5] An additional step is needed to address the case in which A and B are both honest. Then P_3 and P_4 can both be corrupt and learn x_0 and x_1, the inputs of A. This can be handled with the technique of OT correction, using a one-time pad and the secure point-to-point channel between A and B. Equivalently, we could run the protocol on random inputs, and then use method of [9] to obtain 1-out-of-2 OT from random OT. If A and B are both corrupt then there is nothing to prove.

sends shares r_0^1, r_1^1 to A and s^1, c^1 to B, while P_4 sends shares r_0^2, r_1^2 to A and s^2, c^2 to B. A can then reconstruct r_0 and r_1, and B can reconstruct c and r_c. Parties A and B have now established a random OT correlation, which they can use to perform OT with their original inputs using OT correction [9].[6]

We now informally argue the 2-security of this protocol. If A and B are both honest, then the corrupt parties receive no information about their inputs, while if A and B are both corrupt then there is nothing to prove. Consequently we can assume that exactly one of A and B is corrupt and that either P_3 or P_4 is honest. If A is corrupt and P_3 or P_4 is honest, then the adversary learns nothing about c and r_c, since it only sees one of the two shares of each. The OT correction phase uses these strings as one-time pads for inputs which are unknown to the adversary, and consequently are information-theoretically hidden from the adversary. Consequently A learns nothing about B. The case where B is corrupt and P_3 or P_4 is honest follows from the same argument.

This construction can be extended to obtain a t-secure OT protocol whenever the OT graph contains a t-clique consisting of t parties which are not the OT sender or receiver. We call this protocol the t-clique protocol and describe it in detail in Chapter 4.5.2.

4.3.4 Case 4 : Figure 4-2(d)

We also obtain a positive result for graphs $G \simeq_{A,B} G_4$. We introduce here a technique we call cascading. The idea is as follows. Using the protocol described in Chapter 4.3.2 for network G_2 of Figure 4-2(b), we have 2-secure OT protocol with P_3 as the sender and P_4 as the receiver. This effectively gives us an OT channel between P_3 and P_4. Applying the protocol from Chapter 4.3.3 on the augmented network, we now have a 2-secure OT protocol with A as the sender and B as the receiver. We describe this pictorially in Figure 4-3.

The 2-security of the protocol follows from the 2-security of the underlying pro-

[6]This OT correction step can be performed as follows. Party B sends $b' = b \oplus c$ to A. A responds with $y_0 = x_0 \oplus r_{b'}$ and $y_1 = x_1 \oplus r_{1-b'}$. Finally, B computes $y_b \oplus r_c = x_b$.

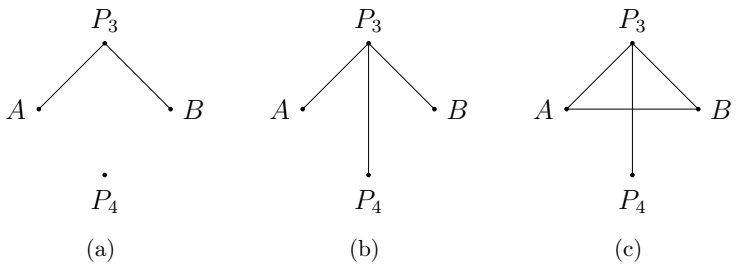


Figure 4-3: The cascading protocol for Case 4 : Figure 4-2(d); (a) → (b) → (c)

tocols of Sections 4.3.2 and 4.3.3. The technique of cascading for combining t-secure protocols is described in detail in Chapter 4.5.3.

4.3.5 Cases 1–4 are exhaustive

Note that a t-secure OT protocol with sender A and receiver B in an OT network G trivially yields a t-secure protocol for any network G' such that $G \subseteq_{A,B} G'$. From cases 1 and 4, we can securely compute f_{OT} in a network G containing at most a single edge if and only if the edge is $\{A, B\}$ or $\{P_3, P_4\}$. From cases 1, 2, and 3, we can compute f_{OT} in a network G containing two or more edges including neither of $\{A, B\}$ or $\{P_3, P_4\}$ if and only if there is some vertex with degree at least 2 in the OT graph. This completes the characterization of 4-party networks with 2 corruptions.

4.4 Reformulating the Lower Bound of [77]

We now describe a family of impossibility results by a generic reduction to the impossibility result in [77], which we restate in our language below.

Lemma 17. *[77] Consider any three party network G with $G \simeq_{A',B'} G_{\text{HIK}}$, the graph in Figure 4-1(b). Then any 2-secure OT protocol with A' as the sender and B' as the receiver can be used (as a black box) to obtain a 1-secure OT protocol for a network G' with $G' \simeq_{A',B'} G_{\text{Kus}}$, the graph in Figure 4-1(a), with A' as the sender and B' as the receiver.*

The theorem below describes a family of impossibility results over a corresponding family of networks. We note that this result was observed in [77]; we restate it our language and provide a formal proof.

Theorem 8. *Let $n \geq 2$ and $n/2 \leq t < n$, and let G be an n party network such that $G \subseteq \Lambda_{n-t}^{2t-n}$, with $P_1 \in V_A$ and $P_2 \in V_B$. Any t-secure OT protocol for G with P_1 as the sender and P_2 as the receiver can be used (as a black box) to obtain a 1-secure OT protocol for a network G' with $G' \simeq_{A,B} G_{\text{Kus}}$ with A' as the sender and B' as the receiver.*

Proof. Let G be an n party network with $G = (V, E)$ such that $G \subseteq \Lambda_{n-t}^{2t-n}$. Then, we may write $V_{\mathsf{U}} = V_A \dot{\bigcup} V_S \dot{\bigcup} V_B$, where $|V_A| = |V_B| = n - t$ and $|V_S| = 2t - n$, with $P_1 \in V_A$ and $P_2 \in V_B$ and $E(V_A, V_B) = \emptyset$, where $E(V_A, V_B)$ represents the set of edges with one endpoint in V_A and the other in V_B.

Let Π be a t-secure OT protocol for G with P_1 as the sender and P_2 as the receiver. If $t > n/2$, then we can use Π to construct a 2-secure OT protocol Π' for any three party network G' with $G' \simeq_{A',B'} G_{\text{HIK}}$ with A' as the sender and B' as the receiver below. Combining this with Lemma 17, the conclusion follows. We describe the construction of Π' below. If $t = n/2$, then we can use Π to construct a 1-secure OT protocol Π'' for any two party network G'' with $G'' \simeq_{A'',B''} G_{\text{CK}}$ with A'' as the sender and B'' as the receiver. The construction of Π'' is exactly the same as that of Π' and hence we omit its description here.

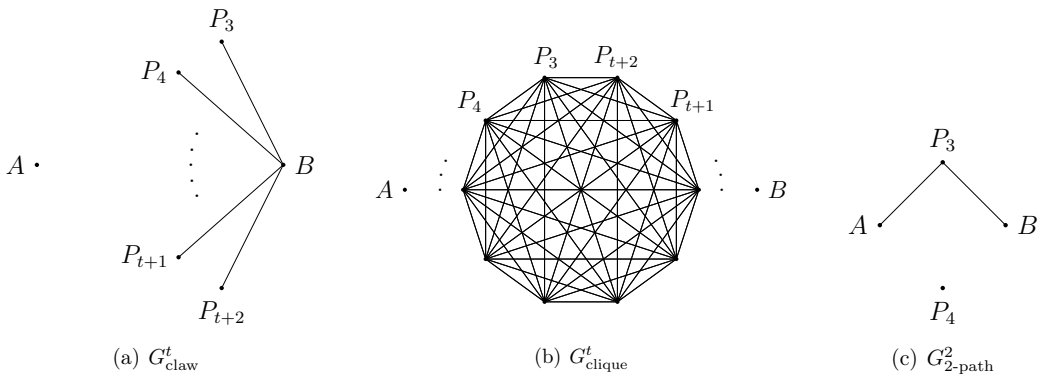
In protocol Π', party A' simulates the parties of component V_A, party C' simulates the parties of component V_S, and party B' simulates the parties of component V_B. Executions of 1-secure OT protocols between parties of the same component are be handled locally and executions of 1-secure OT protocols between parties in different components is handled as follows:

- If the parties are in components V_A and V_S, then executions of 1-secure OT protocols between the parties are carried out using the OT edge $\{A', C'\}$ in the network G'.

- If the parties are in components V_B and V_S, then executions of 1-secure OT protocols between the parties are carried out using the OT edge $\{B', C'\}$ in the network G'.

Since $G \subseteq \Lambda_{n-t}^{2t-n}$, there are no executions of 1-secure OT protocols between parties in components V_A and V_B in the protocol Π.

Correctness of Π' is obvious. We now prove 2-security of Π'. Intuitively, since Π is t-secure, in particular, it is secure against corruptions of parties $V_A \cup V_S$ or the parties $V_B \cup V_S$. Consequently protocol Π' is secure against corruptions $\{A', C'\}$ or $\{B', C'\}$ and hence Π' is a 2-secure protocol. $\square$



(a) G^t_{claw} (b) G^t_{clique} (c) $G^2_{\text{2-path}}$

Figure 4-4: Building block networks – (a) t-claw; (b) t-clique; (c) 2-path

4.5 Building Blocks

In this chapter, we describe a few key protocols and techniques which we use in the subsequent sections to prove our main theorem.

4.5.1 The t-claw Protocol

The first protocol we describe is the t-claw protocol, where the graph G describing the network is such that $G \simeq_{A,B} G^t_{\text{claw}}$. The protocol is described in Protocol 1. The protocol is a straightforward generalization of the one described in Chapter 4.3.2. The idea is for A to compute t-out-of-t shares of its inputs and distribute them among the t parties connected to B. These t parties then perform OT with B so that B receives the shares to reconstruct his output.

Lemma 18. *Protocol 1 is an efficient t-secure OT protocol for a network $G \simeq_{A,B} G^t_{\text{claw}}$ with A as the sender and B as the receiver.*

Proof Intuition. The t-security of the protocol can be seen as follows. Steps 1, 2 and 7 perform OT correction, that is, they perform a random OT to 1-out-of-2 OT transformation. This transformation protects against the case that the parties $P_3, \ldots, P_{t+2}$ (that is, all but A and B) are corrupt. Suppose A were corrupt and B were honest. Clearly, A colluding with any of the parties $P_3, \ldots, P_{t+2}$ provides A with no additional information since all they possess are shares sent by A. Next, if A

Protocol 1: t-claw Protocol

Preliminaries: Let $A, B, P_3, \ldots, P_{t+2}$ be the $t+2$ parties in a network $G \simeq_{A,B} G^t_{\text{claw}}$. A has inputs $x_0, x_1 \in \{0,1\}^m$ and B has input $b \in \{0,1\}$.

Protocol:

1. B chooses a random bit $c \in \{0,1\}$ and sends $b' = b \oplus c$ to A.
2. A chooses two random one-time pads $r_0, r_1 \in \{0,1\}^m$ and sends $y_0 = x_0 \oplus r_{b'}$ and $y_1 = x_1 \oplus r_{1-b'}$ to B.
3. A then computes t-out-of-t shares $(r_0^1, \ldots, r_0^t)$ and $(r_1^1, \ldots, r_1^t)$ of r_0, r_1, respectively.
4. For each $i \geq 3$, A sends shares r_0^i and r_1^i to party P_i.
5. For each $i \geq 3$, parties P_i and B execute the OT protocol $\Pi^{G,1}_{P_i \to B}$ with inputs (r_0^i, r_1^i) and c respectively.
6. B uses the obtained shares $r_c^1, \ldots, r_c^t$ to reconstruct r_c.
7. B finally computes $y_b \oplus r_c = x_b$.

were honest and B corrupt, at least one of the parties $P_3, \ldots, P_{t+2}$ must be honest. B has no information about those shares and hence does not learn anything. Finally, if both A and B were corrupt, there is nothing to prove.

Proof. Let $\mathcal{A}$ be a t-threshold adversary which corrupts parties T, $|T| \leq t$. We will construct a simulator $\mathcal{S}$ which plays the role of the uncorrupted parties. If $\{A, B\} \subset T$ then the uncorrupted parties receive no input, so $\mathcal{S}$ can perfectly simulate the uncorrupted parties. If $\{A, B\} \cap T = \emptyset$ then $\mathcal{S}$ chooses arbitrary inputs x_0, x_1, b and runs the protocol, invoking the OT simulator for each OT invocation with an uncorrupted party in step 5. Since corrupted parties only learn secret shares of independently random values, the view of the adversary is independent of the choice of x_0, x_1, b and is identical to the real world.

Otherwise, we have that the corrupted parties T include exactly one of A, B. If $A \in T$ but $B \notin T$, then $\mathcal{S}$ chooses arbitrary input b and runs the protocol, invoking the OT simulator for each OT invocation with an uncorrupted party in step 5. Since the OT simulator does not reveal the input c, and since the adversary only learns the direct sum of b with the random bit c, the view of the adversary is identical regardless of the value of b and in particular is identical to the real world.

Finally we have the case $B \in T, A \notin T$. Here the simulator $\mathcal{S}$ is given the output

value x_b. $\mathcal{S}$ runs the protocol with (x_b, x_b) as the input to A, again invoking the OT simulator for each OT invocation with an uncorrupted party in step 5. Since $|T| \leq t$ and $B \in T$, at most $t-1$ of the t parties $P_3, \ldots, P_{t+2}$ are corrupted. Consequently the adversary observes at most $t-1$ shares of the random one-time pads r_0, r_1, so by the security of t-out-of-t secret sharing, conditioned on the remaining shares being hidden, the distribution of the observed shares is independent of r_0, r_1. The adversary learns the shares of r_c, but by the security of the OT channels, the view of the adversary in step 3 and onward is independent of the remaining shares of r_{1-c} and consequently is independent of the choice of r_{1-c}. Consequently the view of the adversary in step 3 and onward is independent of r_{1-c}. In step 2, the adversary sees $y_b = x_b \oplus r_c$ and $y_{1-b} = x_{1-b} \oplus r_{1-c}$, so by the security of the one-time pad, the view of the adversary is independent of x_{1-b}. Consequently the overall view of the adversary is identical in the real and ideal worlds. $\square$

Protocol 2: t-clique Protocol

Preliminaries: Let $A, B, P_3, \ldots, P_{t+2}$ be the $t+2$ parties in a network $G \simeq_{A,B} G^t_{\text{clique}}$. A has inputs $x_0, x_1 \in \{0,1\}^m$ and B has input $b \in \{0,1\}$.

Protocol:

1. Parties $P_3, \ldots, P_{t+2}$ use their pairwise OT channels to run t-secure MPC for the function f using the protocol from Lemma 16 for the function f described ahead. The function f is to securely compute t-out-of-t shares $(r_0^1, \ldots, r_0^t)$, $(r_1^1, \ldots, r_1^t)$ of two randomly sampled one-time pad keys r_0, r_1, $(c^1, \ldots, c^t)$ of a random bit $c \in \{0,1\}$, and independent shares $(s^1, \ldots, s^t)$ of key r_c, so that party $i+2$ receives only shares r_0^i, r_1^i, s^i, c^i for each i.
2. Each party P_{i+2} for $i \geq 1$ sends shares r_0^i, r_1^i to A and s^i, c^i to B.
3. A uses shares $(r_0^1, \ldots, r_0^t)$ and $(r_1^1, \ldots, r_1^t)$ to reconstruct r_0 and r_1.
4. B uses shares $(c^1, \ldots, c^t)$ and $(s^1, \ldots, s^t)$ to reconstruct c and r_c and sends $b' = b \oplus c$ to A.
5. A computes $y_0 = x_0 \oplus r_{b'}$ and $y_1 = x_1 \oplus r_{1-b'}$ and sends both to B.
6. B computes $y_b \oplus r_c = x_b$.

4.5.2 The t-clique Protocol

The next protocol we describe is the t-clique protocol, where the graph G describing the network is such that $G \simeq_{A,B} G^t_{\text{clique}}$. The protocol is described in Protocol 2. The protocol is a straightforward generalization of the one described in Chapter 4.3.3. The idea is for the parties $P_3, \ldots, P_{t+2}$ to compute t-out-of-t shares of OT correlations and send them to A and B respectively. This is done via multiparty computation since the parties have a complete network of OT channels (Lemma 16). A and B then perform OT correction using their secure channel.

Lemma 19. *Protocol 2 is an efficient t-secure OT protocol for a network $G \simeq_{A,B} G^t_{\text{clique}}$ with A as the sender and B as the receiver.*

Proof Intuition. The t-security of the protocol can be seen as follows. Steps 4, 5 and 6 perform OT correction, that is, they perform a random OT to 1-out-of-2 OT transformation. This transformation protects against the case that the parties $P_3, \ldots, P_{t+2}$ (that is, all but A and B) are corrupt. If one of A and B were corrupt,

there exists at least one honest party among the parties $P_3, \ldots, P_{t+2}$. Hence, even by colluding, A or B would have no information about those shares and would not learn anything. Finally, if both A and B were corrupt, there is nothing to prove.

Proof. Let $\mathcal{A}$ be a t-threshold adversary which corrupts parties T, $|T| \leq t$. We will construct a simulator $\mathcal{S}$ which plays the role of the uncorrupted parties. As above, if $\{A, B\} \subset T$ then the uncorrupted parties receive no input, so $\mathcal{S}$ can perfectly simulate the uncorrupted parties. If $\{A, B\} \cap T = \emptyset$ then $\mathcal{S}$ chooses arbitrary inputs x_0, x_1, b and runs the protocol. Since the only steps which depend at all on the inputs are on point-to-point channels between A and B, the view of the adversary in the real and ideal worlds is identical.

Otherwise, we have that the corrupted parties T include exactly one of A, B. If $A \in T$ but $B \notin T$, then $\mathcal{S}$ chooses arbitrary input b and runs the protocol, invoking the MPC simulator for the protocol in step 1 (the existence of this simulator follows from Lemma 16). Since at least one of the parties $P_3, \ldots, P_{t+2}$ is uncorrupted, the security of the MPC protocol implies that the view of the adversary is independent of the uncorrupted parties' shares s^i and c^i, and so by the security of the secret sharing scheme is independent of the value of the bit c. The only message received by the adversary which depends on b is the bit $b' = b \oplus c$, so it follows that the view of the adversary is independent of the bit b and therefore is identical in the real and ideal worlds.

Finally we have the case $B \in T, A \notin T$. Here the simulator $\mathcal{S}$ is given the output value x_b. $\mathcal{S}$ runs the protocol with (x_b, x_b) as the input to A, again invoking the MPC simulator for the protocol in step1. Since at least one of the parties $P_3, \ldots, P_{t+2}$ is uncorrupted, the security of the MPC protocol implies that the view of the adversary is independent of the uncorrupted parties' shares r_0^i and r_1^i, so by the security of t-out-of-t secret sharing, conditioned on the remaining shares being hidden, the distribution of the observed shares is independent of r_0, r_1. The adversary learns the shares of r_c, but by the security of the OT channels, the view of the adversary through step 4 is independent of the remaining shares of r_{1-c} and consequently is independent of the choice of r_{1-c}. In step 5, the adversary sees $y_b = x_b \oplus r_c$ and $y_{1-b} = x_{1-b} \oplus r_{1-c}$,

so by the security of the one-time pad, the view of the adversary is independent of x_{1-b}. Consequently the overall view of the adversary is identical in the real and ideal worlds. □

4.5.3 Cascading

The following building block is a generalization of the technique described in Chapter 4.3.4. The technique describes a general method of combining protocols iteratively. In our context, this can be thought of a tool for transforming a network described by a graph G to one described by a graph G', where $G \subseteq_V G'$ and G and G' are both graphs on the same vertex set V. In other words, it describes protocols as adding new edges indicating the establishment of OT correlations between new pairs of parties in the network. With this abstraction, it is easy to view the technique of cascading as one which combines protocols iteratively to transform the underlying network by adding new edges. This is described formally below.

Definition 12. *Let $G = (V, E)$ and $G' = (V, E')$ be two graphs on the same set of vertices, V, with $G \subseteq_V G'$. We say that a protocol Π t-*transforms *a network G into the network G' if for each $\{P_i, P_j\} \in E' \setminus E$, Π is a t-secure OT protocol for a network G with P_i as the sender and P_j as the receiver.*[7]

Lemma 20. *If Π_1 is a protocol that runs in time T_1 and t-transforms network G_1 into G_2, and Π_2 is a protocol that runs in time T_2 and t-transforms network G_2 into G_3, then there exists a protocol Π that runs in time T_1T_2 and t-transforms G_1 into G_3.*

Proof. The protocol Π simply runs Π_2, running protocol Π_1 to obtain correlations whenever Π_2 invokes OT on an edge of $G_2 \setminus G_1$. Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be the simulators associated with Π_1 and Π_2 respectively. The simulator for Π simply runs $\mathcal{S}_2$, invoking $\mathcal{S}_1$ for OT calls made on edges in $G_2 \setminus G_1$. □ □

[7] Note that a single protocol Π may set up independent OT correlations for several pairs of parties $\{P_i, P_j\} \in E' \setminus E$.

Using OT extension [10, 84], we can also obtain a computationally secure version of cascading with improved efficiency.

Lemma 21. *Let λ be a computational security parameter. Assuming the existence of one-way functions or correlation-robust hash functions, if Π_1 is a protocol that runs in time T_1 and t-transforms network G_1 into G_2, and Π_2 is a protocol that runs in time T_2 and t-transforms network G_2 into G_3, then there exists a computationally secure protocol Π that runs in time $\lambda \cdot T_1 + T_2 \cdot \text{poly}(\lambda)$ and t-transforms G_1 into G_3.*

Proof. First, run protocol Π_1 λ times on random inputs to obtain λ independent OT correlations for each edge of $G_2 \setminus G_1$. Then run Protocol Π_2, using OT extension [10, 84] to obtain OT correlations for OT calls made on edges in $G_2 \setminus G_1$. □

4.5.4 The 2-path graph

The protocol described here is a commonly used subroutine in several of the protocols which follow. It is a particular combination of the tools encountered in Chapters 4.5.1, 4.5.2 and 4.5.3). The subroutine, which we call 2-path, is the same as the one described in Chapter 4.3.4. It is used to obtain OT correlations between parties who have a common neighbor in a four-party network with at most two corruptions (see Figure 4-4(c)).

Lemma 22. *Protocol 3 is an efficient 2-secure OT protocol for a network $G \simeq_{A,B} G^2_{\text{2-path}}$ with A as the sender and B as the receiver.*

Proof. This follows immediately from Lemma 20 and the 2-security of Protocols 1 and 2 for $t = 2$ (Lemmata 18 and 19). □

Protocol 3: 2-path

Preliminaries: Let A, B, C, D be the parties, and let there exist OT channels (A, C) and (B, C). A has input (x_0, x_1), and B has input $b \in \{0, 1\}$.

Protocol:

1. Invoke Protocol 1 (2-claw) on parties (D, C, A, B) to obtain OT correlations on edge (D, C).
2. By Lemma 20, we have an OT channel between D and C.
3. Invoke Protocol 2 (2-clique) on parties (A, B, C, D).

4.5.5 Combiners [109, 78]

The notion of OT combiners is one which aims at combining several candidate protocols for establishing OT correlations between two parties with the property that a majority of them remain secure in the presence of any adversary $\mathcal{A}$ from a class of adversaries $\mathbb{A}$ into a single protocol which remains secure in the presence of any adversary from the same class $\mathbb{A}$. The following lemma is due to [109, 78], relying on prior work by [79, 124] based on a construction by [46].

Lemma 23. *[109, 78] Let $\mathbb{A}$ be an adversary class. Suppose there exist m protocols $\Pi_1, \ldots, \Pi_m$ for $f_{OT}(A, B, P_1, \ldots, P_n)$ such that for any adversary $\mathcal{A} \in \mathbb{A}$ a majority of the protocols are secure. Then, there exists a protocol $\Pi^*(\Pi_1, \ldots, \Pi_m)$ for $f_{OT}(A, B, P_1, \ldots, P_n)$ which is secure against all adversaries $\mathcal{A} \in \mathbb{A}$. Moreover, if each protocol Π_i is efficient and perfectly secure, then so is Π^*.*

4.6 The case $t = n/2$

We now consider the specific case of $t = n/2$, that is, at most half the parties are corrupt. We note that this is the smallest value of t for which the question is non-trivial. From the lower bounds proven in Theorem 8, we already have that for all n-party networks G containing A and B such that $G \subseteq_{A,B} \Lambda^0_{n/2}$, there exists no $n/2$-secure OT protocol with A as the sender and B as the receiver. Quite surprisingly, we show ahead, in Theorem 9, that all other networks $G \not\subseteq_{A,B} \Lambda^0_{n/2}$ admit an $n/2$-secure OT protocol with A as the sender and B as the receiver. In fact, we show an explicit $n/2$-secure OT protocol with A as the sender and B as the receiver whenever the network G is $(n/2, A, B)$-unsplittable, that is, when $G \not\subseteq_{A,B} \Lambda^0_{n/2}$.

Theorem 9. *Consider an n-party network G, which contains A and B as two of the parties. Protocol 5 is an $n/2$-secure OT protocol with A as the sender and B as the receiver if and only if G is $(n/2, A, B)$-unsplittable.*

Protocol 4: Completing Triangles

Preliminaries: Let $A, B, C, P_4, \ldots, P_n$ be the n parties, and let there exist OT channels (A, C) and (B, C). A has input (x_0, x_1), and B has input $b \in \{0, 1\}$.

Protocol:

1. Run a combined protocol $\Pi^*(\Pi_4, \ldots, \Pi_n)$ on the $n-3$ protocols $\Pi_4, \ldots, \Pi_n$, where
   - For each $i \geq 4$, Π_i denotes an invocation of Protocol 3 (2-path) with the four parties A, B, C, P_i with A as the sender and B as the receiver.

The protocol we describe proceeds in two stages. In the first stage, the protocol transforms every connected component of the network into a clique. This transformation is very specific to the case of $t = n/2$, in particular, that each connected component can actually function as a clique is true only in this case. This transformation is carried out by means of repeatedly calling a protocol we call "Completing Triangles" which obtains OT correlations between parties who have a common neighbour. We have already seen a special case of this protocol in the analysis for $n = 4$ and $t = 2$ in Chapter 4.5.4. In fact, the protocol which achieves the same goal in the case of general $n \geq 4$ and $t = n/2$ uses the building block Protocol 3 along with the building block of OT combiners described in Chap 4.5.5. Protocol 4 achieves this.

Lemma 24. *Let G be an n-vertex OT network with edges $\{A, C\}$ and $\{B, C\}$. Protocol 4 is an $n/2$-secure OT protocol for the network G with A as the sender and B as the receiver.*

Proof. We consider cases depending on the number of corrupted parties in the set $T = \{A, B, C\}$. If T contains at most one corrupted party, then each tuple (A, B, C, P_i) for $i \geq 4$ contains at most 2 corrupted parties, so each protocol Π_i in step 1 is secure. If T contains two corrupted parties, then there are at most $t - 2 = (n-4)/2$ corrupted parties among $P_4, \ldots, P_n$, so a majority of these parties are honest. Consequently a majority of the protocols Π_i which are combined in step 1 are secure. Thus, in either case, by Lemma 23 the protocol is secure. Finally, if all three parties of T are corrupted, then all uncorrupted parties receive no input, so the simulator $\mathcal{S}$ can

perfectly simulate the uncorrupted parties by running the honest protocol. Therefore Protocol 4 is $n/2$-secure. □

Proof Intuition (Theorem 9): It is easy to see that by executing Protocol 4 repeatedly, one can obtain OT correlations between any pair of parties in the same connected component. In other words, for $t = n/2$, we can assume that we are given a network which consists of disjoint cliques (Lemma 20). This is done in step 1 of Protocol 5. Hence, if A and B were in the same connected component in G, this process would end up with correlations between A and B and we can terminate the protocol.

Assume this is not the case. A natural next step to try is to run the clique protocol described in Chapter 4.5.2 with each of the cliques and parties A and B with the intent of setting up OT correlations between A and B. The troubling aspect, however, is that we are unable to fix the parameter t in any of the cliques. Indeed, in many of these invocations, the number of corrupted parties may be more than what Protocol 2 can handle in order to guarantee security. However, for an invocation to be secure, we only require that the clique contains at least one honest party. This is because we can assume without loss of generality that at least one of A or B is honest since otherwise we have nothing to prove. But now, this gives us that a majority of the cliques actually contain at least one honest party. Hence, if we invoke Protocol 2 for each of the parties on their respective cliques, a majority of them would be secure and now we can combine these candidate invocations to obtain a secure protocol following Lemma 23. This is performed in step 5 of the Protocol 5. Finally, we note that steps 3, 4 and 6 perform OT correction, that is, they perform a random OT to 1-out-of-2 OT transformation. This describes $n/2$-security of Protocol 5.

Proof of Theorem 9. The "only if" part of theorem has been proved by virtue of the lower bound proven in Theorem 8 with $t = n/2$. We now prove the "if" part. We note that in the case where A and B are in the same connected component in the network G, by the $n/2$-security of Protocol 4 and Lemma 20, we note that Protocol 5 is an $n/2$-secure OT protocol with A as the sender and B as the receiver.

Protocol 5: $n/2$ corruptions

Preliminaries: Let $P_1 = A, P_2 = B, P_3, \ldots, P_n$ be the n parties in a network $G = (V, E)$. A has input (x_0, x_1), and B has input $b \in \{0, 1\}$.

Protocol:

1. While there exist parties $P_i, P_j, P_k \in V$ such that $\{P_i, P_j\} \in E$, $\{P_j, P_k\} \in E$, but $\{P_i, P_k\} \notin E$:
   (a) Let S be the set of triples of distinct vertices $(X, Y, Z) \in V^3$ which satisfy the conditions $\{X, Y\} \in E, \{Y, Z\} \in E$, and $\{X, Z\} \notin E$.
   (b) For each triple $(X, Y, Z) \in S$, invoke Protocol 4 with independent random inputs $(r_0^{i,k}, r_1^{i,k})$ and $b^{i,k}$, to obtain OT correlations along edge $\{X, Z\}$.
   (c) Invoking cascading (Lemma 20), we can add $\{X, Z\}$ to the edge set E for all triples $(X, Y, Z) \in S$.

   The OT network G now consists of disjoint cliques $\mathcal{C}_1, \ldots, \mathcal{C}_\ell$.
2. If A and B are in the same clique, then halt.
3. B samples a random bit c and sends $b' = b \oplus c$ to A.
4. A chooses random one-time pads r_0, r_1 and sends $y_0 = x_0 \oplus r_{b'}$ and $y_1 = x_1 \oplus r_{1-b'}$ to B.
5. Let $\mathcal{C}_1$ be the clique containing A and $\mathcal{C}_2$ be the clique containing B. For each party P_i, $i \geq 3$, let $\mathcal{C}(i)$ denote the clique containing party i, and let $P_{j_1}, \ldots, P_{j_{|\mathcal{C}(i)|}}$ denote the parties in clique $\mathcal{C}(i)$. Run a combined protocol $\Pi^*(\Pi_1, \ldots, \Pi_n)$ on the n protocols $\Pi_1, \ldots, \Pi_n$, where
   - For each $i \in [n]$, Π_i denotes an invocation of Protocol 2 on the $|\mathcal{C}(i)| + 2$ parties $A, B, P_{j_1}, \ldots, P_{j_{|\mathcal{C}(i)|}}$ with inputs (r_0, r_1) and c.[a]
6. Finally, B computes $x_b = y_b \oplus r_c$.

[a] In the case $\mathcal{C}(i) = \mathcal{C}_1$, A is both the OT sender and a member of the clique. A similar condition holds for B in the case $\mathcal{C}(i) = \mathcal{C}_2$.

We now proceed to the case where A and B are not in the same connected component in G. We must show that the protocol is secure against t-threshold adversaries as long as the vertices cannot be partitioned into two sets V_A, V_B each of size $t = n/2$ with $A \in V_A, B \in V_B$ such that there are no edges between V_A and V_B. Let $\mathcal{A}$ be a t-threshold adversary which corrupts parties T, $|T| \leq t$. We will construct a simulator $\mathcal{S}$ which plays the role of the uncorrupted parties.

If $\{A, B\} \subset T$ then the uncorrupted parties receive no input, so the simulator can perfectly simulate the uncorrupted parties. If $\{A, B\} \cap T = \emptyset$ then $\mathcal{S}$ chooses arbitrary inputs x_0, x_1, b and runs the protocol. Since the only steps which depend on the input at all are on point-to-point channels between A and B, the view of the

adversary in the real and ideal worlds is identical.

Otherwise, we have that the corrupted parties T include exactly one of A, B. If $A \in T$ but $B \notin T$, then $\mathcal{S}$ chooses an arbitrary bit b and runs the protocol, invoking the OT simulator for each invocation of Protocol 4. It follows that as long as the combined protocol Π^* in step 5 is secure against $\mathcal{A}$, Protocol 5 is secure against $\mathcal{A}$. It remains to show that a majority of the n protocols $\Pi_1, \ldots, \Pi_n$ are secure against $\mathcal{A}$. Since party B is honest, by Lemma 19, protocol Π_i is secure against $\mathcal{A}$ as long as at least one of the parties in clique $\mathcal{C}(i)$ is honest. In particular, if party P_i is honest then protocol Π_i is secure against $\mathcal{A}$. At most t of the parties $P_1, \ldots, P_n$ are corrupt, so the only protocols which may be insecure against $\mathcal{A}$ are the t protocols Π_i corresponding to the corrupted parties P_i. Assume that all all t of these protocols are insecure against $\mathcal{A}$. We then have the corrupted parties lie in completely corrupted cliques who sizes sum up to $n/2$. This then gives a set $V_A = T$ of $n/2$ parties containing A but not B such that there are no edges from V_A to the remaining vertices $V_B = \overline{T}$. However, we know that G possesses no such partition. Hence, at most $t - 1 < n/2$ of the n protocols are insecure against $\mathcal{A}$ and hence by Lemma 23, the combined protocol Π^* in step 5 is secure and hence Protocol 5 is secure against $\mathcal{A}$.

The remaining case that $B \in T$ but $A \notin T$ is similar. Here, the simulator $\mathcal{S}$ is given the output value x_b. $\mathcal{S}$ runs the protocol with (x_b, x_b) as the input to A, again invoking the OT simulator for each invocation of Protocol 4. As above, as long as the combined protocol Π^* in step 5 is secure against $\mathcal{A}$, Protocol 5 is secure against $\mathcal{A}$. By the same argument, the only protocols Π_i which may be insecure against $\mathcal{A}$ are the t protocols corresponding to the corrupted parties P_i. If all t of these protocols are insecure against $\mathcal{A}$, as above, we have a set $V_A = \overline{T}$ of $n/2$ parties containing A but not B such that there are no edges from V_A to the remaining vertices $V_B = T$. However, we know that G possesses no such paritition. Hence, at most $t - 1 < n/2$ of the n protocols are insecure against $\mathcal{A}$ and hence by Lemma 23, the combined protocol Π^* in step 5 is secure and hence Protocol 5 is secure against $\mathcal{A}$. $\square$

We analyze the efficiency of the protocol in Theorem 10 below. The protocol as stated runs in quasi-polynomial time. We can also obtain a computationally secure protocol

which runs in polynomial time.

Theorem 10. *Protocol 5 runs in quasi-polynomial time. Assuming one-way functions, we can obtain a computationally secure protocol which runs in polynomial time using computationally secure cascading (Lemma 21).*

Proof. Each iteration of step 1 decreases the length of a path between any pair of vertices from ℓ to $\lceil \ell + 1 \rceil / 2$. Consequently, after $O(\log n)$ iterations the graph will consist of a collection of disjoint cliques, and the protocol will move on to the next step. By Lemma 20 (Cascading), if each iteration can be performed in time at most T assuming the augmented graph, then the full cascaded protocol runs in time at most $T^{O(\log n)}$. Since $T = \text{poly}(n)$ and each other step of the protocol is efficient, this implies that Protocol 5 runs in quasi-polynomial time.

Replacing the cascading of step 1 with the more efficient but computationally secure cascading of Lemma 21, we have the cascaded protocol runs in time $O(T\text{poly}(\lambda) \cdot \log n)$. Since each other step of the protocol is efficient, this implies that assuming one-way functions, we have a computationally-secure version of Protocol 5 that runs in quasi-polynomial time. □

4.7 The case $t = n - 2$

On account of the lower bound proven in [77], we note that $t = n - 2$ is the largest value of t which the question is non-trivial. In this section we present an improved efficient protocol for the special case $t = n-2$ for all networks G with A as the sender and B as the receiver whenever the network G satisfies G is $(2, A, B)$-unsplittable.

Theorem 11. *Consider an n-party network G, which contains A and B as two of the parties. Let $t = n - 2$. Protocol 6 is an efficient t-secure OT protocol with A as the sender and B as the receiver if and only if G is $(2, A, B)$-unsplittable.*

Protocol 6 is built upon the following structural aspect of the network G under consideration. We are only concerned with networks G that are $(2, A, B)$-unsplittable. This means that for any two sets vertices V_A and V_B such that $|V_A| = |V_B| = 2$, $A \in V_A$ and $B \in V_B$, there exists an edge crossing the sets V_A and V_B. In particular, by definition, this implies that for any two parties P_i, P_j where $i, j \geq 3$, the sub-network $G_{i,j}$ induced by the parties A, B, P_i and P_j is $(2, A, B)$-unsplittable. Flipping the argument on its head, this also means that for any two parties P_i, P_j, $G_{i,j}$ is $(2, P_i, P_j)$-unsplittable. Hence, we could try to obtain OT correlations between every pair of vertices P_i, P_j by running Protocol 5 on every $G_{i,j}$ for $n = 4$ parties. Notice that if these invocations were secure, then we would obtain an $(n-2)$-clique in the network after which we can execute Protocol 2 in order to obtain OT correlations between A and B. This describes Protocol 6. The only concern however is that each of the executions of Protocol 5 would be secure only if each of them contained at most two corrupt parties. While this need not be true in general, and hence we cannot leverage the security of the executions of Protocol 5, we will argue that Protocol 6 still remains secure against $t = n - 2$ corruptions.

Proof Intuition (Theorem 11): We take off from the description of Protocol 6 above. It is easy to see that in order to analyze the $(n - 2)$-security of Protocol 6, we need to analyze whether the invocations of Protocol 5 on sub-networks $G_{i,j}$ are secure. In particular, we know if at most two of the four parties in $G_{i,j}$ are honest, then

Protocol 6: $n-2$ corruptions

Preliminaries: Let $P_1 = A, P_2 = B, P_3, \ldots, P_n$ be the n parties, and let graph $G = (V, E)$ be the OT network among the parties. A has input (x_0, x_1), and B has input $b \in \{0, 1\}$.

Protocol:

1. For all pairs of parties $P_i, P_j \in V$ with $i, j \geq 3$ such that $\{P_i, P_j\} \notin E$:
   (a) Invoke Protocol 5 (or any 2-secure protocol for $n' = 4$) on the induced OT subgraph $G_{i,j} := G \cap \{P_i, Pj, A, B\}$ with independent random inputs $(r_0^{i,j}, r_1^{i,j})$ and $b^{i,j}$, to obtain OT correlations along edge $\{P_i, P_j\}$.
   (b) By virtue of cascading (Lemma 20), we can add edge $\{P_i, P_j\}$ to the graph G.[a]

   The OT network G now contains a $(n-2)$-clique among vertices $P_3, \ldots, P_n$.
2. Invoke Protocol 2 (t-clique) with input (x_0, x_1) and b.

[a] We will only have OT security over this edge when at least two of the parties P_i, P_j, A, B are honest, but we obtain the functionality of the edge regardless. We address security of the overall protocol in the proof.

that particular invocation of Protocol 5 is secure and yields secure OT correlations between the parties P_i and P_j. And then, appealing to Lemma 20, we have that the network G now possesses the edge $\{P_i, P_j\}$.

Note however that each $G_{i,j}$ has at least one honest party since at most one of A or B is corrupt (otherwise, there is nothing to prove). We now consider a sub-network $G_{i,j}$ in which three of the parties are corrupt. Since at least one of A or B is honest, this would mean that both P_i and P_j are corrupt. Thus, there is nothing to prove regarding the security of the invocation of Protocol 5 on $G_{i,j}$ since we are looking to establish OT correlations between P_i and P_j and they are both corrupt. Combining these claims, we have that each of the invocations of Protocol 5 is secure and yields secure OT correlations between the pairs of parties P_i, P_j for all $i, j \geq 3$. By virtue of Lemma 20, we obtain an $(n-2)$-clique in the network and the $(n-2)$-security of Protocol 2 with $t = n-2$ proves the $(n-2)$-security of Protocol 6.

Proof of Theorem 11. The "only if" part of theorem has been proved by virtue of the lower bound proven in Theorem 8 with $t = n-2$. We now prove the "if" part. Let $\mathcal{A}$ be a t-threshold adversary which corrupts parties T, $|T| \leq t = n-2$. If A and B are both corrupt then the uncorrupted parties receive no input, so the simulator $\mathcal{S}$ can

perfectly simulate the uncorrupted parties.

Otherwise we have that at least one of A or B is uncorrupted. We first show how to simulate step 1. Since G is $(2, A, B)$-unsplittable, for each pair of parties P_i, P_j considered in step 1, we have that $G_{i,j}$ is $(2, A, B)$-unsplittable. Since $\{P_i, P_j\} \notin E$, this implies that some vertex among A, B, P_3, and P_4 has degree at least 2 in $G_{i,j}$. In particular, we have $G_{i,j} \not\subseteq_{P_3,P_4} \Lambda_2^0$, where Λ_2^0 is labeled so that P_3 and P_4 are in separate components. By Theorem 9, we securely obtain OT correlations between P_3 and P_4 whenever at most two of the parties A, B, P_3, P_4 are corrupt.

We have that A and B are not both corrupt. Therefore, if P_3 and P_4 are not both corrupt, then at most two of the parties A, B, P_3, P_4 are corrupt. Consequently we securely obtain OT correlations between P_3 and P_4. On the other hand, P_3 and P_4 are both corrupted, then there is nothing to prove. More formally, there is a simulator which can perfectly simulate the invocation of Protocol 5 in step 1a by executing the protocol of the honest parties, since the honest parties receive no input. Consequently we securely obtain OT correlations between the corrupted parties P_3 and P_4. In both cases, by Lemma 20, we have an OT simulator which can simulate subsequent invocations of the OT channel between P_3 and P_4.

Therefore, by the end of step 1 we t-securely obtained an OT channel between every pair of parties P_i, P_j for $i, j \geq 3$. Then by Lemma 19, the invocation of Protocol 2 in step 2 is a t-secure OT protocol with A as the sender and B as the receiver. $\square$

4.8 The General Case: $t \geq n/2$

In this section, we investigate the question for general $t \geq n/2$. Note that from the protocols in Chapters 4.6 and 4.7 we already have tight answers for the cases $t = n/2$ and $t = n - 2$. We address the question from both ends of the spectrum, namely for t larger than $n/2$ and t smaller than $n - 2$. These analyses culminate in the descriptions of two distinct protocols using the protocols from Chapters 4.6 and 4.7 as their respective base cases. In particular, we note that the protocols we describe are efficient closer to their respective ends of the spectrum. That is, the protocol described in Chapter 4.8.1 is quasi-polynomially efficient[8] when $t = n/2 + \mathcal{O}(1)$, while the protocol described in Chapter 4.8.2 is efficient when $t = n - \mathcal{O}(1)$. We describe these protocols ahead and observe that the combination of these protocols yields one which is efficient under computational security when either $t = n/2 + \mathcal{O}(1)$ or $t = n - \mathcal{O}(1)$. We note that the problem of recognizing whether there exists a t-secure OT protocol is efficient in these cases, while the recognition problem for general n, t is coNP-complete.

4.8.1 General Protocol (Quasi-poly for $t = n/2 + \mathcal{O}(1)$)

We now describe a t-secure OT protocol for all networks G with A as the sender and B as the receiver whenever the network G is $(n - t, A, B)$-unsplittable. Notice that on account of the lower bound described in Chapter 4.4, this result is tight.

Theorem 12. *Consider an n-party network G which contains parties A and B. Let $t \geq n/2$. Protocol γ is a t-secure OT protocol with A as the sender and B as the receiver if and only if G is $(n - t, A, B)$-unsplittable. The protocol achieves perfect security and runs in quasi-polynomial time for $t = n/2 + \mathcal{O}(1)$. Assuming one-way functions, we can also obtain a protocol which achieves computational security and runs in polynomial time for $t = n/2 + \mathcal{O}(1)$.*

The main idea behind the protocol is recursion, that is, to reduce the problem of obtaining an OT protocol on an n-vertex graph with $t > n/2$ corrupted parties to

[8] or polynomially efficient under computational security

a number of instances of $(n-1)$-vertex graphs, most of which have at most $t-1$ corrupted parties. An important point to note, which we prove ahead, is that the $(n-1)$-vertex sub-graphs, say G', have structure similar to G, in the sense that, G' is $(n'-t', A, B)$-unsplittable if G is $(n-t, A, B)$-unsplittable, where $n' = n-1$ and $t' = t-1$. We can now try the natural strategy of recursing on these smaller problem instances and invoking an OT combiner to obtain the final protocol.

More precisely, the protocol constructs $n-2$ instances of subgraphs on $n-1$ vertices each where each one is obtained by deleting exactly one of the vertices other than A and B. It is these sub-graphs which preserve the structure in G, as described above. The candidate $n-2$ protocols, each run on one of the subgraphs, are $(t-1)$-secure OT protocols with A as the sender and B as the receiver. The final protocol is to simply run the protocol which combines all the candidate protocols. What remains to be proven is that a majority of the subgraphs defined actually possesses at most $t-1$ corrupt parties.

Proof Intuition (Theorem 12): We may assume that at least one of A or B is honest since otherwise there is nothing to prove. As described above, we wish to argue that a majority of the sub-graphs defined actually possesses at most $t-1$ corrupt parties. Note that this claim combined with the observation that these specially chosen sub-graphs preserve the structure of G (that is, for any of these sub-graphs G', G' is $(n-t, A, B)$-unsplittable if G is $(n-t, A, B)$-unsplittable) complete the proof. Furthermore, the claim can be proved rather easily as follows. Since $t > n/2$, if exactly t parties are corrupt then a majority of the sub-graphs definitely contain at most $t-1$ corrupt parties since A and B are not both corrupt. If strictly fewer than t parties are corrupt then all of the sub-graphs contain at most $t-1$ corrupt parties. In either case, for a majority of sub-graphs, at most $t-1$ of the parties are corrupt. Invoking Lemma 23 completes the argument.

We first present and prove a structure lemma.

Lemma 25. *Given graph $G = (V, E)$ and a vertex i, let G_i be the induced graph*

Protocol 7: General Protocol I

Preliminaries: Let $A, B, P_3, \ldots, P_n$ be the n parties in a network G and let $t \geq n/2$ be the maximum number of corruptions. A has input (x_0, x_1), and B has input $b \in \{0, 1\}$.

Protocol:

1. If $t = n/2$, then invoke Protocol 5 and halt.
2. Otherwise, run a combined protocol $\Pi^*(\Pi_3, \ldots, \Pi_n)$, where
   - For each $i \geq 3$, Π_i denotes the recursive invocation of this protocol on the $n-1$ parties excluding party P_i with the induced sub-network $G \setminus \{P_i\}$ and $t' = t - 1$ corruptions.

on the $n-1$ vertices $V \setminus \{i\}$. If G is $(n-t, A, B)$-unsplittable, then G_i is also $(n-t, A, B)$-unsplittable.

Proof. We will prove the contrapositive. Suppose that $G_i \subseteq_{A,B} \Lambda_{n-t}^{2t-n-1}$. This means there exists a partition of the vertex set of G_i as $V \setminus \{i\} = V_A \dot{\bigcup} V_S \dot{\bigcup} V_B$ where $A \in V_A$, $B \in V_B$, $|V_A| = |V_B| = n - t$ and $|V_S| = 2t - n - 1$. Now, we note that there exists a partition of the vertex set of G as $V = V_A \dot{\bigcup} V_S' \dot{\bigcup} V_B$ where $A \in V_A$, $B \in V_B$, $V_S' = V_S \cup \{i\}$ and hence $|V_A| = |V_B| = n - t$ and $|V_S'| = 2t - n$. This implies that $G \subseteq_{A,B} \Lambda_{n-t}^{2t-n}$, which is a contradiction. □

Using the lemma, we now prove Theorem 12.

Proof of Theorem 12: The "only if" part of theorem has been proved by virtue of the lower bound proven in Theorem 8. The efficiency claim follows immediately from Theorem 10. We now prove the "if" direction by induction on $2t - n$. The base case of $2t - n = 0$ follows from Theorem 9. Now assume for induction that the statement holds for $n' = n - 1, t' = t - 1$ with $t > n/2$.

Let $\mathcal{A}$ be a t-threshold adversary which corrupts parties T, $|T| \leq t$. If $\{A, B\} \subset T$ then the uncorrupted parties receive no input, so the simulator can perfectly simulate the uncorrupted parties. Consequently it suffices to consider the case $\{A, B\} \cap T \leq 1$. By Lemma 25, each sub-network $G \setminus \{P_i\}$ is $(n'-t', A, B)$-unsplittable. For each $i \geq 3$, let $T_i = T \setminus \{P_i\}$ denote the corrupted parties in sub-network $G \setminus \{P_i\}$. If $|T| < t$ then $|T_i| \leq |T| \leq t'$ for all i. Otherwise, $|T| = t$. In this case, since $\{A, B\} \cap T \leq 1$, it follows that at least $t-1$ of the $n-2$ parties $P_3, \ldots, P_n$ are corrupted. Since $t > n/2$, this is a majority of the parties $P_3, \ldots, P_n$. For each corrupted party P_i, $|T_i| = t'$.

Hence, in each case, we have that a majority of the sets $T_3, \ldots, T_n$ satisfy $|T_i| \leq t'$. Therefore by our inductive assumption, a majority of the protocols $\Pi_3, \ldots, \Pi_n$ in step 2 are secure against $\mathcal{A}$. Consequently, by Lemma 23, we have that the combined protocol Π^* is secure against $\mathcal{A}$. Hence, by induction, the theorem holds for all n, t with $t \geq n/2$. □

Corollary 1. *Let G be an n-party network. For $t \geq n/2$, we can t-securely generate OT correlations between all pairs of parties (thus, completing the OT network) if and only if the G is $(n-t)$-unsplittable.*

Proof. If G is $(n-t)$-unsplittable, then for all pairs of vertices A, B, G is $(n-t, A, B)$-unsplittable and hence from Theorem 12, we can generate OT correlations between A and B. Conversely, if $G \subseteq \Lambda_n^{2t-n}$, then choosing vertices $A \in V_A$ and $B \in V_B$, we have that $G \subseteq_{A,B} \Lambda_{n-t}^{2t-n}$ and hence Theorem 12 rules out the existence of such a protocol. Hence, there exist a pair of vertices between whom we cannot t-securely generate OT correlations. This completes the proof. □

Protocol 8: General protocol II

Preliminaries: Let $P_1 = A, P_2 = B, P_3, \ldots, P_n$ be the n parties in a network $G = (V, E)$. A has input (x_0, x_1), and B has input $b \in \{0, 1\}$. Let $k = n - t$.

Protocol:

1. Invoke Protocol 6 with $t' = n - 2$ on the n'-node network G' with inputs (x_0, x_1) and b, where $n' = \binom{n-2}{k-1} + 2$, and
   - S_{k-1} is the set of subsets of $\{P_3, \ldots, P_n\}$ of size $k - 1$.
   - The n' vertices of G' correspond to A, B, and the $\binom{n-2}{k-1}$ subsets of S_{k-1}.
   - The edges of G' are defined as follows. Two subsets $X, Y \in S_{k-1}$ will have an edge if either $X \cap Y \neq \emptyset$ or there exists a pair of parties $P_i \in X$ and $P_j \in Y$ with $\{P_i, P_j\} \in E$.
   - Invocation of OT over an edge $\{X, Y\}$ in G' with inputs (z_0, z_1) and c is performed as follows.
     - If $X \cap Y \neq \emptyset$, then choose some party $P_i \in X \cap Y$. $P_i \in X$ and hence knows (z_0, z_1); similarly, $P_i \in Y$ and knows c. Consequently P_i knows z_c, and sends it to the other members of set Y.
     - If $X \cap Y = \emptyset$, there is a pair of parties $P_i \in X, P_j \in Y$ such that $\{P_i, P_j\} \in E$. P_i knows (z_0, z_1) and P_j knows c, so they can invoke OT over the channel (P_i, P_j) in G, and P_j can then send the value z_c to the other members of Y.

4.8.2 General Protocol (Efficient for $t = n - \mathcal{O}(1)$)

We now describe another t-secure OT protocol for all networks G with A as the sender and B as the receiver whenever the network G is $(n - t, A, B)$-unsplittable. This protocol uses, in spirit, a reduction in the opposite sense that than the one described in Chapter 4.8.1. The protocol is efficient whenever $t = n - \mathcal{O}(1)$.

Theorem 13. *Consider an n-party network G, which contains A and B as two of the parties. Let $t \geq n/2$. Protocol 8 is a t-secure OT protocol with A as the sender and B as the receiver if and only if G is $(n - t, A, B)$-unsplittable. The protocol is efficient for $t = n - \mathcal{O}(1)$.*

The idea behind this protocol is the following. We blow up the network in order to obtain a large number, N, of parties apart from A and B such that at least one them is guaranteed to be honest. With this guarantee in mind, the protocol descrip-

tion is straightforward. We may assume that at least one of A and B is honest, as otherwise there is nothing to prove. Now, the total number of parties is $n' = N + 2$ and the number of honest parties is at least 2 since one of A or B and one of the other parties is guaranteed to be honest. This corresponds to the case that $t' = n' - 2$. We can now apply the protocol from Chapter 4.7. What remains to be discussed is the construction of these parties, a structural lemma that if the network G we began with were $(n-t, A, B)$-unsplittable, then the newly constructed network G' is $(2, A, B)$-unsplittable, and why G' is guaranteed to have at least two honest parties, in particular, why one of the parties other than A and B is honest.

Proof Intuition (Theorem 13): We first describe the new network generated by Protocol 8. The parties other than A and B in the newly constructed network are constructed as subsets of the parties in G other than A and B. The sizes of these subsets are $n-t-1$, that is, in G', the parties other than A and B are all $(n-t-1)$-size subsets of the remaining parties in G. It can be proved that this new network G' is $(2, A, B)$-unsplittable if G is $(n-t, A, B)$-unsplittable, where the edges of G' are as in described in Protocol 8 (Lemma 26). A party X, which is formed as a $(n-t-1)$-size subset will be considered honest if all constituent parties $P_i \in X$ are honest. Now, the reason for there being one honest party among these $(n-t-1)$-size subset parties is the following trivial observation. Since one of A and B is honest and at most t parties are corrupt, at least $n-t$ parties are honest and in particular, at least $n-t-1$ of the non-A, B parties must be honest. This means that one of the subsets is completely honest. In particular, one of the parties other than A and B are honest in G' and hence G' is guaranteed to have at least two honest parties. Combining these facts and invoking Theorem 11 completes the argument.

We will use the following structural lemma about the network G' constructed in Protocol 8.

Lemma 26. *If G is $(n-t, A, B)$-unsplittable, then G' is a $(2, A, B)$-unsplittable network on $n' = \binom{n-2}{n-t-1} + 2$ vertices, where G' is the network produced in Protocol 8.*

Proof. We prove the contrapositive. Assume that $G' \subseteq_{A,B} \Lambda_2^{n'-2}$. Then there exist vertices $X, Y \in S_{k-1}$ such that there are no edges in G' between any of the parties in $\{A, X\}$ and any of the parties in $\{B, Y\}$. In particular, $X \cap Y = \emptyset$, since otherwise $\{X, Y\}$ would be an edge of G'. This implies that we have $2k = 2(n-t)$ parties $\{A, B\} \cup X \cup Y$ such that there are no edges in G from the $n-t$ parties $\{A\} \cup X$ to any of the $n-t$ parties $\{B\} \cup Y$. By definition, this means that $G \subseteq_{A,B} \Lambda_{n-t}^{2t-n}$, which is a contradiction. □

Using this lemma, we will now prove the correctness of Protocol 8.

Proof of Theorem 13. As before, we only need to prove the "if" part. Let $\mathcal{A}$ be a t-threshold adversary which corrupts parties T, $|T| \leq t$. If A and B are both corrupt, then the honest parties have no input, so the simulator $\mathcal{S}$ can perfectly simulate the uncorrupted parties. If A and B are both honest, then $\mathcal{S}$ chooses arbitrary inputs x_0, x_1, b and runs the protocol. Since the only steps which depend at all on the inputs are on point-to-point channels between A and B, the view of the adversary in the real and ideal worlds is identical.

Otherwise, we have that at most $t-1$ of the parties $P_3, \ldots, P_n$ are corrupt and that either A or B is honest. In particular, for $k = n-t$, there are at least $k-1$ uncorrupted parties among $P_3, \ldots, P_n$. Consequently, S_{k-1} contains some set X consisting only of honest parties. We treat a party X in G' as honest if all constituent parties $P_i \in X$ are honest. Since A and B are not both corrupt, we have that G' contains at least two honest parties. By Lemma 26, G' is $(2, A, B)$-unsplittable. Consequently by Theorem 11 there is a simulator $\mathcal{S}'$ which simulates the roles of the honest parties in the invocation of Protocol 6 on G' in step 1. We define a simulator $\mathcal{S}$ for Protocol 8 which behaves exactly as an honest party for communication within each party $X \in S_{k-1}$ and invokes $\mathcal{S}'$ for any communication between parties in G'. The behavior of this protocol is identical to the behavior of $\mathcal{S}'$. Hence, by the correctness of simulator $\mathcal{S}'$, we have that the view of the adversary is identical in the real and ideal worlds. □

4.9 Bounding the number of edges in $\approx \frac{n}{2}$-unsplittable graphs.

In this section we discuss the minimum number of edges in a graph which is $(n-t)$-unsplittable for $t = \lfloor (n+1)/2 \rfloor$. We show that the minimum edge count is $n/2$ if n is even and $t = n/2$, and $(n+3)/2$ if n is odd and $t = (n+1)/2$. These bounds give the minimum number of OT channels required to obtain t-secure MPC among n parties in a network for $t = \lfloor (n+1)/2 \rfloor$. They constitute the first nontrivial cases, since no OT channels are needed in the case of an honest majority.

Theorem 14. *Let n be even and $t = n/2$. Then any $(n-t)$-unsplittable graph must contain at least t edges. This bound is tight.*

Proof. To show that the bound is tight, note that the t-claw graph (Figure 4-4(a)) (or any graph with a tree on $t+1$ vertices and no other edges) has t edges and contains a connected component consisting of $t+1 = n/2+1$ vertices, so it is $(n/2)$-unsplittable. We now show that every graph with fewer edges can be split.

Let G be a graph containing $t-1$ edges. Let m be the number of connected components of G, let $C_1, C_2, \ldots, C_m$ be the components in non-increasing order of size, and let $\alpha_i = |C_i|$, so that $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_m$ and $\sum_{i=1}^{m} \alpha_i = n$. G is $(n-t)$-splittable if and only if there is some subset $I \subset [n]$ such that $\sum_{i \in I} \alpha_i = n/2$, that is, some subset of the values α_i sum to $n/2$.

For all natural numbers x, let $a_x = |\{i : \alpha_i \geq x\}|$ denote the number of *connected components* with size at least x, and let $b_x = |\{i : \alpha_i < x\}|$ denote the number of *vertices* in connected components with size smaller than x. Note that a_x counts the components of certain sizes, while b_x counts the vertices contained in components of certain sizes. Since a component of size s must contain at least $s-1$ edges, for any $x > 1$ we have that $a_x \leq (t-1)/(x-1) < n/(2x-2)$. In particular, we have that $a_2 < n/2$, $a_3 < n/4$, and $a_4 < n/6$. Since G has at most $n/2 - 1$ edges, at least two vertices have degree zero, so $\alpha_1 = \alpha_2 = 1$. Consequently $b_2 \geq 2$. For any $x > 1$, we

have that

$$b_x \geq n - (a_x + t - 1) = \frac{n}{2} + 1 - a_x$$

This gives us that $\alpha_3 \leq n/3$ and hence $\alpha_i \leq n/2$ for all i.

Consider the following greedy algorithm. Initially let $S_2 = C_2$. For i from 3 to n, if $|S_{i-1} \cup C_i| \leq n/2$ then set $S_i = S_{i-1} \cup C_i$, and otherwise set $S_i = S_{i-1}$. We show that at the end of this loop, the set S_m will always have size $n/2$. Since $|C_m| \leq n/2$, the sum of the sizes of the other components is at least $n/2$, so if $|S_{i+1} \cup C_i| \leq n/2$ for every i considered in the loop then the loop must terminate with $|S_1| = n/2$. Otherwise there is some i such that $|S_{i-1} \cup C_i| > n/2$. Choose the last iteration i for which this is true. Since $i \geq 3$, we must have that $\alpha_i \leq \alpha_3 \leq n/3$. If $\alpha_i > 3$, then since $n/2 - |S_i| \leq \alpha_i - 1 \leq n/3 - 1$ and $b_4 > \frac{n}{3} + 1$, we reach a contradiction with the assumption that i is the last such iteration, since there are enough vertices in components $C_{i+1}, \ldots, C_m$ to make some subsequent $|S_{j-1} \cup C_j| > n/2$. Otherwise $\alpha_i \leq 3$, so since $n/2 - |S_i| \leq \alpha_i - 1 \leq 2$, so since $b_2 \geq 2$, we must have that $|S_m| = n/2$.

Consequently at the end of the loop we always have that $|S_m| = n/2$. Since S_m consists only of entire connected components, S_m consists of $n/2$ vertices with no edges to the rest of the graph, so $G \subset \Lambda^0_{n/2}$ cannot be $(n/2)$-unsplittable. □

Using similar techniques, we can also prove the following theorem.

Theorem 15. *Let n be odd and $t = (n+1)/2$. Then any $(n-t)$-unsplittable graph must contain at least $t+1$ edges. This bound is tight.*

Chapter 5

Infrastructure for Fair Computation

Fairness in secure multiparty computation refers to the property that if any party obtains the output of the computation, then so do all other parties. In this work, we introduce a new 2-party primitive $\mathcal{F}_{\mathsf{SyX}}$ ("synchronizable fair exchange") and show that it is complete for realizing any n-party functionality with fairness *in a setting where all n parties are pairwise connected by independent instances of $\mathcal{F}_{\mathsf{SyX}}$.*

In the $\mathcal{F}_{\mathsf{SyX}}$-hybrid model, the two parties load *$\mathcal{F}_{\mathsf{SyX}}$ with some input, and following this, either party can* trigger *$\mathcal{F}_{\mathsf{SyX}}$ with a suitable "witness" at a later time to receive the output from $\mathcal{F}_{\mathsf{SyX}}$. Crucially the other party also receives output from $\mathcal{F}_{\mathsf{SyX}}$ when $\mathcal{F}_{\mathsf{SyX}}$ is triggered. The trigger witnesses allow us to* synchronize *the trigger phases of multiple instances of $\mathcal{F}_{\mathsf{SyX}}$, thereby aiding in the design of fair multiparty protocols. Additionally, a pair of parties may* reuse *a single* a priori *loaded instance of $\mathcal{F}_{\mathsf{SyX}}$ in any number of multiparty protocols (possibly involving different sets of parties).*

The results presented in this chapter also appear in a joint work with Ranjit Kumaresan and Adam Sealfon [97].

5.1 Preliminaries

5.1.1 Adversaries

We consider security against *malicious static t-threshold adversaries*, that is, adversaries that corrupt a set of at most t parties, where $0 \leq t < n$[1]. We assume the adversary to be *malicious*. That is, the corrupted parties may deviate arbitrarily from an assigned protocol.

5.1.2 Fairness versus Guaranteed Output Delivery

We recall here some of the results from [42].

Lemma 27. *[42] Consider n parties $P_1, \ldots, P_n$ in a model without a broadcast channel. Then, there exists a functionality $f : \mathcal{X}^n \to \mathcal{Y}^n$ such that f cannot be securely computed with guaranteed output delivery in the presence of t-threshold adversaries for $n/3 \leq t < n$.*

Lemma 28. *[42] Consider n parties $P_1, \ldots, P_n$ in a model with a broadcast channel. Then, assuming the existence of one-way functions, for any functionality $f : \mathcal{X}^n \to \mathcal{Y}^n$, if there exists a protocol π which securely computes f with fairness, then there exists a protocol π' which securely computes f with guaranteed output delivery.*

Lemma 29. *[42] Consider n parties $P_1, \ldots, P_n$ in a model with a broadcast channel. Then, assuming the existence of one-way functions, for any functionality $f : \mathcal{X}^n \to \mathcal{Y}^n$, if there exists a protocol π which securely computes f with fairness, then there exists a protocol π' which securely computes f with fairness and does not make use of the broadcast channel.*

[1] Note that when $t = n$, there is nothing to prove.

Preliminaries: $x_1, x_2 \in \{0,1\}^*$; f_1, f_2 are 2-input, 2-output functionalities. The functionality proceeds as follows:

- Upon receiving inputs (x_1, f_1) from P_1 and (x_2, f_2) from P_2, check if $f = f_1 = f_2$. If not, abort. Else, send $f^1(x_1, x_2)$ to P_1 and $f^2(x_1, x_2)$ to P_2.

Figure 5-1: The ideal functionality $\mathcal{F}_{\mathsf{2PC}}$.

5.2 Synchronizable Exchange

We are interested in solving the problem of securely computing functionalities with fairness, most commonly referred to as fair secure computation. We begin with the case of two parties. It is known that fair two-party secure computation is impossible in the standard model as well as in the $(\mathcal{F}_{\mathsf{bc}}, \mathcal{F}_{\mathsf{OT}})$-hybrid model [41]. This result generalizes to the setting of n parties that are trying to compute in the presence of a t-threshold adversary for any $n/2 \leq t < n$.

One could define the ideal functionality, $\mathcal{F}_{\mathsf{2PC}}$ as in Figure 5-1. Clearly, any 2-party functionality can be securely computed with fairness in the $(\mathcal{F}_{\mathsf{2PC}}, \mathsf{fair})$-hybrid model. One can then ask the following question in the context of $n > 2$ parties:

Consider n parties $P_1, \ldots, P_n$ in the OT-broadcast *model. Does there exist a protocol that securely computes $\mathcal{F}_{\mathsf{MPC}}$ with fairness in the $(\mathcal{F}_{\mathsf{2PC}}, \mathsf{fair})$-hybrid model?*

We are interested in security in the presence of a t-threshold adversary for any $n/2 \leq t < n$. While we do not know the answer to this question, it seems that the answer to this question would be negative. The intuition for this is that the various invocations of the ideal functionality $\mathcal{F}_{\mathsf{2PC}}$ cannot "synchronize" with each other and thus we would run into issues similar to the those highlighted by the impossibility result in [41], namely, some party/parties obtain information about the output of the protocol before the others and if these parties were corrupt, they may choose to abort the protocol without the honest parties receiving their outputs.

Equipped with this intuition, we propose the primitive, $\mathcal{F}_{\mathsf{SyX}}$, which we call "synchronizable exchange". We define the ideal functionality for $\mathcal{F}_{\mathsf{SyX}}$ in Figure 5-2. We associate the type $\mathsf{g.d.}$ to the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$ when working in the $\mathcal{F}_{\mathsf{SyX}}$-hybrid

Preliminaries: $x_1, x_2 \in \{0,1\}^*$; f_1, f_2 are 2-input, 2-output functions; ϕ_1, ϕ_2 are boolean predicates. The functionality proceeds as follows:

- **Load phase.** Upon receiving inputs $(x_1, f = (f_1, f_2, \phi_1, \phi_2))$ from P_1 and (x_2, f') from P_2, check if $f = f'$. If not, abort. Else, compute $f_1(x_1, x_2)$. If $f_1(x_1, x_2) = \perp$[a], abort. Else, send $f_1(x_1, x_2)$ to both parties, and go to next phase.
- **Trigger phase.** Upon receiving input w from party P_i, check if $\phi_i(w) = 1$. If yes, then send $(w, f_2(x_1, x_2, w))$ to both P_1 and P_2.

[a] We crucially require that $\perp$ is a special symbol different from the empty string. We use $\perp$ as a means of signalling that the load phase of $\mathcal{F}_{\mathsf{SyX}}$ did not complete successfully. We will however allow parties to attempt to invoke the load phase of the functionality at a later time. However, as we proceed, we will also have our functionality be clock-aware and thus only accept invocations to the load phase until a certain point in time. After the load phase times out, the functionality is rendered completely unusable. Similarly, if the load phase has been completed successfully, a clock-oblivious version of the functionality can be triggered at any point in time as long as a valid witness is provided, no matter the number of failed attempts. The clock-aware version of the functionality, however, will only accept invocations of the trigger phase until a certain point in time. After the trigger phase times out, the functionality is rendered completely unusable.

Figure 5-2: The ideal functionality $\mathcal{F}_{\mathsf{SyX}}$.

model. When interacting with this functionality, parties first submit their inputs to $\mathcal{F}_{\mathsf{SyX}}$ which then gives them a "receipt" acknowledging the end of the input submission phase. Following this, the functionality simply waits for a trigger from one of the parties. Once the trigger is received (we specify conditions for the validity of a trigger), then the functionality will deliver the outputs according to the specification. In the formal specification, we allow parties P_1, P_2 to submit two functions f_1, f_2 and two Boolean predicates (that check validity of a trigger value) ϕ_1, ϕ_2 along with their inputs x_1, x_2. $\mathcal{F}_{\mathsf{SyX}}$ then computes $f_1(x_1, x_2)$ and sends this value as a "receipt" that the input submission phase has ended. The actual output of the computation is $f_2(x_1, x_2)$ and this will be provided to the parties at the end of the trigger phase. Note that the trigger phase can be activated by either party P_i. However, P_i would need to provide a "witness" w that satisfies ϕ_i.

Note that $\mathcal{F}_{\mathsf{SyX}}$ is at least as strong as $\mathcal{F}_{\mathsf{2PC}}$. In order to realize $\mathcal{F}_{\mathsf{2PC}}$ in the $\mathcal{F}_{\mathsf{SyX}}$-hybrid model, we set $f_1 = \varepsilon$ (the empty string), $f_2 = f$, $\phi_1 = \phi_2 = 1$. The hope in defining this reactive functionality, however, is to achieve synchronization of multiple invocations of the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$. In a nutshell, the synchronization of multiple invocations of the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$ is enabled by the "trigger" phase of functionality. We will be using f_1 to provide a proof to parties other than P_1, P_2 that the input submission phase has ended for parties P_1, P_2. In other words, if we wish to synchronize multiple invocations of the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$, we set the witness for the trigger phase of each of the invocations to be the set of all receipts obtained from the inputs phases of the invocations. The set of all receipts acts as a proof that every invocation of the ideal functionality completed its load phase successfully. We use this feature of $\mathcal{F}_{\mathsf{SyX}}$ in order to design a protocol for fair secure computation.

Multiple Triggers and Witnesses. Note that as described, the load phase of the functionality $\mathcal{F}_{\mathsf{SyX}}$ can only be executed successfully once. And, once it has been successfully executed, the functionality is in the trigger phase. However, whilst in the trigger phase, the primitive may be triggered any number of times successfully or unsuccessfully. Furthermore, triggering the primitive with different witnesses may

actually produce different outputs, as modeled by having the output f_2 depend on the witness w in addition to x_1, x_2. This will be important for us in Section 5.5.

Clock-awareness. A technicality that arises in the protocol is that of guaranteed termination. Specifically, we will need our ideal functionality to be "clock-aware". The issue of modeling a trusted clock has been studied in the literature. In this work, we stick to the formalism outlined in [114]. We recall the main ideas here. We assume a synchronous execution model, where protocol execution proceeds in atomic time steps called *rounds*. We assume that the trusted clocks of attested execution processors and the network rounds advance at the same rate. It is easy to adapt our model and results if the trusted clocks of the processors and the network rounds do not advance at the same rate. In each round, the environment must activate each party one by one, and therefore, all parties can naturally keep track of the current round number. We will use the symbol r to denote the current round number. A party can perform any fixed polynomial (in λ) amount of computation when activated, and send messages. We consider a synchronous communication model where messages sent by an honest party will be delivered at the beginning of the next round. Whenever a party is activated in a round, it can read a buffer of incoming messages to receive messages sent to itself in the previous round. To model trusted clocks in attested execution processors, we will provide a special instruction such that ideal functionalities, in particular $\mathcal{F}_{\mathsf{SyX}}$ can query the current round number. We say that a functionality $\mathcal{F}$ is *clock-aware* if the functionality queries the local time; otherwise we say that the functionality $\mathcal{F}$ is *clock-oblivious*. For the rest of the work, we will always assume that $\mathcal{F}_{\mathsf{SyX}}$ is clock-aware. We would also like to stress that we require only relative clocks - in other words, trusted clocks of all functionalities need not be synchronized, since our protocol will only make use of the number of rounds that have elapsed since initialization. Therefore, we will assume that when a functionality reads the clock, a relative round number since the first invocation of the functionality is returned. Thus, when working in this model, we assume that every party and every invocation of the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$ has access to a variable r that reflects the current round

Preliminaries: $x_1, x_2 \in \{0,1\}^*$; f_1, f_2 are 2-input, 2-output functions; ϕ_1, ϕ_2 are boolean predicates; r denotes the current round number; $\mathsf{INPUT\_TIMEOUT} < \mathsf{TRIGGER\_TIMEOUT}$ are round numbers representing time outs. The functionality proceeds as follows:

- **Load phase.** If $r > \mathsf{INPUT\_TIMEOUT}$, abort. Otherwise, upon receiving inputs of the form $(x_1, f = (f_1, f_2, \phi_1, \phi_2))$ from P_1 and (x_2, f') from P_2, check if $f = f'$. If not, abort. Else, compute $f_1(x_1, x_2, r)$. If $f_1(x_1, x_2, r) = \perp$, abort. Else, send $f_1(x_1, x_2, r)$ to both parties, and go to next phase.
- **Trigger phase.** If $r > \mathsf{TRIGGER\_TIMEOUT}$, abort. Otherwise, upon receiving input w from party P_i, check if $\phi_i(w, r) = 1$. If yes, send $(w, f_2(x_1, x_2, w, r))$ to both P_1 and P_2.

Figure 5-3: The clock-aware ideal functionality $\mathcal{F}_{\mathsf{SyX}}$.

number. More generally, every function and predicate that is part of the specification of $\mathcal{F}_{\mathsf{SyX}}$ may also take r as an input. Finally, the functionality may also time out after a pre-programmed amount of time. We describe this clock-aware functionality in Figure 5-3.

Infinite Timeouts. We note here that it is possible to set either one or both of $\mathsf{INPUT\_TIMEOUT}$ and $\mathsf{TRIGGER\_TIMEOUT}$ to be ∞. What this means is that the functionality retains its state even if it goes offline. Its state would comprise (x_1, x_2) and which phase (input or trigger) it is currently in. We also require that if the functionality does go offline and come back online, it can still access the current value of the clock, r. The only time we use this feature of the primitive is in Section 5.5 where we are able to preprocess the functionality for an unbounded number of fair multiparty computations that would be run in the future. In this case, we would need to trigger this functionality whenever an adversary attempts to break fairness. Since we have no bound on how many computations we will run, we will set the $\mathsf{TRIGGER\_TIMEOUT}$ to be ∞. In practice, one could also just set $\mathsf{TRIGGER\_TIMEOUT}$ to be a very large number. We stress however that the functionality is stateful and able to read time irrespective of whether it goes offline intermittently.

5.3 Our contributions

We introduce our main result:

Theorem (**Main Theorem for Fair Computation Infrastructure**). *Consider n parties $P_1, \ldots, P_n$ in the* point-to-point *model. Then, assuming the existence of one-way permutations, there exists a protocol π in the programmable random oracle model which securely preprocesses for and computes an arbitrary (polynomial) number of instances of $\mathcal{F}_{\mathsf{MPC}}$ with fairness in the presence of t-threshold adversaries for any $0 \leq t < n$ in the $\mathcal{F}_{\mathsf{SyX}}$-hybrid model.*

Organization: Roadmap to our final result

The remainder of this chapter is organized as follows:

- In Section 5.4, we present our protocol for fair secure computation in the $\mathcal{F}_{\mathsf{SyX}}$-hybrid model.
- Finally, in Section 5.5, we show how $\mathcal{F}_{\mathsf{SyX}}$ can be preprocessed.

5.4 Fair Secure Computation in the $\mathcal{F}_{\mathsf{SyX}}$-hybrid model

In this section, we will describe how a set of n parties in the OT-network model that have pairwise access to the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$ can implement n-party fair secure function evaluation. To begin with, we will assume that the n-parties are in the point-to-point model and develop a protocol in the $(\mathcal{F}_{\mathsf{bc}}, \mathcal{F}_{\mathsf{MPC}}, \mathcal{F}_{\mathsf{SyX}})$-hybrid model. We first provide some intuition for our construction.

5.4.1 Intuition

We first start with the 3-party case as a warm-up. Let P_1, P_2, and P_3 be the three parties with inputs x_1, x_2 and x_3 respectively. For $i, j \in \{1, 2, 3\}$ with $i < j$, we have that parties P_i and P_j have access to the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$. In particular, let $\mathcal{F}_{\mathsf{SyX}}^{i,j}$ represent the instantiation of the $\mathcal{F}_{\mathsf{SyX}}$ functionality used by parties P_i, P_j. We wish to perform fair secure function evaluation of some 3-input 3-output functionality F.

Reduction to single output functionalities. Let $(y_1, y_2, y_3) \xleftarrow{\$} F(x_1, x_2, x_3)$ be the output of the function evaluation. We define a new four input single output functionality F' such that

$$F'(x_1, x_2, x_3, z) = F^1(x_1, x_2, x_3) \| F^2(x_1, x_2, x_3) \| F^3(x_1, x_2, x_3) \oplus z = y_1 \| y_2 \| y_3 \oplus z$$

where $z = z_1 \| z_2 \| z_3$ and $|y_i| = |z_i|$ for all $i \in [3]$. The idea is that the party P_i will obtain $z' = F'(x_1, x_2, x_3, z)$ and z_i. Viewing $z' = z'_1 \| z'_2 \| z'_3$ where $|z'_i| = |z_i|$[2] for all $i \in [3]$, party P_i reconstructs its output as

$$y_i = z_i \oplus z'_i$$

[2]We may assume without loss of generality that the lengths of the outputs of each party are known beforehand.

Now, we may assume that the input of party P_i is (x_i, z_i) (or we can generate random z_is as part of the computation) which determines z. It thus suffices to consider fair secure function evaluation of single output functionalities.

Reduction to fair reconstruction. We will use ideas similar to [73, 98] where instead of focusing on fair secure evaluation of an arbitrary function, we only focus on fair reconstruction of an additive secret sharing scheme. The main idea is to let the three parties run a secure computation protocol that computes the output of the secure function evaluation on the parties' inputs, and then additively secret shares the output. Given this step, fair secure computation then reduces to fair reconstruction of the underlying additive secret sharing scheme.

The underlying additive secret sharing scheme. We use an additive secret sharing of the output y. Let the shares be y_i for $i \in [3]$. That is, it holds that

$$y = \bigoplus_{i \in [3]} y_i$$

We would like party P_i to reconstruct y by obtaining all shares y_i for each $i \in [3]$. Initially, each party P_i is given y_i. Therefore, each party P_i only needs to obtain y_j and y_k for $j, k \neq i$.

Fair reconstruction via $\mathcal{F}_{\mathsf{SyX}}$. We assume that the secure function evaluation also provides commitments to all the shares of the output. That is, P_i receives $(y_i, \overrightarrow{c})$ for each $i \in [3]$, where Com is a commitment scheme and

$$\overrightarrow{c} = \{\mathsf{Com}(y_1), \mathsf{Com}(y_2), \mathsf{Com}(y_3)\}$$

Furthermore, we assume that each party P_i picks its own verification key vk_i and signing key sk_i with respect to a signature scheme with a signing algorithm Sign and a verification algorithm Verify, for each $i \in [3]$. All parties then broadcast their

verification keys to all parties. Let

$$\overrightarrow{\mathsf{vk}} = \{\mathsf{vk}_1, \mathsf{vk}_2, \mathsf{vk}_3\}$$

Each pair of parties P_i and P_j then initializes $\mathcal{F}_{\mathsf{SyX}}^{i,j}$ with inputs

$$x_i = \left(\overrightarrow{\mathsf{vk}}, \mathsf{sk}_i, y_i, \overrightarrow{c}\right)$$

and

$$x_j = \left(\overrightarrow{\mathsf{vk}}, \mathsf{sk}_j, y_j, \overrightarrow{c}\right)$$

The function f_1 checks if both parties provided the same value for $\overrightarrow{\mathsf{vk}}$, $\overrightarrow{c}$ and checks the y_i and y_j are valid openings to the corresponding commitments. It also checks that the signing keys provided by the parties are consistent with the corresponding verification keys (more precisely, we will ask for randomness provided to the key generation algorithm of the signature scheme). If all checks pass, then $\mathcal{F}_{\mathsf{SyX}}^{i,j}$ computes

$$\sigma_{i,j} = \mathsf{Sign}((i,j); \mathsf{sk}_i) \| \mathsf{Sign}((i,j); \mathsf{sk}_j)$$

This completes the description of f_1.

Synchronization step. The output of f_1 for each of the $\mathcal{F}_{\mathsf{SyX}}^{i,j}$ will provide a way to synchronize all $\mathcal{F}_{\mathsf{SyX}}$ instances. By synchronization, we mean that an $\mathcal{F}_{\mathsf{SyX}}^{i,j}$ instance cannot be triggered unless every other instance has already completed its load phase successfully. We achieve synchronization by setting the predicate $\phi_k(w)$ (for $k \in \{i,j\}$) to output 1 if and only if w consists of all signatures

$$\overrightarrow{\sigma} = \{\sigma_{i,j}\}_{i<j}$$

That is, each instance $\mathcal{F}_{\mathsf{SyX}}^{i,j}$ will accept the same trigger $w = \overrightarrow{\sigma}$. We define f_2 to simply output both y_i and y_j to both parties if $\phi_k(w) = 1$.

Protocol intuition. We briefly discuss certain malicious behaviors and how we handle them. From the description above, it is clear that parties have no information about the output until one of the $\mathcal{F}_{\mathsf{SyX}}$ instances is triggered. Furthermore, note that this implies that the corrupt parties must successfully complete the load phases of the instances of $\mathcal{F}_{\mathsf{SyX}}$ that it shares with all of the honest parties in order to obtain the witness that can be used to trigger the $\mathcal{F}_{\mathsf{SyX}}$ instances. Following the load phases of all of the $\mathcal{F}_{\mathsf{SyX}}$ instances, we ask each party to broadcast the receipt $\sigma_{i,j}$ obtained from $\mathcal{F}_{\mathsf{SyX}}^{i,j}$. Now suppose parties P_i and P_j are both dishonest, and suppose they do not broadcast $\sigma_{i,j}$. Note also that since P_i and P_j collude, they do not need the help of $\mathcal{F}_{\mathsf{SyX}}$ to compute $\sigma_{i,j}$. Since honest P_k does not know the synchronizing witness $\overrightarrow{\sigma}$, it will not be able to trigger any of the $\mathcal{F}_{\mathsf{SyX}}$ instances. However, note that for the adversary to learn the output of the computation, the corrupt party P_i (without loss of generality) will need to trigger $\mathcal{F}_{\mathsf{SyX}}^{i,k}$ to obtain P_k's share of the key. However, once P_i triggers $\mathcal{F}_{\mathsf{SyX}}^{i,k}$, it follows that P_k would obtain the synchronizing witness $\overrightarrow{\sigma}$ using which it can trigger both $\mathcal{F}_{\mathsf{SyX}}^{i,k}$ and $\mathcal{F}_{\mathsf{SyX}}^{j,k}$ and learn its output.

Termination. The protocol as described up until this point does not have guaranteed termination. In particular, the honest parties will need to wait for the corrupted parties to broadcast their receipts in order to be able to trigger the instances of $\mathcal{F}_{\mathsf{SyX}}$ and obtain the output. Time outs do not help in this case as the adversary may simply wait until the last moment to trigger instances of $\mathcal{F}_{\mathsf{SyX}}$ and obtain their outputs leaving only insufficient time for the honest parties to trigger their instances of $\mathcal{F}_{\mathsf{SyX}}$ and obtain their outputs. In order to ensure termination, we make use of the clock. The main invariant that we want to guarantee is that if an instance of $\mathcal{F}_{\mathsf{SyX}}$ involving an (honest) party is triggered, then every other instance of $\mathcal{F}_{\mathsf{SyX}}$ that the (honest) party is involved in, also needs to be triggered. One way to implement this idea is to assume that all instances of $\mathcal{F}_{\mathsf{SyX}}$ time out after

$$T = \binom{3}{2} = 3$$

rounds. Furthermore, an instance of $\mathcal{F}_{\mathsf{SyX}}$ accepts triggers in some round $\tau \in [T]$ (that is, until it times out) if and only if you provide a proof that $t-1$ other instances of $\mathcal{F}_{\mathsf{SyX}}$ were triggered until now. As before, we will have $\mathcal{F}_{\mathsf{SyX}}$ leak the triggering witness to the parties. Thus, if $\mathcal{F}_{\mathsf{SyX}}^{i,j}$ is triggered in some round t, then P_i (and/or P_j) can trigger all the other $\mathcal{F}_{\mathsf{SyX}}^{i,k}$ (and/or $\mathcal{F}_{\mathsf{SyX}}^{j,k}$) channels that it is involved in, in round $\tau + 1$.

Suppose some honest party, say P_i, does not obtain the output of the computation while the adversary has learned the output. Since the adversary learned the output, this means that the adversary triggered $\mathcal{F}_{\mathsf{SyX}}^{i,j}$ for some j (otherwise the adversary would not have learnt y_i and would not have received the output). That means P_i would have been able to trigger all the other channels that it is involved in and generate the final output in the next round. The only issue with this argument would be when $\mathcal{F}_{\mathsf{SyX}}^{i,j}$ was triggered last, that is, in round $\tau = T$. However this is not possible since until this time, at most $T - n + 1 < T - 1$, assuming $n \geq 3$, instances of $\mathcal{F}_{\mathsf{SyX}}$ could have been be triggered. This is because $n - 1$ instances of $\mathcal{F}_{\mathsf{SyX}}$ must be left untriggered in round $\tau = T - 1$ since the honest party didn't get its output.

Reducing the duration of time outs. A more clever solution will allow us to terminate within $T = n$ rounds. In order to trigger an instance of $\mathcal{F}_{\mathsf{SyX}}$ in some round $\tau \in [T]$, you must provide a proof that other instances of $\mathcal{F}_{\mathsf{SyX}}$ involving at least τ different parties have been triggered. Consider the first round τ in which P_i is an honest party and $\mathcal{F}_{\mathsf{SyX}}^{i,j}$ is triggered for some j. If $\tau = 1$, then the single invocation already gives a proof that channels involving two parties, namely, i, j, have been triggered. Otherwise, by assumption, proofs of invocations of instances of $\mathcal{F}_{\mathsf{SyX}}$ involving τ different parties were needed to trigger $\mathcal{F}_{\mathsf{SyX}}^{i,j}$. But P_i is not one of these parties as τ is the first round in which $\mathcal{F}_{\mathsf{SyX}}^{i,j}$ was triggered for any j. Consequently, P_i, on this invocation, obtains a proof that instances of $\mathcal{F}_{\mathsf{SyX}}$ involving at least $\tau + 1$ parties have been triggered, and can thus trigger all channels in round $\tau + 1$. As before, the only gap in the argument is the case $\tau = T$. One can trivially see that since $\mathcal{F}_{\mathsf{SyX}}^{i,j}$ has not been triggered for any j, it is impossible to obtain a proof that

instances of $\mathcal{F}_{\mathsf{SyX}}$ involving at least T different parties have been triggered.

Simulation. We look ahead for the issues that come up while trying to prove security, that is, during the simulation. The simulator will release to the adversary, the adversary's shares of the output, which can be simulated. But, it also releases commitments to all the shares of the output. Since the simulator does not know the output *apriori*, and does not know whether the adversary is going to abort the computation, in which case, no one knows the output, it has to produce commitments that it can later *equivocate*. In this context, we use, not regular commitments, but honest-binding commitments. In this case, the simulator can produce commitments to garbage but can later open them to be *valid* shares of the output. The rest of the computations can be trivially simulated. The only other detail to be looked into is that of the clock. We need to determine if the adversary has decided to abort the computation, that is, if the adversary is going to receive the output of the computation of not. This is done by noticing if and when the adversary decides to trigger the instances of $\mathcal{F}_{\mathsf{SyX}}$ that involve honest parties. We know that if the adversary ever triggers an instance of $\mathcal{F}_{\mathsf{SyX}}$ involving an honest party, then all parties will be in a position to receive the output. Thus, the simulator can simply run the adversary to determine whether it has decided to enable parties to obtain the output, in which the simulator would ask the trusted party to continue, or not, in which case the simulator would ask the trusted party to abort.

5.4.2 Protocol

We now present the protocol for fair secure computation in the $(\mathcal{F}_{\mathsf{bc}}, \mathcal{F}_{\mathsf{MPC}}, \mathcal{F}_{\mathsf{SyX}})$-hybrid model.

Preliminaries. F is the n-input n-output functionality to be computed; x_i is the input of party P_i for $i \in [n]$; $\mathcal{F}_{\mathsf{SyX}}^{a,b}$ represents the instantiation of the $\mathcal{F}_{\mathsf{SyX}}$ functionality used by parties P_a, P_b with time out round numbers $\mathsf{INPUT\_TIMEOUT} = 0$ and $\mathsf{TRIGGER\_TIMEOUT} = n$ for $a < b$, where $a, b \in [n]$; $\left(\mathsf{Com}, \mathsf{Open}, \widetilde{\mathsf{Com}}, \widetilde{\mathsf{Open}}\right)$ is an

honest-binding commitment scheme; $\mathcal{V} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ is a signature scheme; r denotes the current round number.

Protocol. The protocol Π_{FMPC} proceeds as follows:

- Define F' to the be the following n-input n-output functionality: On input $\overrightarrow{x} = (x_1, \ldots, x_n)$:
  - Let $(y_1, \ldots, y_n) = F(x_1, \ldots, x_n)$ and let

    $$y = y_1 \| \ldots \| y_n$$

    Sample random strings $\alpha_i \xleftarrow{\$} \{0,1\}^*$ such that $|\alpha_i| = |y_i|$ for each $i \in [n]$. Let

    $$\alpha = \alpha_1 \| \ldots \| \alpha_n$$

    Let $z = y \oplus \alpha$.
  - Sample a random additive n-out-of-n secret sharing $z_1, \ldots, z_n$ of z such that

    $$z = \bigoplus_{i \in [n]} z_i$$

  - Compute commitments along with their openings $(c_i^z, \omega_i^z) \xleftarrow{\$} \mathsf{Com}(z_i)$ to each of the shares z_i for each $i \in [n]$. Let

    $$\overrightarrow{c^z} = (c_1^z, \ldots, c_n^z)$$

  - Sample random proof values $\pi_1, \ldots, \pi_n \xleftarrow{\$} \{0,1\}^\lambda$. Compute commitments along with their openings $(c_i^\pi, \omega_i^\pi) \xleftarrow{\$} \mathsf{Com}(\pi_i)$ to each of the proof values π_i for each $i \in [n]$. Let

    $$\overrightarrow{c^\pi} = (c_1^\pi, \ldots, c_n^\pi)$$

  - Party P_i receives output $\left(\alpha_i, \overrightarrow{c^z}, \omega_i^z, z_i, \overrightarrow{c^\pi}, \omega_i^\pi, \pi_i\right)$ for each $i \in [n]$.

- The parties invoke the ideal functionality $\mathcal{F}_{\mathsf{MPC}}$ with inputs $((x_1, F'), \ldots, (x_n, F'))$. If the ideal functionality returns $\perp$ to party P_i, then P_i aborts for any $i \in [n]$[3]. Otherwise, party P_i receives output $\left(\alpha_i, \overrightarrow{c^z}, \omega_i^z, z_i, \overrightarrow{c^\pi}, \omega_i^\pi, \pi_i\right)$ for each $i \in [n]$.

- Each party P_i, for each $i \in [n]$, picks a random $\beta_i \in \{0,1\}^*$ and uses this randomness to pick a signing and verification key pair $(\mathsf{sk}_i, \mathsf{vk}_i) = \mathcal{V}.\mathsf{Gen}(1^\lambda; \beta_i)$. It then invokes the ideal functionality $\mathcal{F}_{\mathsf{bc}}$ and broadcasts vk_i to all other parties. If it does not receive vk_j for all $j \neq i$, it aborts. Otherwise, it obtains

$$\overrightarrow{\mathsf{vk}} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$$

- For each $a, b \in [n]$ with $a < b$, define the following functions.

  - Let $f_1^{a,b}$ be the function that takes as input (γ, γ') and parses

$$\gamma = \left(\overrightarrow{\mathsf{vk}}, \mathsf{sk}, \beta, \overrightarrow{c^z}, \omega^z, z, \overrightarrow{c^\pi}, \omega^\pi, \pi\right)$$

    and

$$\gamma' = \left(\overrightarrow{\mathsf{vk}}', \mathsf{sk}', \beta', \overrightarrow{c^z}', \omega^{z\prime}, z', \overrightarrow{c^\pi}', \omega^{\pi\prime}, \pi'\right)$$

    It checks that:

    * $\overrightarrow{\mathsf{vk}} = \overrightarrow{\mathsf{vk}}'$, $\overrightarrow{c^z} = \overrightarrow{c^z}'$, $\overrightarrow{c^\pi} = \overrightarrow{c^\pi}'$
    * $(\mathsf{sk}, \mathsf{vk}_a) = \mathcal{V}.\mathsf{Gen}(1^\lambda; \beta)$, $(\mathsf{sk}', \mathsf{vk}_b) = \mathcal{V}.\mathsf{Gen}(1^\lambda; \beta')$
    * $\mathsf{Open}(c_a^z, \omega^z, z) = \mathsf{Open}(c_b^z, \omega^{z\prime}, z') = 1$
    * $\mathsf{Open}(c_a^\pi, \omega^\pi, \pi) = \mathsf{Open}(c_b^\pi, \omega^{\pi\prime}, \pi') = 1$

[3] In the $\mathcal{F}_{\mathsf{OT}}$-hybrid model, let $\pi_{F'}$ denote the protocol for the functionality F' defined in Lemma 16. The parties execute $\pi_{F'}$. If the execution of $\pi_{F'}$ aborts, we are assuming that all (honest) parties are aware of the round when the execution of $\pi_{F'}$ aborts, that is, when the adversary has decided to abort the execution of $\pi_{F'}$. Since we are working in the $\mathcal{F}_{\mathsf{MPC}}$-hybrid model, we know that in the ideal model, this is the case when the honest parties receive $\perp$ as their output. If we assume that in the case when the the adversary decides to let the honest parties obtain their outputs, no honest party ever receives $\perp$, this could be used to identify the scenario when the adversary has decided to abort the execution of $\pi_{F'}$. Thus, we could, in principle, replace this instruction with: If party P_i receives $\perp$ as its output, it aborts. Furthermore, since we are considering the case of *unanimous abort*, if the adversary has decided to abort the execution of $\pi_{F'}$, all honest parties abort the protocol.

If all of these checks pass, then $f_1^{a,b}$ outputs

$$\sigma_{a,b} = (\mathcal{V}.\mathsf{Sign}((a,b);\mathsf{sk}_a), \mathcal{V}.\mathsf{Sign}((a,b);\mathsf{sk}_b))$$

and otherwise it outputs $\perp$.

– Let $\phi_1^{a,b}$ be the function that takes as input a witness w, which is either of the form $(0, \overrightarrow{\sigma})$ or of the form $\left(1, \overrightarrow{z}, \overrightarrow{\omega^z}, \overrightarrow{\pi}, \overrightarrow{\omega^\pi}, \overrightarrow{\mathsf{ind}}\right)$.

∗ If w is of the first form, then it tests if $r = 1$ and

$$\mathcal{V}.\mathsf{Verify}\left(\sigma_{a,b,1}, (a,b); \mathsf{vk}_a\right) = 1$$

and

$$\mathcal{V}.\mathsf{Verify}\left(\sigma_{a,b,2}, (a,b); \mathsf{vk}_b\right) = 1$$

for all $a, b \in [n]$ with $a < b$, outputting 1 if so and 0 if not.

∗ If w is of the second form, then it checks that:

· $|\overrightarrow{\pi}| = \left|\overrightarrow{\omega^\pi}\right| = \left|\overrightarrow{\mathsf{ind}}\right| = r$

· $\overrightarrow{\mathsf{ind}}$ consists of distinct indices in $[n]$.

· $\mathsf{Open}\left(c^z_{\mathsf{ind}_j}, \omega^z_j, z_j\right) = 1$ for every $j \in [r]$.

· $\mathsf{Open}\left(c^\pi_{\mathsf{ind}_j}, \omega^\pi_j, \pi_j\right) = 1$ for every $j \in [r]$.

If all of these checks pass, then $\phi_1^{a,b}$ outputs 1 and otherwise it outputs 0.

– Let $\phi_2^{a,b}$ be identical to $\phi_1^{a,b}$.

– Let $f_2^{a,b}$ be the function that takes as input (γ, γ') where γ, γ' are as above, and outputs $(\omega^z, z, \omega^\pi, \pi, \omega^{z\prime}, z', \omega^{\pi\prime}, \pi')$.

• Set $r = 0$[4]. Each party P_a for each $a \in [n]$ will now run the load phase to set up each instance of $\mathcal{F}_{\mathsf{SyX}}$ that it is involved in. For each pair of parties P_a, P_b

[4]This does not entail actually setting $r = 0$, but rather viewing the current round as round zero and henceforth referencing rounds with respect to it, that is, viewing r as the round number relative to the round number when this statement was executed.

with $a \neq b$ for $a, b \in [n]$, let $a' = \mathsf{min}(a, b)$ and $b' = \mathsf{max}(a, b)$. For each such pair of parties P_a, P_b, party P_a runs the load phase of $\mathcal{F}_{\mathsf{SyX}}^{a',b'}$, providing inputs (x_a, f), where

$$x_a = \left(\overrightarrow{\mathsf{vk}}, \mathsf{sk}_a, \beta_a, \overrightarrow{c^z}, \omega_a^z, z_a, \overrightarrow{c^\pi}, \omega_a^\pi, \pi_a\right)$$

and

$$f = \left(f_1^{a',b'}, f_2^{a',b'}, \phi_1^{a',b'}, \phi_2^{a',b'}\right)$$

- If $r > n$, abort. Otherwise, while $r \leq n$,
  - If a party P_a for $a \in [n]$ receives $\sigma_{a',b'}$ from each $\mathcal{F}_{\mathsf{SyX}}^{a',b'}$ it is involved in, indicating that the load phase of all such $\mathcal{F}_{\mathsf{SyX}}$ functionalities were completed successfully, and $r = 0$, it invokes the ideal functionality $\mathcal{F}_{\mathsf{bc}}$ and broadcasts

    $$\overrightarrow{\sigma_a} = \{\sigma_{a',b'}\}_{a'=a \ \vee \ b'=a}$$

    to all the parties. Otherwise, it invokes the ideal functionality $\mathcal{F}_{\mathsf{bc}}$ when $r = 1$ and broadcasts abort to all the parties and aborts.
  - If a party P_a for $a \in [n]$ receives $\overrightarrow{\sigma}$ such that

    $$\mathcal{V}.\mathsf{Verify}\left(\sigma_{a,b,1}, (a, b); \mathsf{vk}_a\right) = 1$$

    and

    $$\mathcal{V}.\mathsf{Verify}\left(\sigma_{a,b,2}, (a, b); \mathsf{vk}_b\right) = 1$$

    for all $a, b \in [n]$ with $a < b$, and $r = 1$, then it uses the witness $w = (0, \overrightarrow{\sigma})$ to invoke the trigger phase of each instance of $\mathcal{F}_{\mathsf{SyX}}$ that it is involved in. Once all such instances of $\mathcal{F}_{\mathsf{SyX}}$ involving party P_a have been triggered, use the shares $z_1, \ldots, z_n$ to reconstruct z, parses z as $z_1 \| \ldots \| z_n$ where $|z_i| = |y_i|$ for all $i \in [n]$[5] and computes $y_i = z_i \oplus \alpha_i$ to obtain the output of the computation.

[5] We may assume without loss of generality that the lengths of the outputs of each party are known beforehand.

– If party P_a for $a \in [n]$ has not received the output of the computation and an instance of $\mathcal{F}_{\mathsf{SyX}}$ involving party P_a is first triggered in round $1 \leq r < n$, it triggers each instance of $\mathcal{F}_{\mathsf{SyX}}$ that it is involved in during round $r+1$ using the output out it receives from the instance of $\mathcal{F}_{\mathsf{SyX}}$ as follows:

  * If $\mathsf{out}_1 = (0, \overrightarrow{\sigma})$, then $r = 1$. Let $\mathcal{F}_{\mathsf{SyX}}^{a',b'}$ be the instance of $\mathcal{F}_{\mathsf{SyX}}$ that was triggered, where $a' = a \vee b' = a$. Parse $\mathsf{out}_2 = (\omega^z, z, \omega^\pi, \pi, \omega^{z\prime}, z', \omega^{\pi\prime}, \pi')$. It prepares the witness

$$w = \left(1, (z, z'), \left(\omega^z, \omega^{z\prime}\right), (\pi, \pi'), \left(\omega^\pi, \omega^{\pi\prime}\right), (a', b')\right)$$

  * If $\mathsf{out}_1 = \left(1, \overrightarrow{z}, \overrightarrow{\omega^z}, \overrightarrow{\pi}, \overrightarrow{\omega^\pi}, \overrightarrow{\mathsf{ind}}\right)$, it prepares the witness

$$w = \left(1, \overrightarrow{z}', \overrightarrow{\omega^z}', \overrightarrow{\pi}', \overrightarrow{\omega^\pi}', \overrightarrow{\mathsf{ind}}'\right)$$

    where

    · $|\overrightarrow{z}'| = r+1,\ \overrightarrow{z}'|_{[r]} = \overrightarrow{z}|_{[r]},\ z'_{r+1} = z_a$

    · $\left|\overrightarrow{\omega^z}'\right| = r+1,\ \overrightarrow{\omega^z}'\Big|_{[r]} = \overrightarrow{\omega^z}\Big|_{[r]},\ {\omega^z_{r+1}}' = \omega^z_a$

    · $|\overrightarrow{\pi}'| = r+1,\ \overrightarrow{\pi}'|_{[r]} = \overrightarrow{\pi}|_{[r]},\ \pi'_{r+1} = \pi_a$

    · $\left|\overrightarrow{\omega^\pi}'\right| = r+1,\ \overrightarrow{\omega^\pi}'\Big|_{[r]} = \overrightarrow{\omega^\pi}\Big|_{[r]},\ {\omega^\pi_{r+1}}' = \omega^\pi_a$

    · $\left|\overrightarrow{\mathsf{ind}}'\right| = r+1,\ \overrightarrow{\mathsf{ind}}'\Big|_{[r]} = \overrightarrow{\mathsf{ind}}\Big|_{[r]},\ \mathsf{ind}'_{r+1} = a$

  Once all instances of $\mathcal{F}_{\mathsf{SyX}}$ involving party P_a have been triggered, it uses the shares $z_1, \ldots, z_n$ to reconstruct z, parses z as $z_1 \| \ldots \| z_n$ where $|z_i| = |y_i|$ for all $i \in [n]$ and computes $y_i = z_i \oplus \alpha_i$ to obtain the output of the computation.

– If party P_a for $a \in [n]$ has not received the output of the computation and an instance of $\mathcal{F}_{\mathsf{SyX}}$ involving party P_a is triggered and $r = n$, it receives all shares of z. It uses the shares $z_1, \ldots, z_n$ to reconstruct z, parses z as $z_1 \| \ldots \| z_n$ where $|z_i| = |y_i|$ for all $i \in [n]$ and computes $y_i = z_i \oplus \alpha_i$ to obtain the output of the computation.

Remark. It is possible to replace the $\mathcal{O}(n^2)$ signatures with n other commitments to n other independent random proof values (akin to π) that can be used to prove that all the instances of $\mathcal{F}_{\mathsf{SyX}}$ completed their load phases successfully.

5.4.3 Correctness

We sketch the proof of correctness of the above protocol. The correctness of the computation of the functionality F' follows by definition from the correctness of the ideal functionality $\mathcal{F}_{\mathsf{MPC}}$. Furthermore, we have that at the end of the invocation of the ideal functionality $\mathcal{F}_{\mathsf{MPC}}$, either all honest parties *unanimously abort* or all honest parties *unanimously continue*. Thus, assuming that $\mathcal{F}_{\mathsf{MPC}}$ did not abort, every party receives the output of F'. For every $i \in [n]$, let $\overrightarrow{\mathsf{vk}_i}$ denote the set of verification keys that were obtained by party P_i. Note that, by the correctness of the ideal functionality $\mathcal{F}_{\mathsf{bc}}$,

$$\overrightarrow{\mathsf{vk}} = \overrightarrow{\mathsf{vk}_i}$$

for all $i \in [n]$. If $\overrightarrow{\mathsf{vk}}$ does not contain vk_j for every $i \in [n]$, which would happen in the case that some corrupt parties do not broadcast their verification keys, all honest parties *unanimously abort*. Otherwise, all honest parties *unanimously continue*. Assuming the honest parties have not aborted, we note that if the corrupt parties do not provide valid inputs to the load phase of even one of the instances of $\mathcal{F}_{\mathsf{SyX}}$ that they are involved in along with an honest party, say P_i for some $i \in [n]$, by the correctness of the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$ and the binding property for the honestly generated commitments, that particular instance of $\mathcal{F}_{\mathsf{SyX}}$ will not complete its load phase successfully. In this case P_i will force all honest parties to *unanimously abort*, since no party (not even the corrupt ones) can obtain their output. We thus consider the case where all instances of $\mathcal{F}_{\mathsf{SyX}}$ have completed their load phases successfully. At this point, if all parties broadcast all the signatures they obtained from the instances of $\mathcal{F}_{\mathsf{SyX}}$, all parties can trigger the instances of $\mathcal{F}_{\mathsf{SyX}}$ that they are involved in to receive all the shares of z, reconstruct z and finally obtain their output correctly. The issue arises when some corrupt parties do not broadcast the signatures they obtained

from the instances of $\mathcal{F}_{\mathsf{SyX}}$. If a corrupt party triggers any instance of $\mathcal{F}_{\mathsf{SyX}}$ involving an honest party, say P_i for some $i \in [n]$, with a witness of the form $(0, \overrightarrow{\sigma})$ in round 1, then the honest party obtains a tuple of values $(z, \omega^z, \pi, \omega^\pi)$ from the corrupt. In addition its own such tuple of values, it obtains a valid witness to trigger all the instances of $\mathcal{F}_{\mathsf{SyX}}$ that it is involved in in round 2. Since $n \geq 2$, P_i succeeds in doing this and obtaining the shares of z, z and hence finally its output correctly. Consider any honest party P_j for $j \neq i$. Since $n > 2$, P_j, as did P_i, proceeds to trigger all the instances of $\mathcal{F}_{\mathsf{SyX}}$ that it is involved in in round 3. If no corrupt party triggers any instance of $\mathcal{F}_{\mathsf{SyX}}$ involving an honest party with a witness of the form $(0, \overrightarrow{\sigma})$ in round 1, if the adversary is to obtain the output, it must instruct a corrupt party to trigger an instance of $\mathcal{F}_{\mathsf{SyX}}$ that it is involved in along with an honest party, but now using a witness of the form $\left(1, \overrightarrow{z}, \overrightarrow{\omega^z}, \overrightarrow{\pi}, \overrightarrow{\omega^\pi}, \overrightarrow{\mathsf{ind}}\right)$. Let r be the first round when a corrupt party triggers an instance of $\mathcal{F}_{\mathsf{SyX}}$ that it is involved in along with an honest party, say P_i for some $i \in [n]$, using a witness of the form $\left(1, \overrightarrow{z}, \overrightarrow{\omega^z}, \overrightarrow{\pi}, \overrightarrow{\omega^\pi}, \overrightarrow{\mathsf{ind}}\right)$. Then, it must be the case that $i \notin \overrightarrow{\mathsf{ind}}$ and that P_i now obtains the tuple of values $(z, \omega^z, \pi, \omega^\pi)$ corresponding to r parties other than itself. Combining this information with its own tuple of values $(z, \omega^z, \pi, \omega^\pi)$, it obtains a valid witness to trigger all the instances of $\mathcal{F}_{\mathsf{SyX}}$ that it is involved in in round $r+1$. If $r < n$, P_i succeeds in doing this and obtaining the shares of z, z and hence finally its output correctly. Consider any honest party P_j for $j \neq i$. If $r + 1 = n$, then P_j receives all the shares of z and consequently its output correctly. If $r + 1 < n$, then P_j, as did P_i, proceeds to trigger all the instances of $\mathcal{F}_{\mathsf{SyX}}$ that it is involved in in round $r + 2$. Finally, we note that $r < n$ since r is the first round when a corrupt party triggers an instance of $\mathcal{F}_{\mathsf{SyX}}$ that it is involved in along with an honest party, which means that the witness it used to trigger the instance of $\mathcal{F}_{\mathsf{SyX}}$ can have the tuple of values $(z, \omega^z, \pi, \omega^\pi)$ corresponding to at most $n - 1$ parties as at least one of the parties is honest. If this does not happen, then no party (not even the corrupt ones) obtains their output. This completes the proof of correctness.

5.4.4 Security

We now prove the following lemma.

Lemma 30. *If* $\left(\mathsf{Com}, \mathsf{Open}, \widetilde{\mathsf{Com}}, \widetilde{\mathsf{Open}}\right)$ *is an honest-binding commitment scheme and* $\mathcal{V}$ *is a signature scheme, then the protocol* Π_{FMPC} *securely computes* $\mathcal{F}_{\mathsf{MPC}}$ *with fairness in the* $(\mathcal{F}_{\mathsf{bc}}, \mathcal{F}_{\mathsf{MPC}}, \mathcal{F}_{\mathsf{SyX}})$*-hybrid model.*

Proof. Let $\mathcal{A}$ be an adversary attaching the execution of the protocol described in Section 5.4.2 in the $(\mathcal{F}_{\mathsf{bc}}, \mathcal{F}_{\mathsf{MPC}}, \mathcal{F}_{\mathsf{SyX}})$-hybrid model. We construct an ideal-model adversary $\mathcal{S}$ in the ideal model of type fair. Let F be the n-input n-output functionality to be computed. Let $\mathcal{I}$ be the set of corrupted parties. If $\mathcal{I}$ is empty, then there is nothing to simulate. $\mathcal{S}$ begins by simulating the first step of the protocol, namely, the invocation of the ideal functionality $\mathcal{F}_{\mathsf{MPC}}$. Here, $\mathcal{S}$ behaves as the ideal functionality $\mathcal{F}_{\mathsf{MPC}}$. Recall that the type of $\mathcal{F}_{\mathsf{MPC}}$ is abort. $\mathcal{S}$ obtains the inputs $\{(x_i, f_i)\}_{i\in\mathcal{I}}$ of the corrupted parties from $\mathcal{A}$. If $(x_i, f_i) = \mathsf{abort}$ for any $i \in \mathcal{I}$, $\mathcal{S}$ forwards $\{(x_i, f_i)\}_{i\in\mathcal{I}}$ to the trusted party computing $\mathcal{F}_{\mathsf{MPC}}$ with fairness, receives $\perp$ as the output of all parties, which it forwards $\mathcal{A}$. Suppose $(x_i, f_i) \neq \mathsf{abort}$ for all $i \in \mathcal{I}$. If there exists a $j \in \mathcal{I}$ such that $f_j \neq F'$ as defined in protocol Π_{FMPC}, $\mathcal{S}$ forwards $\{(x_i, f_i)\}_{i\in\mathcal{I}}$ to the trusted party computing $\mathcal{F}_{\mathsf{MPC}}$ with fairness, which aborts, and then aborts itself. If there exists a $j \in \mathcal{I}$ such that (x_j, f_j) is not of the specified format, $\mathcal{S}$ replaces (x_j, f_j) with a default value. Going forward, we assume that for all $i \in \mathcal{I}$, (x_i, f_i) is well-formed and that $f_i = F'$ as defined in Π_{FMPC}.

$\mathcal{S}$ now needs to simulate the outputs received by the corrupted parties from the ideal functionality $\mathcal{F}_{\mathsf{MPC}}$. For each $i \in [n]$, $\mathcal{S}$ samples a random string $\alpha_i \xleftarrow{\$} \{0,1\}^*$ of length equal to the length of the ith output of F. Let

$$\alpha = \alpha_1 \| \ldots \| \alpha_n$$

For each $i \in \mathcal{I}$, $\mathcal{S}$ samples a random string $z_i \xleftarrow{\$} \{0,1\}^*$ of length equal to the sum of the lengths of all the outputs of F. It then computes commitments along with their openings $(c_i^z, \omega_i^z) \xleftarrow{\$} \mathsf{Com}(z_i)$ to each of the shares z_i for each $i \in \mathcal{I}$. For each

$i \in [n] \setminus \mathcal{I}$, it samples a equivocable commitment $(c_i^z, \mathsf{state}_i) \xleftarrow{\$} \widetilde{\mathsf{Com}}(1^\lambda)$. Let

$$\overrightarrow{c^z} = (c_1^z, \ldots, c_n^z)$$

For each $i \in [n]$, $\mathcal{S}$ samples random proof values $\pi_1, \ldots, \pi_n \xleftarrow{\$} \{0,1\}^\lambda$ and compute commitments along with their openings $(c_i^\pi, \omega_i^\pi) \xleftarrow{\$} \mathsf{Com}(\pi_i)$ to each of the proof values π_i. Let

$$\overrightarrow{c^\pi} = (c_1^\pi, \ldots, c_n^\pi)$$

and

$$\overrightarrow{\omega^\pi} = (\omega_1^\pi, \ldots, \omega_n^\pi)$$

Thus, the simulator constructs the output $\left(\alpha_i, \overrightarrow{c^z}, \omega_i^z, z_i, \overrightarrow{c^\pi}, \omega_i^\pi, \pi_i\right)$ for each $i \in \mathcal{I}$ and forwards it to $\mathcal{A}$. If $\mathcal{A}$ then sends abort, $\mathcal{S}$ forwards $\{(x_i, f_i)\}_{i \in \mathcal{I}}$ to the trusted party computing $\mathcal{F}_{\mathsf{MPC}}$ with fairness, with (x_j, f_j) replaced with abort for some $j \in \mathcal{I}$, receives $\perp$ as the output of all parties, which it forwards $\mathcal{A}$. Otherwise, $\mathcal{A}$ responds with $\mathsf{continue}$. At this point, $\mathcal{S}$ has completed simulating the invocation of the ideal functionality $\mathcal{F}_{\mathsf{MPC}}$.

For each $i \in [n] \setminus \mathcal{I}$, $\mathcal{S}$ picks a random $\beta_i \in \{0,1\}^*$ and uses this randomness to pick a signing and verification key pair $(\mathsf{sk}_i, \mathsf{vk}_i) = \mathcal{V}.\mathsf{Gen}(1^\lambda; \beta_i)$. Now, $\mathcal{S}$ must simulate the invocations of the ideal functionality $\mathcal{F}_{\mathsf{bc}}$ by the corrupt parties. Here, $\mathcal{S}$ behaves as the ideal functionality $\mathcal{F}_{\mathsf{bc}}$. Recall that the type of $\mathcal{F}_{\mathsf{bc}}$ is $\mathsf{g.d.}$. For all $i \in [n] \setminus \mathcal{I}$, $\mathcal{S}$ "broadcasts" vk_i to all the corrupt parties. For any $i \in \mathcal{I}$, if $\mathcal{A}$ instructs P_i to invoke $\mathcal{F}_{\mathsf{bc}}$ with input vk_i, $\mathcal{S}$ "broadcasts" vk_i to all the corrupt parties and stores vk_i. At the end of this round, if $\mathcal{A}$ did not instruct some corrupt party to invoke $\mathcal{F}_{\mathsf{bc}}$, $\mathcal{S}$ forwards $\{(x_i, f_i)\}_{i \in \mathcal{I}}$ to the trusted party computing $\mathcal{F}_{\mathsf{MPC}}$ with fairness, with (x_j, f_j) replaced with abort for some $j \in \mathcal{I}$, receives $\perp$ as the output of all parties, and aborts itself. Otherwise, $\mathcal{S}$ successfully constructs

$$\overrightarrow{\mathsf{vk}} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$$

At this point, $\mathcal{S}$ has completed simulating the invocations of the ideal functionality

$\mathcal{F}_{\mathsf{bc}}$ used to broadcast the verification keys of all the parties.

$\mathcal{S}$ maintains a virtual round counter and initializes it to zero. Now, $\mathcal{S}$ has to simulate the invocations of the load phases of the instances of the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$ that involve corrupt parties. Here, $\mathcal{S}$ behaves as the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$. Recall that the type of $\mathcal{F}_{\mathsf{SyX}}$ is $\mathsf{g.d.}$. For any $a, b \in [n]$ with $a < b$ and $a \in \mathcal{I}$ and $b \in [n] \setminus \mathcal{I}$, if $\mathcal{A}$ instructs P_a to invoke the load phase of $\mathcal{F}_{\mathsf{SyX}}^{a,b}$ with inputs

$$\gamma = \left(\overrightarrow{\mathsf{vk}'}, \mathsf{sk}_a, \beta_a, \overrightarrow{c^z}', \omega^z, z, \overrightarrow{c^\pi}', \omega^\pi, \pi\right)$$

$\mathcal{S}$ computes $f_1^{a,b}(\gamma, \gamma')$ as defined in Π_{FMPC}, where

$$\gamma' = \left(\overrightarrow{\mathsf{vk}}, \mathsf{sk}_b, \beta_b, \overrightarrow{c^z}, \omega_b^z, z_b, \overrightarrow{c^\pi}, \omega_b^\pi, \pi_b\right)$$

Note that since $b \in [n] \setminus \mathcal{I}$, $\mathcal{S}$ does in fact have $\mathsf{sk}_b, \beta_b, \omega_b^\pi, \pi_b$. The only values it does not have are ω_b^z, z_b. In the execution of $f_1^{a,b}$, ω_b^z, z_b are needed to check that

$$\mathsf{Open}(c_b^z, \omega_b^z, z_b) = 1$$

Note that since P_b is an honest party, it would always supply inputs such that this check passes. Furthermore, the outcome of this check does not depend on any input that the adversary sends. Thus, in simulating the computation of $f_1^{a,b}$, $\mathcal{S}$ performs all the checks that $f_1^{a,b}$, except this one. If all the checks pass, $\mathcal{S}$ computes

$$\sigma_{a,b} = (\mathcal{V}.\mathsf{Sign}((a,b); \mathsf{sk}_a), \mathcal{V}.\mathsf{Sign}((a,b); \mathsf{sk}_b))$$

and forwards $\sigma_{a,b}$ to the adversary. $\mathcal{S}$ also stores sk_a, β_a. If any of the checks do not pass, $\mathcal{S}$ simply aborts simulating the load phase of this particular instance $\mathcal{F}_{\mathsf{SyX}}^{a,b}$. $\mathcal{S}$ behaves symmetrically if for any $a, b \in [n]$ with $a < b$ and $b \in \mathcal{I}$ and $a \in [n] \setminus \mathcal{I}$, if $\mathcal{A}$ instructs P_b to invoke the load phase of $\mathcal{F}_{\mathsf{SyX}}^{a,b}$. The final case to consider is if for any $a, b \in [n]$ with $a < b$ and $a, b \in \mathcal{I}$, if $\mathcal{A}$ instructs P_a, P_b to invoke the load phase of

$\mathcal{F}_{\mathsf{SyX}}^{a,b}$ with inputs

$$\gamma = \left(\overrightarrow{\mathsf{vk}}', \mathsf{sk}_a, \beta_a, \overrightarrow{c^z}', \omega^z, z, \overrightarrow{c^\pi}', \omega^\pi, \pi\right)$$

and

$$\gamma' = \left(\overrightarrow{\mathsf{vk}}'', \mathsf{sk}_b, \beta_b, \overrightarrow{c^z}'', \omega^{z\prime}, z', \overrightarrow{c^\pi}'', \omega^{\pi\prime}, \pi'\right)$$

$\mathcal{S}$ computes $f_1^{a,b}(\gamma, \gamma')$ as defined in Π_{FMPC}. If all the checks pass, $\mathcal{S}$ computes

$$\sigma_{a,b} = (\mathcal{V}.\mathsf{Sign}((a,b); \mathsf{sk}_a), \mathcal{V}.\mathsf{Sign}((a,b); \mathsf{sk}_b))$$

and forwards $\sigma_{a,b}$ to the adversary. $\mathcal{S}$ also stores $\mathsf{sk}_a, \beta_a, \mathsf{sk}_b, \beta_b$. If any of the checks do not pass, $\mathcal{S}$ simply aborts simulating the load phase of this particular instance $\mathcal{F}_{\mathsf{SyX}}^{a,b}$. At the end of this round, let $\mathsf{LoadFailed}$ denote the set of all i such that P_i is an honest party and $\mathcal{A}$ did not instruct some corrupt party to invoke the load phase of an instance of $\mathcal{F}_{\mathsf{SyX}}$ that it was involved in with P_i. If $\mathsf{LoadFailed}$ is not empty, for each $i \in \mathsf{LoadFailed}$, $\mathcal{S}$ must simulate the invocations of the ideal functionality $\mathcal{F}_{\mathsf{bc}}$ by party P_i to broadcast abort. For each $i \in \mathsf{LoadFailed}$, $\mathcal{S}$ "broadcasts" abort to all the corrupt parties. $\mathcal{S}$ then forwards $\{(x_i, f_i)\}_{i \in \mathcal{I}}$ to the trusted party computing $\mathcal{F}_{\mathsf{MPC}}$ with fairness, with (x_j, f_j) replaced with abort for some $j \in \mathcal{I}$, receives $\perp$ as the output of all parties, and aborts itself. Otherwise, $\mathcal{S}$ successfully constructs

$$\overrightarrow{\mathsf{sk}} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$$

and

$$\overrightarrow{\beta} = (\beta_1, \ldots, \beta_n)$$

$\mathcal{S}$ computes

$$\sigma_{a,b} = (\mathcal{V}.\mathsf{Sign}((a,b); \mathsf{sk}_a), \mathcal{V}.\mathsf{Sign}((a,b); \mathsf{sk}_b))$$

for every $a < b \in [n]$ and defines

$$\overrightarrow{\sigma} = \{\sigma_{a,b}\}_{a<b, a,b \in [n]}$$

Now, $\mathcal{S}$ must simulate the invocations of the ideal functionality $\mathcal{F}_{\mathsf{bc}}$ by the corrupt parties. For all $a \in [n] \setminus \mathcal{I}$, $\mathcal{S}$ "broadcasts"

$$\overrightarrow{\sigma_i} = \{\sigma_{a',b'}\}_{a'=i \ \vee \ b'=i}$$

to all the corrupt parties. For any $i \in \mathcal{I}$, if $\mathcal{A}$ instructs P_i to invoke $\mathcal{F}_{\mathsf{bc}}$ with input $\overrightarrow{\sigma_i}$, $\mathcal{S}$ "broadcasts" $\overrightarrow{\sigma_i}$ to all the corrupt parties.

Once round 0 is completed, $\mathcal{S}$ has completed simulating the invocations of the load phase of all the instances of the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$ and the ideal functionality $\mathcal{F}_{\mathsf{bc}}$. What remains is to determine whether the adversary wishes to obtain its output and to simulate the invocations of the trigger phases of the instances of the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$ that the adversary instructs corrupt parties to trigger. We consider two cases. First, we make the following definition: a witness w is *valid* if

$$w = (0, \overrightarrow{\sigma})$$

in round 1 with

$$\mathcal{V}.\mathsf{Verify}\left(\sigma_{a,b,1}, (a,b); \mathsf{vk}_a\right) = 1$$

and

$$\mathcal{V}.\mathsf{Verify}\left(\sigma_{a,b,2}, (a,b); \mathsf{vk}_b\right) = 1$$

for all $a, b \in [n]$ with $a < b$, or

$$w = \left(1, \overrightarrow{z}, \overrightarrow{\omega^z}, \overrightarrow{\pi}, \overrightarrow{\omega^\pi}, \overrightarrow{\mathsf{ind}}\right)$$

in round $1 \leq r \leq n$ with

- $|\overrightarrow{\pi}| = \left|\overrightarrow{\omega^\pi}\right| = \left|\overrightarrow{\mathsf{ind}}\right| = r$
- $\overrightarrow{\mathsf{ind}}$ consists of distinct indices in $[n]$.
- $\mathsf{Open}\left(c^z_{\mathsf{ind}_j}, \omega^z_j, z_j\right) = 1$ for every $j \in [r]$.
- $\mathsf{Open}\left(c^\pi_{\mathsf{ind}_j}, \omega^\pi_j, \pi_j\right) = 1$ for every $j \in [r]$.

Case A. The adversary instructed all corrupt parties to broadcast all of their signatures. $\mathcal{S}$ forwards $\{(x_i, f_i)\}_{i\in\mathcal{I}}$ to the trusted party computing $\mathcal{F}_{\mathsf{MPC}}$ with fairness. It receives the corrupt parties outputs, namely, $\{y_i\}_{i\in\mathcal{I}}$. $\mathcal{S}$ chooses the outputs of the honest party completely at random, that is, it samples random strings $y_i \xleftarrow{\$} \{0,1\}^*$ of length equal to the length of the ith output of F, for $i \in [n] \setminus \mathcal{I}$. $\mathcal{S}$ then constructs

$$y = y_1 \| \ldots \| y_n$$

It then defines

$$z = y \oplus \alpha$$

Let j be an arbitrary index in $[n] \setminus \mathcal{I}$. $\mathcal{S}$ samples random strings $z_i \xleftarrow{\$} \{0,1\}^*$ of length equal to the sum of the lengths of all the outputs of F, for $i \in [n] \setminus (\mathcal{I} \cup \{j\})$. $\mathcal{S}$ then computes

$$z_j = z \oplus \bigoplus_{i\in[n]\setminus\{j\}} z_i$$

and constructs

$$\overrightarrow{z} = (z_1, \ldots, z_n)$$

$\mathcal{S}$ computes $\omega_i^z \xleftarrow{\$} \widetilde{\mathsf{Open}}(\mathsf{state}_i, z_i)$ for each $i \in [n] \setminus \mathcal{I}$ and constructs

$$\overrightarrow{\omega^z} = (\omega_1^z, \ldots, \omega_n^z)$$

Note that, at this point, $\mathcal{S}$ has every value ever used in the protocol. For every $i \in [n] \setminus \mathcal{I}$ and every $j \in \mathcal{I}$, letting $a = \mathsf{min}(i,j)$ and $b = \mathsf{max}(i,j)$, $\mathcal{S}$ sends $\left((\overrightarrow{\sigma}, 0), (\omega_a^z, z_a, \omega_a^\pi, \pi_a, \omega_b^z, z_b, \omega_b^\pi, \pi_b)\right)$ to P_j. Going forward, $\mathcal{S}$ simulates invocations of the trigger phases of the instances of the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$ that the adversary instructs corrupt parties to trigger as follows.

- Suppose the adversary instructs a corrupt party, say P_i for $i \in \mathcal{I}$, to trigger an instance of $\mathcal{F}_{\mathsf{SyX}}$ involving another corrupt party, say P_j for $j \in \mathcal{I}$, with a *valid* witness w, $\mathcal{S}$ sends $(w, (\omega_i^z, z_i, \omega_i^\pi, \pi_i, \omega_j^z, z_j, \omega_j^\pi, \pi_j))$[6] to parties P_i and P_j.

[6]Technically, this would have to be reordered as before. We ignore this technicality for ease of

- Suppose the adversary instructs a corrupt party, say P_i for $i \in \mathcal{I}$, to trigger an instance of $\mathcal{F}_{\mathsf{SyX}}$ involving an honest party, say P_j for $j \in [n] \setminus \mathcal{I}$, with a *valid* witness w, $\mathcal{S}$ sends $(w, (\omega_i^z, z_i, \omega_i^\pi, \pi_i, \omega_j^z, z_j, \omega_j^\pi, \pi_j))$ to P_i.

- Suppose the adversary instructs a corrupt party to trigger an instance of $\mathcal{F}_{\mathsf{SyX}}$ with an *invalid* witness. $\mathcal{S}$ simply sends no response.

- Suppose the an honest party, say P_i for $i \in [n] \setminus \mathcal{I}$, triggers an instance of $\mathcal{F}_{\mathsf{SyX}}$ involving a corrupt party, say P_j for $j \in \mathcal{I}$. $\mathcal{S}$ sends $(w, (\omega_i^z, z_i, \omega_i^\pi, \pi_i, \omega_j^z, z_j, \omega_j^\pi, \pi_j))$ to P_j.

Case B. The adversary did not instruct all corrupt parties to broadcast all of their signatures. We first discuss how $\mathcal{S}$ simulates certain invocations of the trigger phases of the instances of the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$ that the adversary instructs the corrupt parties to trigger.

- Suppose the adversary instructs a corrupt party, say P_i for $i \in \mathcal{I}$, to trigger an instance of $\mathcal{F}_{\mathsf{SyX}}$ involving another corrupt party, say P_j for $j \in \mathcal{I}$, with a *valid* witness. $\mathcal{S}$ sends $(w, (\omega_i^z, z_i, \omega_i^\pi, \pi_i, \omega_j^z, z_j, \omega_j^\pi, \pi_j))$ to parties P_i and P_j.

- Suppose the adversary instructs a corrupt party to trigger an instance of $\mathcal{F}_{\mathsf{SyX}}$ with an *invalid* witness. $\mathcal{S}$ simply sends no response.

Suppose the adversary does not instruct a corrupt party, say P_i for some $i \in \mathcal{I}$, to trigger an instance of $\mathcal{F}_{\mathsf{SyX}}$ involving an honest party, say P_j for some $j \in [n] \setminus \mathcal{I}$, with a *valid* witness and the round counter exceeds n, $\mathcal{S}$ forwards $\{(x_i, f_i)\}_{i \in \mathcal{I}}$ to the trusted party computing $\mathcal{F}_{\mathsf{MPC}}$ with fairness, with (x_j, f_j) replaced with abort for some $j \in \mathcal{I}$, receives $\perp$ as the output of all parties, and aborts itself. Otherwise, at the first instant the adversary instructs a corrupt party to trigger an instance of $\mathcal{F}_{\mathsf{SyX}}$ involving an honest party with a *valid* witness w, $\mathcal{S}$ forwards $\{(x_i, f_i)\}_{i \in \mathcal{I}}$ to the trusted party computing $\mathcal{F}_{\mathsf{MPC}}$ with fairness. It receives the corrupt parties outputs, namely, $\{y_i\}_{i \in \mathcal{I}}$. $\mathcal{S}$ chooses the outputs of the honest party completely at random,

presentation.

that is, it samples random strings $y_i \xleftarrow{\$} \{0,1\}^*$ of length equal to the length of the ith output of F, for $i \in [n] \setminus \mathcal{I}$. $\mathcal{S}$ then constructs

$$y = y_1 \| \ldots \| y_n$$

It then defines

$$z = y \oplus \alpha$$

Let j be an arbitrary index in $[n] \setminus \mathcal{I}$. $\mathcal{S}$ samples random strings $z_i \xleftarrow{\$} \{0,1\}^*$ of length equal to the sum of the lengths of all the outputs of F, for $i \in [n] \setminus (\mathcal{I} \cup \{j\})$. $\mathcal{S}$ then computes

$$z_j = z \oplus \bigoplus_{i \in [n] \setminus \{j\}} z_i$$

and constructs

$$\overrightarrow{z} = (z_1, \ldots, z_n)$$

$\mathcal{S}$ computes $\omega_i^z \xleftarrow{\$} \widetilde{\mathsf{Open}}(\mathsf{state}_i, z_i)$ for each $i \in [n] \setminus \mathcal{I}$ and constructs

$$\overrightarrow{\omega^z} = (\omega_1^z, \ldots, \omega_n^z)$$

Note that, at this point, $\mathcal{S}$ has every value ever used in the protocol. $\mathcal{S}$ sends the tuple of values $(w, (\omega_i^z, z_i, \omega_i^\pi, \pi_i, \omega_j^z, z_j, \omega_j^\pi, \pi_j))$ to P_i. Going forward, $\mathcal{S}$ simulates invocations of the trigger phases of the instances of the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$ that involve corrupt parties as follows.

- Suppose the adversary instructs a corrupt party, say P_i for $i \in \mathcal{I}$, to trigger an instance of $\mathcal{F}_{\mathsf{SyX}}$ involving another corrupt party, say P_j for $j \in \mathcal{I}$, with a *valid* witness w, $\mathcal{S}$ sends $(w, (\omega_i^z, z_i, \omega_i^\pi, \pi_i, \omega_j^z, z_j, \omega_j^\pi, \pi_j))$ to parties P_i and P_j.

- Suppose the adversary instructs a corrupt party, say P_i for $i \in \mathcal{I}$, to trigger an instance of $\mathcal{F}_{\mathsf{SyX}}$ involving an honest party, say P_j for $j \in [n] \setminus \mathcal{I}$, with a *valid* witness w, $\mathcal{S}$ sends $(w, (\omega_i^z, z_i, \omega_i^\pi, \pi_i, \omega_j^z, z_j, \omega_j^\pi, \pi_j))$ to P_i.

- Suppose the adversary instructs a corrupt party to trigger an instance of $\mathcal{F}_{\mathsf{SyX}}$

with an *invalid* witness. $\mathcal{S}$ simply sends no response.

- Suppose the an honest party, say P_i for $i \in [n] \backslash \mathcal{I}$, triggers an instance of $\mathcal{F}_{\mathsf{SyX}}$ involving a corrupt party, say P_j for $j \in \mathcal{I}$. $\mathcal{S}$ sends $(w, (\omega_i^z, z_i, \omega_i^\pi, \pi_i, \omega_j^z, z_j, \omega_j^\pi, \pi_j))$ to P_j.

Finally, $\mathcal{S}$ outputs whatever $\mathcal{A}$ outputs. It is easy to see that the view of $\mathcal{A}$ is indistinguishable in the execution of the protocol Π_{FMPC} and the simulation with $\mathcal{S}$, if $\left(\mathsf{Com}, \mathsf{Open}, \widetilde{\mathsf{Com}}, \widetilde{\mathsf{Open}}\right)$ is an honest-binding commitment scheme and $\mathcal{V}$ is a signature scheme. We therefore conclude that the protocol Π_{FMPC} securely computes $\mathcal{F}_{\mathsf{MPC}}$ with fairness in the $(\mathcal{F}_{\mathsf{bc}}, \mathcal{F}_{\mathsf{MPC}}, \mathcal{F}_{\mathsf{SyX}})$-hybrid model, as required. □

Remark. In the proof of Lemma 30, we ignore some annoying technicalities. For instance, the adversary may cause the honest parties to abort, will be unable to obtain its output but still pointlessly interact with some of the ideal functionalities. In the proof, however, the simulator would have aborted. We note that these details are not particularly enlightening and are of no consequence. One can deal with these sorts of attacks by asking the simulator to wait in these scenarios until the adversary says that it is done and then finally abort if it has to. Thus, we assume, for the purpose of the proof, that if the adversary forces the honest parties to abort in a situation where it will be unable to obtain its output, without loss of generality, it halts. Other examples of such technicalities are when the adversary sends "unexpected" messages, "incomplete" messages, etc. Note that such messages can be easily detected and ignored, and do not affect the protocol in any way.

5.4.5 Getting to the $\mathcal{F}_{\mathsf{SyX}}$-hybrid model

Combining Lemmas 1, 29, 16 and 30, we obtain the following theorem.

Theorem 16. *Consider n parties $P_1, \ldots, P_n$ in the* point-to-point *model. Then, assuming the existence of one-way functions, there exists a protocol π which securely computes $\mathcal{F}_{\mathsf{MPC}}$ with fairness in the presence of t-threshold adversaries for any $0 \leq t < n$ in the $(\mathcal{F}_{\mathsf{OT}}, \mathcal{F}_{\mathsf{SyX}})$-hybrid model.*

As discussed in Section 5.2, $\mathcal{F}_{\mathsf{2PC}}$, and hence $\mathcal{F}_{\mathsf{OT}}$, can be realized in the $\mathcal{F}_{\mathsf{SyX}}$-hybrid model. We thus have the following theorem.

Theorem 17. *Consider n parties $P_1, \ldots, P_n$ in the* point-to-point *model. Then, assuming the existence of one-way functions, there exists a protocol π which securely computes $\mathcal{F}_{\mathsf{MPC}}$ with fairness in the presence of t-threshold adversaries for any $0 \leq t < n$ in the $\mathcal{F}_{\mathsf{SyX}}$-hybrid model.*

It is important to note that via this transformation, we have not introduced a need for the parties to have access to multiple instances of the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$ as opposed to one. This is because, in the protocol Π_{FMPC}, the ideal functionality $\mathcal{F}_{\mathsf{OT}}$ will only be used to emulate the ideal functionality $\mathcal{F}_{\mathsf{MPC}}$. During this stage, we do not make any use of the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$. Once we are done with the signle invocation of $\mathcal{F}_{\mathsf{MPC}}$, we only invoke the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$. As a consequence, parties can reuse the same instance of $\mathcal{F}_{\mathsf{SyX}}$ to first emulate $\mathcal{F}_{\mathsf{OT}}$ and then as a complete $\mathcal{F}_{\mathsf{SyX}}$ functionality. We note that this however does increase the number of times the functionality is invoked.

5.5 Preprocessing $\mathcal{F}_{\mathsf{SyX}}$

In this section, we will describe how a pair of parties can "preprocess" an instance of the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$.We first describe what we mean by "preprocess". What we would like to enable is the following. We already know that the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$ allows the pair of parties to perform fair two-party computations. We would like to set up the $\mathcal{F}_{\mathsf{SyX}}$ functionality such that after a *single* invocation of the load phase, the two parties can perform an arbitrary (*apriori* unknown) polynomial number of fair two-party computations. Furthermore, if the parties are honest, they would not need to invoke the ideal functionality, that is, the "preprocessing" of the functionality is optimistic. Combining this with the protocol for fair multiparty computation in the $\mathcal{F}_{\mathsf{SyX}}$-hybrid model from Section 5.4, we are able to show how an arbitrary set of n parties in the point-to-point model that have pairwise access to the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$ that has been preprocessed, can perform an arbitrary (*apriori* unknown) polynomial number of fair multiparty computations. To begin with, we will assume that the n-parties are in the point-to-point model and develop a protocol in the $(\mathcal{F}_{\mathsf{bc}}, \mathcal{F}_{\mathsf{MPC}}, \mathcal{F}_{\mathsf{SyX}})$-hybrid model. We first provide some intuition for our construction.

5.5.1 Intuition

We first start with the 3-party case as a warm-up. Let P_1, P_2, and P_3 be the three parties, subsets (or all) of which would like to perform an unbounded (*apriori* unknown polynomial) number of secure function evaluations. For $i, j \in \{1, 2, 3\}$ with $i < j$, we have that parties P_i and P_j have access to the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$. In particular, let $\mathcal{F}_{\mathsf{SyX}}^{i,j}$ represent the instantiation of the $\mathcal{F}_{\mathsf{SyX}}$ functionality used by parties P_i, P_j. We wish to perform fair secure function evaluation of some 3-input 3-output functionality F.

Reduction to single output functionalities. Let $(y_1, y_2, y_3) \xleftarrow{\$} F(x_1, x_2, x_3)$ be the output of a function evaluation[7]. We define a new four input single output

[7]This discussion can be trivially extended to function evaluations with two inputs as opposed to three.

functionality F' such that

$$F'(x_1, x_2, x_3, z) = F^1(x_1, x_2, x_3) \| F^2(x_1, x_2, x_3) \| F^3(x_1, x_2, x_3) \oplus z = y_1 \| y_2 \| y_3 \oplus z$$

where $z = z_1 \| z_2 \| z_3$ and $|y_i| = |z_i|$ for all $i \in [3]$. The idea is that the party P_i will obtain $z' = F'(x_1, x_2, x_3, z)$ and z_i. Viewing $z' = z'_1 \| z'_2 \| z'_3$ where $|z'_i| = |z_i|$[8] for all $i \in [3]$, party P_i reconstructs its output as

$$y_i = z_i \oplus z'_i$$

Now, we may assume that the input of party P_i is (x_i, z_i) (or we can generate random z_is as part of the computation) which determines z. It thus suffices to consider fair secure function evaluation of single output functionalities.

Reduction to fair reconstruction. We will use ideas similar to [73, 98] where instead of focusing on fair secure evaluation of an arbitrary function, we only focus on fair reconstruction of an additive secret sharing scheme. The main idea is to let the parties run a secure computation protocol that computes the output of the secure function evaluation on the parties' inputs, and then additively secret shares the output. Given this step, fair secure computation then reduces to fair reconstruction of the underlying additive secret sharing scheme.

Instance and party independence. Looking ahead, as in Section 5.4, we will use the instances of the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$ to perform fair reconstruction. In order to be able to preprocess the instances of the functionality for arbitrary reconstructions, what is being reconstructed must be independent of the secure function evaluation and, in particular, the inputs of the parties. Furthermore, it must also be independent of the specific parties that are performing the reconstruction. However, until now, we have been assuming that the output of the secure function evaluation on the parties' inputs is what is being reconstructed, which does not satisfy our requirements and

[8] We may assume without loss of generality that the lengths of the outputs of each party are known beforehand.

hence would not allow preprocessing. In order to fix this, we assume that the output of the secure function evaluation on the parties' inputs is encrypted under a key and that key is what will be reconstructed fairly. Note that the key can be chosen independent of the secure function evaluation and the parties' inputs. We would also like it to be the case that even after reconstructing once, our preprocessing is valid. This would require that the preprocessing allows for the generation and fair reconstruction of multiple independent (to a computational adversary) keys, one for each secure function evaluation. Thus, what is actually done during the preprocessing phase is the following. Each pair of parties P_i and P_j then initializes $\mathcal{F}_{\mathsf{SyX}}^{i,j}$. The function f_1 samples two random values $v_{i,j}, v_{j,i} \xleftarrow{\$} \{0,1\}^\lambda$ and computes the output of f_1 as

$$V_{i,j} = V_{j,i} = v_{i,j} \oplus v_{j,i}$$

along with commitments to these values to ensure that only these values are used by parties in protocols. This completes the description of f_1.

The underlying additive secret sharing scheme. For the instance of secure function evaluation with identifier id, we sample a unique key, K_{id}, to encrypt the output y_{id} of the secure function evaluation. Let Enc denote the encryption algorithm of an encryption scheme. The parties would receive $\mathsf{ct}_{\mathsf{id}} = \mathsf{Enc}(y_{\mathsf{id}}; K_{\mathsf{id}})$ and then fairly reconstruct K_{id}. We use an independent additive secret sharing of the key K_{id} for each party. Let the shares be $k_{\mathsf{id},i,j}$ for $i, j \in [3]$. That is, it holds that

$$K_{\mathsf{id}} = \bigoplus_{j \in [3]} k_{\mathsf{id},i,j}$$

for each $i \in [3]$. We would like party P_i to reconstruct K_{id} by obtaining all shares $k_{\mathsf{id},i,j}$ for each $j \in [3]$. Initially, each party P_i is given $k_{\mathsf{id},i,i}$. Therefore, each party P_i only needs to obtain $k_{\mathsf{id},i,j}$ and $k_{\mathsf{id},i,j'}$ for $j, j' \neq i$. Looking ahead, we would use the instances of the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$ to allow parties to fairly learn all their shares of K_{id}. However, since we are preprocessing the instances, the information needed to compute these shares must be independent of the instance of secure function

evaluation. The value that the instance $\mathcal{F}_{\mathsf{SyX}}^{i,j}$ would release fairly to parties P_i and P_j is $V_{i,j}$. Thus, party P_i additionally receives

$$\mathsf{ct}_{\mathsf{id},i,j} = \mathsf{Enc}(k_{\mathsf{id},i,j}; h_{\mathsf{id},i,j})$$

where

$$h_{\mathsf{id},i,j} = H(V_{i,j} \| \mathsf{id})$$

where H is a hash function (random oracle). The intuition is that the instances of the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$ to allow parties to fairly learn the $V_{i,j}$s, and hence the $h_{\mathsf{id},i,j}$s and finally $k_{\mathsf{id},i,j}$s, thus fairly reconstructing K_{id}. It is important to note that using $V_{i,j}$s that are independent of the instance of secure function evaluation, we can fairly reconstruct, using per-instance (computationally independent) hash values $h_{\mathsf{id},i,j}$ generated using $V_{i,j}$s, per-instance (independent) encryption keys K_{id}.

An attempt at fair reconstruction via $\mathcal{F}_{\mathsf{SyX}}$. We assume that the secure function evaluation with identifier id provides the encryption $\mathsf{ct}_{\mathsf{id}}$ of the output y_{id} of the secure function evaluation. Additionally, party P_i receives $\mathsf{ct}_{\mathsf{id},i,j}$ for each $j \in [n]$ and $k_{\mathsf{id},i,i}$. From our earlier discussion, the instances of the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$ allow parties to fairly learn the $V_{i,j}$s. In order to allow reuse of the preprocessing, however, the instances of the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$ must only allow parties to fairly learn the $h_{\mathsf{id},i,j}$s. As a first attempt to ensure this, we require the secure function evaluation to also give party P_i a signature σ_i on id. That is, P_i receives

$$\left(\mathsf{ct}_{\mathsf{id}}, \{\mathsf{ct}_{\mathsf{id},i,j}\}_{j\in[3]}, k_{\mathsf{id},i,i}, \sigma_i\right)$$

We will have the parties fairly learn the $h_{\mathsf{id},i,j}$s using the instances of the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$. We achieve this by setting the predicate $\phi_k(w)$ (for $k \in \{i,j\}$) to output 1 if and only if w consists of both signatures (σ_i, σ_j). That is, each instance $\mathcal{F}_{\mathsf{SyX}}^{i,j}$ will accept the trigger $w_{i,j} = (\mathsf{id}, \sigma_i, \sigma_j)$. We define f_2 to simply output $h_{\mathsf{id},i,j}$ to both parties if $\phi_k(w) = 1$. Parties learn signatures of other parties by broadcasting

their signatures and waiting for other parties to do so. If party P_i receives signatures from every other party, it can trigger every instance of the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$ it is involved in, thus learning $h_{\mathsf{id},i,j}$ for each $j \in [3]$ and finally learning K_{id}. Malicious parties may however not broadcast their signatures. Concretely, we have the following attack: Suppose P_1 is honest while P_2 and P_3 are corrupt. P_2 and P_3 already know σ_2 and σ_3 and only need σ_1 to learn the output. P_1 broadcasts σ_1 while P_2 and P_3 do not broadcast σ_2 and σ_3. Finally, P_2 triggers the ideal functionality $\mathcal{F}_{\mathsf{SyX}}^{1,2}$ using (σ_1, σ_2) and learns the output. P_1, on the other hand, only learns σ_2 and hence does not learn the output.

Fair reconstruction via $\mathcal{F}_{\mathsf{SyX}}$. We fix the protocol sketch described above using a technique we developed for termination of the protocol described in Section 5.4. The protocol for reconstruction proceeds in $T = n$ rounds. In order to trigger an instance of $\mathcal{F}_{\mathsf{SyX}}$ in some round $\tau \in [T]$, you must provide a proof that other instances of $\mathcal{F}_{\mathsf{SyX}}$ involving at least τ different parties have been triggered. Consider the first round τ in which P_i is an honest party and $\mathcal{F}_{\mathsf{SyX}}^{i,j}$ is triggered for some j. If $\tau = 1$, then the single invocation already gives a proof that channels involving two parties, namely, i, j, have been triggered. Otherwise, by assumption, proofs of invocations of instances of $\mathcal{F}_{\mathsf{SyX}}$ involving τ different parties were needed to trigger $\mathcal{F}_{\mathsf{SyX}}^{i,j}$. But P_i is not one of these parties as τ is the first round in which $\mathcal{F}_{\mathsf{SyX}}^{i,j}$ was triggered for any j. Consequently, P_i, on this invocation, obtains a proof that instances of $\mathcal{F}_{\mathsf{SyX}}$ involving at least $\tau + 1$ parties have been triggered, and can thus trigger all channels in round $\tau + 1$. The only gap in the argument is the case $\tau = T$. One can trivially see that since $\mathcal{F}_{\mathsf{SyX}}^{i,j}$ has not been triggered for any j, it is impossible to obtain a proof that instances of $\mathcal{F}_{\mathsf{SyX}}$ involving at least T different parties have been triggered.

Optimistic preprocessing. In the case where parties are honest, we can simply have the secure function evaluation provide the output instead of parties having to trigger their instances of the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$. We are guaranteed, by virtue of the fair reconstruction techniques dicussed thus far, that in the case where parties

behave adversarially, the honest parties do have a way to obtain the output of the computation. In this way, in the optimistic setting, parties never have to trigger the instances of the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$. Combined with the fact that the actual preprocessing phase is extremely simple, we see that this paradigm makes fair secure function evaluation just as efficient as secure function evaluation with abort in the optimisitic case.

Simulation. We look ahead for the issues that come up while trying to prove security, that is, during the simulation. The simulator will release to the adversary, the encryption of the output and encryptions of the adversary's shares of the key used to encrypt the output. Since the simulator does not know the output *apriori*, and does not know whether the adversary is going to abort the computation, in which case, no one knows the output, it has to produce encryptions that it can later *equivocate*. In this context, we use, not a regular encryption scheme, but non-interactive non-committing encryption commitments. In this case, the simulator can produce encryptions to garbage but can later decrypt them to be *valid* shares of the key and the actual output. The rest of the computations can be trivially simulated. The only other detail to be looked into is that of the clock. We need to determine if the adversary has decided to abort the computation, that is, if the adversary is going to receive the output of the computation of not. This is done by noticing if and when the adversary decides to trigger the instances of $\mathcal{F}_{\mathsf{SyX}}$ that involve honest parties. We know that if the adversary ever triggers an instance of $\mathcal{F}_{\mathsf{SyX}}$ involving an honest party, then all parties will be in a position to receive the output. Thus, the simulator can simply run the adversary to determine whether it has decided to enable parties to obtain the output, in which the simulator would ask the trusted party to continue, or not, in which case the simulator would ask the trusted party to abort.

5.5.2 Protocol

We now present the protocol for preprocessing fair secure computation in the $(\mathcal{F}_{\mathsf{MPC}}, \mathcal{F}_{\mathsf{SyX}})$-hybrid model.

STAGE I: PREPROCESSING

Preliminaries. P_a and P_b are two parties for $a, b \in [N]$ with $a < b$, where N is some universal upper bound on the number of parties; $\mathcal{F}_{\mathsf{SyX}}^{a,b}$ represents the instantiation of the $\mathcal{F}_{\mathsf{SyX}}$ functionality used by parties P_a, P_b with time out round numbers $\mathsf{INPUT\_TIMEOUT} = \infty$ and $\mathsf{TRIGGER\_TIMEOUT} = \infty$; $(\mathsf{Com}, \mathsf{Open})$ is a commitment scheme; $H : \{0,1\}^* \to \{0,1\}^{L(\lambda)}$[9] is a random oracle; r denotes the current round number.

Protocol. The protocol $\Pi_{\mathsf{Preprocess}}$ proceeds as follows:

- Define the following functions.
  - Let $f_1^{a,b}$ be the function that takes no input, samples two strings $v_{a,b}, v_{b,a} \xleftarrow{\$} \{0,1\}^\lambda$ and computes and stores

$$V_{a,b} = V_{b,a} = v_{a,b} \oplus v_{b,a}$$

    It also computes

$$(c_{a,b}^v, \omega_{a,b}^v) \xleftarrow{\$} \mathsf{Com}(v_{a,b})$$

    and

$$(c_{b,a}^v, \omega_{b,a}^v) \xleftarrow{\$} \mathsf{Com}(v_{b,a})$$

    Finally, it outputs $(v_{a,b}, c_{a,b}^v, \omega_{a,b}^v, c_{b,a}^v)$ to party P_a and $(v_{b,a}, c_{a,b}^v, c_{b,a}^v, \omega_{b,a}^v)$ to party P_b.
  - Let $\phi_1^{a,b}$ be the function that takes as input a witness w and parses

$$w = \left(\mathsf{id}, t, \overrightarrow{c^\pi}, \overrightarrow{\omega^\pi}, \overrightarrow{\pi}, \overrightarrow{\mathsf{ind}}\right)$$

    It checks that:

[9] The following type-check must and can be performed: $L(\lambda)$ is the length of the key that will be used to encrypt shares of a key that will be used to encrypt the output of the secure function evaluations to be performed in the future.

* $\mathsf{id} \in \{0,1\}^\lambda$
* t is a valid round number and $t < r$
* $\left|\overrightarrow{\mathsf{ind}}\right| = \left|\overrightarrow{\omega^\pi}\right| = |\overrightarrow{\pi}| = r - t$
* $\overrightarrow{\mathsf{ind}}$ consists of distinct indices in $\left[|\overrightarrow{c^\pi}|\right]$.
* $\mathsf{Open}\left(c^\pi_{\mathsf{ind}_j}, \omega^\pi_j, \pi_j\right) = 1$ for every $j \in [r-t]$.

If all of these checks pass, then $\phi_1^{a,b}$ outputs 1 and otherwise it outputs 0.

– Let $\phi_2^{a,b}$ be identical to $\phi_1^{a,b}$.

– Let $f_2^{a,b}$ be the function that takes as input w where w is as above, and outputs

$$h_{\mathsf{id},a,b} = h_{\mathsf{id},b,a} = H\left(V_{a,b} \| \mathsf{id} \| t \| \overrightarrow{c^\pi}\right)$$

- Parties P_a, P_b run the load phase of $\mathcal{F}^{a,b}_{\mathsf{SyX}}$, providing the same input $(\perp, f)$, where

$$f = \left(f_1^{a,b}, f_2^{a,b}, \phi_1^{a,b}, \phi_2^{a,b}\right)$$

If parties P_a and P_b receive their outputs $v_{a,b}$ and $v_{b,a}$ respectively, we say that their preprocessing phase has been successfully completed.

STAGE II: FAIR SECURE FUNCTION EVALUATION

Preliminaries. $S \subseteq [N]$, where n is some universal upper bound on the number of parties, and $|S| = n$; F is the n-input n-output functionality to be computed; x_i is the input of party P_i for $i \in S$; $\mathcal{F}^{a,b}_{\mathsf{SyX}}$ represents the instantiation of the $\mathcal{F}_{\mathsf{SyX}}$ functionality used by parties P_a, P_b with time out round numbers $\mathsf{INPUT\_TIMEOUT} = \infty$ and $\mathsf{TRIGGER\_TIMEOUT} = \infty$ for $a < b$, where $a, b \in S$; $(\mathsf{Com}, \mathsf{Open})$ is an commitment scheme; $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is non-interactive non-committing encryption scheme; $H : \{0,1\}^* \to \{0,1\}^{L(\lambda)}$[10] is a random oracle; r denotes the current round number.

[10]The following type-check must and can be performed: $L(\lambda)$ is the length of the randomness needed to generate a key long enough to encrypt shares of a key long enough to encrypt the output of the secure function evaluations to be performed in the future.

Protocol. The protocol $\Pi_{\mathsf{FMPC\text{-}preprocess}}$ proceeds as follows:

- Party P_i for every $i \in S$ ensures that its preprocessing phases with every other party P_j for $j \in S \setminus \{i\}$ have been successfully completed. If not, party P_i aborts. Otherwise, it has obtained values $\left\{v_{i,j}, c_{i,j}^{v,i}, \omega_{i,j}^{v,i}, c_{j,i}^{v,i}\right\}_{j \in S \setminus \{i\}}$ as outputs from its preprocessing phases with every other party.

- Define F' to the be the following n-input n-output functionality: On input $\overrightarrow{X} = (X_1, \ldots, X_n)$:

  - Parse
  $$X_i = \left(x_i, t_i, \left\{v_{i,j}, c_{i,j}^{v,i}, \omega_{i,j}^{v,i}, c_{j,i}^{v,i}\right\}_{j \in S \setminus \{i\}}\right)$$
  - Check that $t = t_i = t_j$ for all $i, j \in S$. If not, abort.
  - Check that $c_{i,j}^{v,i} = c_{i,j}^{v,j}$ and that $\mathsf{Open}(c_{i,j}^{v,i}, \omega_{i,j}^{v,i}, v_{i,j}) = 1$ for all $i, j \in S$ with $i \neq j$. If not, abort.
  - Sample a random identifier $\mathsf{id} \in \{0,1\}^\lambda$ for this instance of fair secure function evaluation.
  - Let $(y_1, \ldots, y_n) = F(x_1, \ldots, x_n)$ and let
  $$y = y_1 \| \ldots \| y_n$$
  Sample random strings $\alpha_i \xleftarrow{\$} \{0,1\}^*$ such that $|\alpha_i| = |y_i|$ for each $i \in [n]$. Let
  $$\alpha = \alpha_1 \| \ldots \| \alpha_n$$
  Let $z = y \oplus \alpha$.
  - Sample a random encryption key-pair $(\mathsf{pk}, \mathsf{sk})$ by invoking $\mathsf{Gen}(1^\lambda)$. Compute an encryption of the output
  $$\mathsf{ct} = \mathsf{Enc}(z; \mathsf{pk})$$

– Sample n random additive n-out-of-n secret sharings $\{k_{1,j}\}_{j\in S}, \ldots, \{k_{n,j}\}_{j\in S}$ of sk such that

$$\mathsf{sk} = \bigoplus_{j\in[n]} k_{i,j}$$

for every $i \in S$.

– Sample random proof values $\pi_1, \ldots, \pi_n \xleftarrow{\$} \{0,1\}^\lambda$. Compute commitments along with their openings $(c_i^\pi, \omega_i^\pi) \xleftarrow{\$} \mathsf{Com}(\pi_i)$ to each of the proof values π_i for each $i \in [n]$. Let

$$\overrightarrow{c^\pi} = (c_1^\pi, \ldots, c_n^\pi)$$

– Compute

$$h_{i,j} = h_{j,i} = H\left(V_{i,j} \| \mathsf{id} \| t \| \overrightarrow{c^\pi}\right)$$

where

$$V_{i,j} = V_{j,i} = v_{i,j} \oplus v_{j,i}$$

for every $i, j \in [n]$ with $i \neq j$.

– Sample encryption key-pairs $(\mathsf{pk}_{i,j}, \mathsf{sk}_{i,j})$ by invoking $\mathsf{Gen}(1^\lambda; h_{i,j})$ for every $i, j \in [n]$ with $i \neq j$. Compute

$$\mathsf{ct}_{i,j} = \mathsf{Enc}(k_{i,j}; \mathsf{pk}_{i,j})$$

for every $i, j \in [n]$ with $i \neq j$.

– In the first stage of output delivery[11], party P_i receives output

$$\left(\alpha_i, \mathsf{id}, t, \mathsf{ct}, \{\mathsf{ct}_{i,j}\}_{j\in S\setminus\{i\}}, k_{i,i}, \omega_i^\pi, \pi_i\right)$$

[11]While this would require setting up additional notation, for ease of presentation, we suppress formally defining security of multi-stage primitives. In our case, the functionality F' is a two-stage functionality. While defining security with abort for F', which is all we will need for this work, we consider security with abort for each stage. That is, the adversary obtains its output for the first stage and then decides whether the honest parties may receive their output for the first stage. If not, neither the adversary nor the honest parties obtain their output for the second stage. If yes, the honest parties receive their output for the first stage and the adversary receives its output for the second stage, following which it decides if the honest parties may receive their output for the second stage.

for each $i \in S$.

- In the second stage of output delivery, party P_i receives output $\overrightarrow{c^{\pi}}$ for each $i \in S$.
- Finally party P_i receives the output z for each $i \in S$.

- For each $i \in S$, party P_i estimates the round number t_i when the secure function evaluation of F' using the ideal functionality $\mathcal{F}_{\mathsf{MPC}}$ will be complete. It then constructs

$$X_i = \left(x_i, t_i, \left\{v_{i,j}, c_{i,j}^{v,i}, \omega_{i,j}^{v,i}, c_{j,i}^{v,i}\right\}_{j \in S \setminus \{i\}}\right)$$

- The parties invoke the ideal functionality $\mathcal{F}_{\mathsf{MPC}}$ with inputs $\{(X_i, F')\}_{i \in S}$. If the ideal functionality returns $\perp$ after the first stage to party P_i, then P_i aborts for any $i \in [n]$[12]. Otherwise, party P_i receives output

$$\left(\alpha_i, \mathsf{id}, t, \mathsf{ct}, \{\mathsf{ct}_{i,j}\}_{j \in S \setminus \{i\}}, k_{i,i}, \omega_i^{\pi}, \pi_i\right)$$

for each $i \in S$. If the ideal functionality returns z after the third stage, in round t, to party P_i for any $i \in S$, then P_i parses z as $z_1 \| \ldots \| z_n$ where $|z_j| = |y_j|$ for all $j \in [n]$ and computes $y_i = z_{\mathsf{ind}_i} \oplus \alpha_i$ to obtain the output of the computation, where ind_i is the index of i in S. If the ideal functionality returned $\perp$ after the third stage but did not return $\perp$ after the second stage to party P_i, then party P_i also receives $\overrightarrow{c^{\pi}}$ by round t. Then, in round $r = t + 1$, P_i uses the witness

$$w = \left(\mathsf{id}, t, \overrightarrow{c^{\pi}}, \omega_i^{\pi}, \pi_i, i\right)$$

[12]In the $\mathcal{F}_{\mathsf{OT}}$-hybrid model, let $\pi_{F'}$ denote the protocol for the functionality F' defined in Lemma 16. The parties execute $\pi_{F'}$. If the execution of $\pi_{F'}$ aborts, we are assuming that all (honest) parties are aware of the round when the execution of $\pi_{F'}$ aborts, that is, when the adversary has decided to abort the execution of $\pi_{F'}$. Since we are working in the $\mathcal{F}_{\mathsf{MPC}}$-hybrid model, we know that in the ideal model, this is the case when the honest parties receive $\perp$ as their output. If we assume that in the case when the the adversary decides to let the honest parties obtain their outputs, no honest party ever receives $\perp$, this could be used to identify the scenario when the adversary has decided to abort the execution of $\pi_{F'}$. Thus, we could, in principle, replace this instruction with: If party P_i receives $\perp$ as its output, it aborts. Furthermore, since we are considering the case of *unanimous abort*, if the adversary has decided to abort the execution of $\pi_{F'}$, all honest parties abort the protocol.

to invoke the trigger phase of each instance of $\mathcal{F}_{\mathsf{SyX}}$ it is involved in. Once all such instances of $\mathcal{F}_{\mathsf{SyX}}$ involving party P_i have been triggered, P_i obtains $h_{\mathsf{id},i,j}$ for every $j \in S \setminus \{i\}$. Using these values, it computes encryption key-pairs $(\mathsf{pk}_{i,j}, \mathsf{sk}_{i,j})$ by invoking $\mathsf{Gen}(1^\lambda; h_{\mathsf{id},i,j})$ for every $j \in S \setminus \{i\}$. Then, it computes

$$\mathsf{k}_{i,j} = \mathsf{Dec}(\mathsf{ct}_{i,j}; \mathsf{sk}_{i,j})$$

for every $j \in S \setminus \{i\}$. Then, it computes

$$\mathsf{sk} = \bigoplus_{j \in S} k_{i,j}$$

and

$$z = \mathsf{Dec}(\mathsf{ct}; \mathsf{sk})$$

Finally, P_i parses z as $z_1 \| \ldots \| z_n$ where $|z_j| = |y_j|$ for all $j \in [n]$ and computes the value $y_i = z_{\mathsf{ind}_i} \oplus \alpha_i$ to obtain its output, where ind_i is the index of i in S. Otherwise, if the ideal functionality returned $\perp$ after both the second and third stages, the protocol proceeds as follows.

- If $r > t + n$, abort. Otherwise, wait until $r > t$. While $t + 1 \leq r \leq t + n$,

  - If party P_a for $a \in S$ has not received the output of the computation and an instance of $\mathcal{F}_{\mathsf{SyX}}$ involving party P_a is first triggered in round $t + 1 \leq r \leq t + n - 1$ using a "good" witness, it triggers each instance of $\mathcal{F}_{\mathsf{SyX}}$ that it is involved in during round $r + 1$ using the output out it receives from the instance of $\mathcal{F}_{\mathsf{SyX}}$ as follows:

    * If

      $$\mathsf{out}_1 = \left(\mathsf{id}, t, \overrightarrow{c^\pi}, \overrightarrow{\omega^\pi}, \overrightarrow{\pi}, \overrightarrow{\mathsf{ind}}\right)$$

      it prepares the witness

      $$w = \left(\mathsf{id}, t, \overrightarrow{c^\pi}, \overrightarrow{\omega^\pi}', \overrightarrow{\pi}', \overrightarrow{\mathsf{ind}'}\right)$$

where

- $|\overrightarrow{\pi}'| = r+1,\ \overrightarrow{\pi}'|_{[r]} = \overrightarrow{\pi}|_{[r]},\ \pi'_{r+1} = \pi_a$
- $\left|\overrightarrow{\omega^{\pi}}'\right| = r+1,\ \overrightarrow{\omega^{\pi}}'\Big|_{[r]} = \overrightarrow{\omega^{\pi}}\Big|_{[r]},\ {\omega^{\pi}_{r+1}}' = \omega^{\pi}_a$
- $\left|\overrightarrow{\mathsf{ind}}'\right| = r+1,\ \overrightarrow{\mathsf{ind}}'\Big|_{[r]} = \overrightarrow{\mathsf{ind}}\Big|_{[r]},\ \mathsf{ind}'_{r+1} = a$

Once all instances of $\mathcal{F}_{\mathsf{SyX}}$ involving party P_a have been triggered, P_a obtains $h_{\mathsf{id},a,b}$ for every $b \in S \setminus \{a\}$. Using these values, it computes encryption key-pairs $(\mathsf{pk}_{a,b}, \mathsf{sk}_{a,b})$ by invoking $\mathsf{Gen}(1^\lambda; h_{\mathsf{id},a,b})$ for every $b \in S \setminus \{a\}$. Then, it computes

$$\mathsf{k}_{a,b} = \mathsf{Dec}(\mathsf{ct}_{a,b}; \mathsf{sk}_{a,b})$$

for every $b \in S \setminus \{a\}$. Then, it computes

$$\mathsf{sk} = \bigoplus_{b \in S} k_{a,b}$$

and

$$z = \mathsf{Dec}(\mathsf{ct}; \mathsf{sk})$$

Finally, P_a parses z as $z_1 \| \ldots \| z_n$ where $|z_j| = |y_j|$ for all $j \in [n]$ and computes the value $y_a = z_{\mathsf{ind}_a} \oplus \alpha_a$ to obtain its output, where ind_a is the index of a in S.

5.5.3 Correctness

We sketch the proof of correctness of the above protocol. The correctness of the computation of the functionality F' follows by definition from the correctness of the ideal functionality $\mathcal{F}_{\mathsf{MPC}}$. Furthermore, we have that at the end of the invocation of the ideal functionality $\mathcal{F}_{\mathsf{MPC}}$, either all honest parties *unanimously abort* or all honest parties *unanimously continue*. Thus, assuming that $\mathcal{F}_{\mathsf{MPC}}$ did not abort, and that all honest parties continue, every party receives the output of the first stage of F'. If the honest parties also receive the output of the third stage of F', there is nothing to prove and correctness is obvious. This is the optimistic setting. We now consider

the case that all honest parties obtained the output of the first and second stages of F' but not the third stage of F'. Note that a valid witness to trigger an instance of $\mathcal{F}_{\mathsf{SyX}}$ in round $t+1 \leq r \leq t+n$ consists of, apart from id and t, a set of proof values and openings $\left(\overrightarrow{\omega^{\pi}}, \overrightarrow{\pi}\right)$ of size $r-t$. By valid, we mean that the values of $\left(\mathsf{id}, t, \overrightarrow{c^{\pi}}\right)$ are the ones picked during the execution of F'. Triggering the instances of $\mathcal{F}_{\mathsf{SyX}}$ with other values will produce some output but will not be useful because the values $h_{i,j}$ are computed by hashing $V_{i,j}$ along with $\left(\mathsf{id}, t, \overrightarrow{c^{\pi}}\right)$. Also note that if and only if a party does manage to have all the instances of $\mathcal{F}_{\mathsf{SyX}}$ that it is involved in triggered with valid witnesses during the course of the protocol, it is able to reconstruct its output correctly. If the honest party P_i obtained the output of the second stage of F', then P_i is able to prepare the valid witness $w = \left(\mathsf{id}, t, \overrightarrow{c^{\pi}}, \omega_i^{\pi}, \pi_i, i\right)$ and trigger all instances of $\mathcal{F}_{\mathsf{SyX}}$ that it is involved in in round $r = t+1$, and is thus able to reconstruct its output correctly. Suppose the honest parties receive the output of the first stage but not the second or third stages of F'. It is now possible that the adversary has received its output corresponding to the first and second stages of F'. At this point, if the adversary wishes to learn the output, it must do so by obtaining $k_{i,j}$s for $i \in \mathcal{I}$ and $j \in S \setminus \mathcal{I}$. For this, it must be the case that for some corrupt party P_i with $i \in \mathcal{I}$, all instances of $\mathcal{F}_{\mathsf{SyX}}$ involving P_i and an honest party must be triggered, either by the honest parties or by P_i. Since none of the honest parties possess valid witnesses to trigger an instance of $\mathcal{F}_{\mathsf{SyX}}$ (as they do not have $\overrightarrow{c^{\pi}}$), the corrupt party must be the first to trigger. P_i would only have at most $|\mathcal{I}|$ openings to commitments in $\overrightarrow{c^{\pi}}$ and thus, if the adversary learns the output, it must be the case that P_i triggers an instance of $\mathcal{F}_{\mathsf{SyX}}$ with an honest party P_j for some $j \in S \setminus \mathcal{I}$ in round $r \leq t + |\mathcal{I}| \leq t+n-1$. At this point, P_j learns $\overrightarrow{c^{\pi}}$ and a set of openings to commitments in $\overrightarrow{c^{\pi}}$ corresponding to some set of $r-t$ indices $\overrightarrow{\mathsf{ind}}$ in $S \setminus \{j\}$. Combining this information with id, t and the opening $\left(\omega_j^{\pi}, \pi_j\right)$ received from the output of the first stage of F', P_j is able to prepare a valid witness to trigger all instances of $\mathcal{F}_{\mathsf{SyX}}$ that it is involved in in round $r+1 \leq t+n$ and is thus able to reconstruct its output correctly. Finally, consider any honest party P_j for $j \in S \setminus \mathcal{I}$. If no instance of $\mathcal{F}_{\mathsf{SyX}}$ involving P_j is triggered in a round $r \leq t+n-1$, then for all $i \neq j$, party P_i does not

learn $h_{i,j}$ and hence, does not learn $k_{i,j}$. Without $k_{i,j}$, P_i does not learn sk and hence does not learn its output. In particular, this means that the adversary does not learn its output. Thus, if the adversary learns its output, that for all honest parties P_j for $j \in S \setminus \mathcal{I}$, some instance of $\mathcal{F}_{\mathsf{SyX}}$ involving P_j is triggered in a round $r \leq t + n - 1$ which in turns means that P_j learns its output as well. This completes the proof of correctness.

5.5.4 Security

We now prove the following lemma.

Lemma 31. *If* $(\mathsf{Com}, \mathsf{Open})$ *is a commitment scheme,* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is a non-interactive non-committing encryption scheme and* H *a random oracle, then the protocols* $\Pi_{\mathsf{Preprocess}}, \Pi_{\mathsf{FMPC\text{-}preprocess}}$ *securely preprocess for and compute an arbitrary (polynomial) number of instances of* $\mathcal{F}_{\mathsf{MPC}}$ *with fairness in the* $(\mathcal{F}_{\mathsf{MPC}}, \mathcal{F}_{\mathsf{SyX}})$*-hybrid model.*

Proof. Let $\mathcal{A}$ be an adversary attacking the execution of the protocols described in Section 5.5.2 in the $(\mathcal{F}_{\mathsf{bc}}, \mathcal{F}_{\mathsf{MPC}}, \mathcal{F}_{\mathsf{SyX}})$-hybrid model. We construct an ideal-model adversary $\mathcal{S}$ in the ideal model of type fair. Let $\mathcal{I}$ be the set of corrupted parties. If $\mathcal{I}$ is empty, then there is nothing to simulate. $\mathcal{S}$ begins by simulating the first stage, namely, preprocessing.

STAGE I: PREPROCESSING

$\mathcal{S}$ has to simulate the invocations of the load phases of the instances of the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$ that involve corrupt parties. Here, $\mathcal{S}$ behaves as the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$. Recall that the type of $\mathcal{F}_{\mathsf{SyX}}$ is $\mathsf{g.d.}$. For any $a, b \in [n]$ with $a < b$ and $a \in \mathcal{I}$ and $b \in [n] \setminus \mathcal{I}$, if $\mathcal{A}$ instructs P_a to invoke the load phase of $\mathcal{F}_{\mathsf{SyX}}^{a,b}$ with no input, $\mathcal{S}$ executes $f_1^{a,b}$ as defined in $\Pi_{\mathsf{Preprocess}}$. $\mathcal{S}$ picks two strings $v_{a,b}, v_{b,a} \xleftarrow{\$} \{0,1\}^\lambda$ and computes and stores

$$V_{a,b} = V_{b,a} = v_{a,b} \oplus v_{b,a}$$

It also computes

$$(c^v_{a,b}, \omega^v_{a,b}) \xleftarrow{\$} \mathsf{Com}(v_{a,b})$$

and

$$(c^v_{b,a}, \omega^v_{b,a}) \xleftarrow{\$} \mathsf{Com}(v_{b,a})$$

Finally, it forwards $(v_{a,b}, c^v_{a,b}, \omega^v_{a,b}, c^v_{b,a})$ to the adversary. $\mathcal{S}$ behaves symmetrically if for any $a, b \in [n]$ with $a < b$ and $b \in \mathcal{I}$ and $a \in [n] \setminus \mathcal{I}$, if $\mathcal{A}$ instructs P_b to invoke the load phase of $\mathcal{F}^{a,b}_{\mathsf{SyX}}$. The final case to consider is if for any $a, b \in [n]$ with $a < b$ and $a, b \in \mathcal{I}$, if $\mathcal{A}$ instructs P_a, P_b to invoke the load phase of $\mathcal{F}^{a,b}_{\mathsf{SyX}}$ with no input, $\mathcal{S}$ executes $f_1^{a,b}$ as defined in $\Pi_{\mathsf{Preprocess}}$. $\mathcal{S}$ picks two strings $v_{a,b}, v_{b,a} \xleftarrow{\$} \{0,1\}^\lambda$ and computes and stores

$$V_{a,b} = V_{b,a} = v_{a,b} \oplus v_{b,a}$$

It also computes

$$(c^v_{a,b}, \omega^v_{a,b}) \xleftarrow{\$} \mathsf{Com}(v_{a,b})$$

and

$$(c^v_{b,a}, \omega^v_{b,a}) \xleftarrow{\$} \mathsf{Com}(v_{b,a})$$

Finally, it forwards $v_{a,b}, v_{b,a}, c^v_{a,b}, \omega^v_{a,b}, c^v_{b,a}, \omega^v_{b,a}$ to the adversary. If in any of the invocations, the adversary instructs a corrupt party to invoke the load phase of an instance of $\mathcal{F}_{\mathsf{SyX}}$ incorrectly, $\mathcal{S}$ simply aborts simulating the load phase of this particular instance of $\mathcal{F}_{\mathsf{SyX}}$. At the end of this phase, let $\mathsf{PreprocessSuccess}$ denote the set of all $\{a, b\}$ with $a < b$ such that P_a and P_b successfully invoked the load phase of $\mathcal{F}^{a,b}_{\mathsf{SyX}}$.

STAGE II: FAIR SECURE FUNCTION EVALUATION

Let F be the n-input n-output functionality to be computed. Let S be the set of n parties that are participating in this instance of fair secure function evaluation. If $S \cap \mathcal{I} = \phi$, then there is nothing to simulate. $\mathcal{S}$ then checks that for every $a, b \in S$ with $a < b$ and either $a \in [N] \setminus \mathcal{I}$ or $b \in [N] \setminus \mathcal{I}$, $\{a, b\} \in \mathsf{PreprocessSuccess}$. If not, then $\mathcal{S}$ aborts this instance of fair secure function evaluation as the honest parties would

have done so in the real execution as well. $\mathcal{S}$ then begins by simulating the first step of the protocol, namely, the invocation of the ideal functionality $\mathcal{F}_{\mathsf{MPC}}$. Here, $\mathcal{S}$ behaves as the ideal functionality $\mathcal{F}_{\mathsf{MPC}}$. Recall that the type of $\mathcal{F}_{\mathsf{MPC}}$ is abort. $\mathcal{S}$ obtains the inputs $\{(X_i, f_i)\}_{i \in \mathcal{I}}$ of the corrupted parties from $\mathcal{A}$. If $(X_i, f_i) = \mathsf{abort}$ for any $i \in \mathcal{I}$, $\mathcal{S}$ forwards $\{(X_i, f_i)\}_{i \in \mathcal{I}}$ to the trusted party computing $\mathcal{F}_{\mathsf{MPC}}$ with fairness, receives $\perp$ as the output of all parties, which it forwards $\mathcal{A}$. Suppose $(X_i, f_i) \neq \mathsf{abort}$ for all $i \in \mathcal{I}$. If there exists a $j \in \mathcal{I}$ such that $f_j \neq F'$ as defined in protocol $\Pi_{\mathsf{FMPC\text{-}preprocess}}$, $\mathcal{S}$ forwards $\{(X_i, f_i)\}_{i \in \mathcal{I}}$ to the trusted party computing $\mathcal{F}_{\mathsf{MPC}}$ with fairness, which aborts, and then aborts this instance of fair secure function evaluation. If there exists a $j \in \mathcal{I}$ such that (X_j, f_j) is not of the specified format, $\mathcal{S}$ replaces (X_j, f_j) with a default value. Going forward, we assume that for all $i \in \mathcal{I}$, (X_i, f_i) is well-formed, that is,

$$X_i = \left(x_i, t_i, \left\{v_{i,j}^*, c_{i,j}^{*,v,i}, \omega_{i,j}^{*,v,i}, c_{j,i}^{*,v,i}\right\}_{j \in S \setminus \{i\}}\right)$$

$$X_i = \left(x_i, t_i, \{v_{i,j}^*\}_{j \in S \setminus \{i\}}\right)$$

and that $f_i = F'$ as defined in $\Pi_{\mathsf{FMPC\text{-}preprocess}}$. Note that $v_{i,j}^*$ may not equal $v_{i,j}$ as picked by $\mathcal{S}$ in the simulation of the preprocessing stage, $c_{i,j}^{*,v,i}$ may not equal $c_{i,j}^{v,i}$, and so on.

$\mathcal{S}$ now needs to simulate the outputs received by the corrupted parties from the ideal functionality $\mathcal{F}_{\mathsf{MPC}}$. $\mathcal{S}$ estimates the round number t when the secure function evaluation of F' using the ideal functionality $\mathcal{F}_{\mathsf{MPC}}$ will be complete. It checks that $t = t_i$ for all $i \in \mathcal{I}$ and aborts this instance of fair secure function evaluation otherwise. It then checks that $c_{i,j}^{*,v,i} = c_{i,j}^{v,i}$, $c_{j,i}^{*,v,i} = c_{j,i}^{v,i}$ and that $\mathsf{Open}\left(c_{i,j}^{v,i}, \omega_{i,j}^{*,v,i}, v_{i,j}^*\right) = 1$ for all $i \in \mathcal{I}$ and $j \in S \setminus \{i\}$. If not, it aborts this instance of fair secure function evaluation. Note that if these checks pass, we also have that $v_{i,j}^* = v_{i,j}$ and $\omega_{i,j}^{*,v,i} = \omega_{i,j}^{v,i}$ for all $i \in \mathcal{I}$ and $j \in S \setminus \{i\}$. It then samples a random identifier $\mathsf{id} \in \{0,1\}^\lambda$ for this instance of secure function evaluation. For each $i \in S$, $\mathcal{S}$ samples a random string $\alpha_i \xleftarrow{\$} \{0,1\}^*$ of length equal to the length of the ith output of F. Let

$$\alpha = \alpha_1 \| \ldots \| \alpha_n$$

$\mathcal{S}$ samples a random encryption key-pair $(\mathsf{pk}, \mathsf{sk})$ by invoking $\mathsf{Gen}(1^\lambda)$ and a ciphertext ct representing the encryption of the output z under a secret key sk. Note that z is not known to $\mathcal{S}$ at this point. However, since $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a non-committing encryption scheme, $\mathcal{S}$ can sample ct and later *equivocate* it. For each $i \in \mathcal{I}$, $\mathcal{S}$ samples random additive n-out-of-n secret sharings $k_{i,1}, \ldots, k_{i,n}$ of sk such that

$$\mathsf{sk} = \bigoplus_{j \in [n]} k_{i,j}$$

$\mathcal{S}$ samples random proof values $\pi_1, \ldots, \pi_n \xleftarrow{\$} \{0,1\}^\lambda$ and compute commitments along with their openings $(c_i^\pi, \omega_i^\pi) \xleftarrow{\$} \mathsf{Com}(\pi_i)$ to each of the proof values π_i. Let

$$\overrightarrow{c^\pi} = (c_1^\pi, \ldots, c_n^\pi)$$

and

$$\overrightarrow{\omega^\pi} = (\omega_1^\pi, \ldots, \omega_n^\pi)$$

and

$$\overrightarrow{\pi} = (\pi_1, \ldots, \pi_n)$$

$\mathcal{S}$ then computes

$$h_{i,j} = h_{j,i} = H\left(V_{i,j} \| \mathsf{id} \| t \| \overrightarrow{c^\pi}\right)$$

where

$$V_{i,j} = V_{j,i} = v_{i,j} \oplus v_{j,i}$$

for every $i \in \mathcal{I}$ and $j \in S$ with $i \neq j$. $\mathcal{S}$ then samples encryption key-pairs $(\mathsf{pk}_{i,j}, \mathsf{sk}_{i,j})$ by invoking $\mathsf{Gen}(1^\lambda; h_{i,j})$ for every $i \in \mathcal{I}$ and $j \in S$ with $i \neq j$, and computes

$$\mathsf{ct}_{i,j} = \mathsf{Enc}(k_{i,j}; \mathsf{pk}_{i,j})$$

for every $i \in \mathcal{I}$ and $j \in S$ with $i \neq j$. Thus, the simulator constructs the first stage output

$$\left(\alpha_i, \mathsf{id}, t, \mathsf{ct}, \{\mathsf{ct}_{i,j}\}_{j \in S \setminus \{i\}}, k_{i,i}, \omega_i^\pi, \pi_i\right)$$

for each $i \in \mathcal{I}$ and forwards it to $\mathcal{A}$. If $\mathcal{A}$ then sends abort, $\mathcal{S}$ forwards $\{(x_i, f_i)\}_{i\in\mathcal{I}}$ to the trusted party computing $\mathcal{F}_{\mathsf{MPC}}$ with fairness, with (x_j, f_j) replaced with abort for some $j \in \mathcal{I}$, receives $\perp$ as the output of all parties, which it forwards $\mathcal{A}$. Otherwise, $\mathcal{A}$ responds with $\mathsf{continue}$. $\mathcal{S}$ then forwards $\overrightarrow{c^{\pi}}$ to $\mathcal{A}$.

Case A. The adversary lets the honest parties obtain the output of the second stage of F'. In this case, $\mathcal{A}$ has responded with $\mathsf{continue}$ after receiving $\overrightarrow{c^{\pi}}$. At this point, all parties are going to obtain their outputs. $\mathcal{S}$ forwards $\{(x_i, f_i)\}_{i\in\mathcal{I}}$ to the trusted party computing $\mathcal{F}_{\mathsf{MPC}}$ with fairness. It receives the corrupt parties outputs, namely, $\{y_i\}_{i\in\mathcal{I}}$. $\mathcal{S}$ chooses the outputs of the honest party completely at random, that is, it samples random strings $y_i \xleftarrow{\$} \{0,1\}^*$ of length equal to the length of the ith output of F, for $i \in [n] \setminus \mathcal{I}$. $\mathcal{S}$ then constructs

$$y = y_1 \| \ldots \| y_n$$

It then defines

$$z = y \oplus \alpha$$

$\mathcal{S}$ now ensures that $\mathsf{Dec}(\mathsf{ct}; \mathsf{sk}) = z$. $\mathcal{S}$ then sends the outputs of the corrupt parties, namely, $\{y_i\}_{i\in\mathcal{I}}$, to $\mathcal{A}$. At this point, $\mathcal{S}$ has completed simulating the invocation of the ideal functionality $\mathcal{F}_{\mathsf{MPC}}$. If $\mathcal{A}$ responds with $\mathsf{continue}$, then $\mathcal{S}$ simply terminates. Otherwise, in round $r = t + 1$, it simulates the honest parties triggering all the instances of $\mathcal{F}_{\mathsf{SyX}}$ they are involved in with the corrupt parties and hence for every $i \in \mathcal{I}$ and $j \in S \setminus \mathcal{I}$, $\mathcal{S}$ sends P_i (the adversary $\mathcal{A}$) the set of values $\left(\left(\mathsf{id}, t, \overrightarrow{c^{\pi}}, \omega_j^{\pi}, \pi_j, j\right), h_{i,j}\right)$.

Case B. The adversary does not let the honest parties obtain the output of the second stage of F'. In this case, $\mathcal{A}$ has responded with abort after receiving $\overrightarrow{c^{\pi}}$. At this point, $\mathcal{S}$ has completed simulating the invocation of the ideal functionality $\mathcal{F}_{\mathsf{MPC}}$. We first discuss how $\mathcal{S}$ simulates certain invocations of the trigger phases of the instances of the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$ that the adversary instructs the corrupt parties to trigger.

- Suppose the adversary instructs a corrupt party, say P_i for $i \in \mathcal{I}$, to trigger an instance of $\mathcal{F}_{\mathsf{SyX}}$ involving another corrupt party, say P_j for $j \in \mathcal{I}$, with a *valid* witness. $\mathcal{S}$ sends $(w, h_{i,j})$ to parties P_i and P_j.

- Suppose the adversary instructs a corrupt party to trigger an instance of $\mathcal{F}_{\mathsf{SyX}}$ with an *invalid* witness. $\mathcal{S}$ simply sends no response.

Suppose the adversary does not instruct a corrupt party, say P_i for some $i \in \mathcal{I}$, to trigger an instance of $\mathcal{F}_{\mathsf{SyX}}$ involving an honest party, say P_j for some $j \in S \setminus \mathcal{I}$, with a *valid* witness and the round counter exceeds $t + n$, $\mathcal{S}$ forwards $\{(x_i, f_i)\}_{i \in \mathcal{I}}$ to the trusted party computing $\mathcal{F}_{\mathsf{MPC}}$ with fairness, with (x_j, f_j) replaced with abort for some $j \in \mathcal{I}$, receives $\perp$ as the output of all parties, and aborts itself. Otherwise, at the first instant $r \leq t + n - 1$ that the adversary instructs a corrupt party P_i for $i \in \mathcal{I}$ to trigger an instance of $\mathcal{F}_{\mathsf{SyX}}$ involving an honest party P_j for $j \in S \setminus \mathcal{I}$ with a *valid* witness $w = \left(\mathsf{id}, t, \overrightarrow{c^{\pi}}, \overrightarrow{\omega^{\pi}}, \overrightarrow{\pi}, \overrightarrow{\mathsf{ind}}\right)$, $\mathcal{S}$ forwards $\{(x_i, f_i)\}_{i \in \mathcal{I}}$ to the trusted party computing $\mathcal{F}_{\mathsf{MPC}}$ with fairness. It receives the corrupt parties outputs, namely, $\{y_i\}_{i \in \mathcal{I}}$. $\mathcal{S}$ chooses the outputs of the honest party completely at random, that is, it samples random strings $y_i \xleftarrow{\$} \{0,1\}^*$ of length equal to the length of the ith output of F, for $i \in [n] \setminus \mathcal{I}$. $\mathcal{S}$ then constructs

$$y = y_1 \| \ldots \| y_n$$

It then defines

$$z = y \oplus \alpha$$

$\mathcal{S}$ now ensures that $\mathsf{Dec}(\mathsf{ct}; \mathsf{sk}) = z$. $\mathcal{S}$ then sends $(w, h_{i,j})$ to P_i. In round $r + 1$, it simulates the P_j triggering all the instances of $\mathcal{F}_{\mathsf{SyX}}$ they are involved in with the corrupt parties and hence for every $i \in \mathcal{I}$, $\mathcal{S}$ sends P_i the set of values

$$\left(\left(\mathsf{id}, t, \overrightarrow{c^{\pi}}, \overrightarrow{\omega^{\pi}}', \overrightarrow{\pi}', \overrightarrow{\mathsf{ind}'}\right), h_{i,j}\right)$$

where

- $|\overrightarrow{\pi}'| = r+1,\ \overrightarrow{\pi}'|_{[r]} = \overrightarrow{\pi}|_{[r]},\ \pi'_{r+1} = \pi_j$
- $\left|\overrightarrow{\omega^{\pi}}'\right| = r+1,\ \overrightarrow{\omega^{\pi}}'\Big|_{[r]} = \overrightarrow{\omega^{\pi}}\Big|_{[r]},\ {\omega^{\pi}_{r+1}}' = \omega^{\pi}_j$
- $\left|\overrightarrow{\mathsf{ind}}'\right| = r+1,\ \overrightarrow{\mathsf{ind}}'\Big|_{[r]} = \overrightarrow{\mathsf{ind}}\Big|_{[r]},\ \mathsf{ind}'_{r+1} = j$

Finally, for every $k \in S \setminus \mathcal{I}$ such that P_k did not have an instance of $\mathcal{F}_{\mathsf{SyX}}$ involving itself and some corrupt party triggered in round r, $\mathcal{S}$ simulates P_k triggering all instances of $\mathcal{F}_{\mathsf{SyX}}$ involving P_k and every corrupt party in round $r+2$. Note that by the existence of j, k, $|\mathcal{I}| \leq n-2$ and hence $r \leq t+n-2$, or, $r+2 \leq t+n$. To simulate the triggers, $\mathcal{S}$ sends along the appropriate valid witnesses and $h_{i,k}$s. Note that this is possible to do as by this point, $\mathcal{S}$ has all the values it will ever need in the simulation. Going forward, $\mathcal{S}$ simulates invocations of the trigger phases of the instances of the ideal functionality $\mathcal{F}_{\mathsf{SyX}}$ that the adversary instructs corrupt parties to trigger as follows.

- Suppose the adversary instructs a corrupt party, say P_i for $i \in \mathcal{I}$, to trigger an instance of $\mathcal{F}_{\mathsf{SyX}}$ involving another corrupt party, say P_j for $j \in \mathcal{I}$, with a *valid* witness w, $\mathcal{S}$ sends $(w, h_{i,j})$ to parties P_i and P_j.
- Suppose the adversary instructs a corrupt party, say P_i for $i \in \mathcal{I}$, to trigger an instance of $\mathcal{F}_{\mathsf{SyX}}$ involving an honest party, say P_j for $j \in S \setminus \mathcal{I}$, with a *valid* witness w, $\mathcal{S}$ sends $(w, h_{i,j})$ to P_i.
- Suppose the adversary instructs a corrupt party to trigger an instance of $\mathcal{F}_{\mathsf{SyX}}$ with an *invalid* witness. $\mathcal{S}$ simply sends no response.

Finally, $\mathcal{S}$ outputs whatever $\mathcal{A}$ outputs. It is easy to see that the view of $\mathcal{A}$ is indistinguishable in the execution of the protocols $\Pi_{\mathsf{Preprocess}}, \Pi_{\mathsf{FMPC\text{-}preprocess}}$ and the simulation with $\mathcal{S}$, if $(\mathsf{Com}, \mathsf{Open})$ is a commitment scheme$(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a non-interactive non-committing encryption scheme and H a random oracle. We therefore conclude that the protocols $\Pi_{\mathsf{Preprocess}}, \Pi_{\mathsf{FMPC\text{-}preprocess}}$ securely preprocess for and compute an arbitrary (polynomial) number of instances of $\mathcal{F}_{\mathsf{MPC}}$ with fairness in the $(\mathcal{F}_{\mathsf{MPC}}, \mathcal{F}_{\mathsf{SyX}})$-hybrid model, as required. □

Remark. In the proof of Lemma 30, we ignore some annoying technicalities. For instance, the adversary may cause the honest parties to abort, will be unable to obtain its output but still pointlessly interact with some of the ideal functionalities. In the proof, however, the simulator would have aborted. We note that these details are not particularly enlightening and are of no consequence. One can deal with these sorts of attacks by asking the simulator to wait in these scenarios until the adversary says that it is done and then finally abort if it has to. Thus, we assume, for the purpose of the proof, that if the adversary forces the honest parties to abort in a situation where it will be unable to obtain its output, without loss of generality, it halts. Other examples of such technicalities are when the adversary sends "unexpected" messages, "incomplete" messages, etc. Note that such messages can be easily detected and ignored, and do not affect the protocol in any way.

5.5.5 Getting to the $\mathcal{F}_{\mathsf{SyX}}$-hybrid model

Combining Lemmas 1, 16 and 31, we obtain the following theorem.

Theorem 18. *Consider n parties $P_1, \ldots, P_n$ in the* point-to-point *model. Then, assuming the existence of one-way permutations, there exists a protocol π in the programmable random oracle model which securely preprocesses for and computes an arbitrary (polynomial) number of instances of $\mathcal{F}_{\mathsf{MPC}}$ with fairness in the presence of t-threshold adversaries for any $0 \leq t < n$ in the $(\mathcal{F}_{\mathsf{OT}}, \mathcal{F}_{\mathsf{SyX}})$-hybrid model.*

As discussed in Section 5.2, $\mathcal{F}_{\mathsf{2PC}}$, and hence $\mathcal{F}_{\mathsf{OT}}$, can be realized in the $\mathcal{F}_{\mathsf{SyX}}$-hybrid model. We thus have the following theorem.

Theorem 19. *Consider n parties $P_1, \ldots, P_n$ in the* point-to-point *model. Then, assuming the existence of one-way permutations, there exists a protocol π in the programmable random oracle model which securely preprocesses for and computes an arbitrary (polynomial) number of instances of $\mathcal{F}_{\mathsf{MPC}}$ with fairness in the presence of t-threshold adversaries for any $0 \leq t < n$ in the $\mathcal{F}_{\mathsf{SyX}}$-hybrid model.*

Chapter 6

Conclusions and further work

In this work, we make significant progress towards the goal of identifying optimal infrastructures for secure multiparty computation. In the information-theoretic setting, we improve upon the state-of-the-art networks and protocols for almost-everywhere reliable message transmission, both in achieving logarithmic degree networks and protocols over them that are polylogarithmic in work complexity (an exponential improvement over all current works). Our results, however, do not completely characterize sparse networks in which communication would be possible. In particular, it is unclear whether constant degree networks with efficient protocols are achievable. In light of [122], we know that constant degree networks with exponential work protocols are achievable. Thus, characterizing communication networks precisely might pose the difficulty of potentially overcoming complexity-theoretic challenge in some way. In fact, one might also expect results similar to the case of oblivious transfer infrastructures as described ahead.

We completely characterize infrastructures for oblivious transfer. The only questions left open here are connected with testability, that is, given a network of OT channels and a pair of parties, can we test whether there exists an OT protocol with one of the parties as the sender and the other as the receiver. The characterization of infrastructures is by a property of graphs, testing for which is known to be coNP-complete. While this does not impede the correctness of our protocols, our protocols are inefficient (exponential time) in general, while they are efficient precisely when

testing the graph property is efficient. It is unclear whether this is inherent, that is, is it possible to have efficient protocols even when determining whether they succeed may be inefficient. In other words, is it the case that protocols over an oblivious transfer infrastructure can be used to test whether they succeed. This would be a very interesting follow-up question as either answer seems surprising.

Finally, we make the first steps toward identifying infrastructures for fair secure computation. We introduce a new 2-party primitive, synchronizable fair exchange, that suffices for fair secure multiparty computation, and which can also be preprocessed for arbitrary and arbitrarily many instances of fair secure computation. Truly, synchronizable exchange is to fair secure computation what oblivious transfer is to (unfair) secure computation. This paves the way for a lot of great research. For instance, what are the truly minimal infrastructures for this primitive? How do we realize this primitive? We have parts of the answers to both these questions in follow-up works that are in preparation.

In conclusion, the results presented in this work say a lot about building powerful and efficient infrastructures for multiparty computation. We recall the desiderata of an infrastructure that we laid out in Chapter 1.

- *Reusability/Amortization.* Setting up an infrastructure component could be expensive, but using it and maintaining it should be inexpensive relative to setting up a new component. We achieve this property by preprocessing each of the components of our infrastructure.

- *Transferability/Routing.* It should be possible to combine different components of the infrastructure to deliver benefits to the end users. This property was at the heart of design process, all along. For each of the building block cryptographic primitives, we start with a network where a subset of all possible pairs of parties share the primitive. Through our protocols, we enable several other pairs of parties to also "realize" the same primitive by invoking the existing instances of the primitive among other parties and some fairly inexpensive operations.

- *Robustness/Fault-tolerance.* Failure or unavailability of some components of the infrastructure should not nullify the usefulness of the infrastructure. As shown in each chapter, our infrastructures are resilient to adversary classes belonging to various corruption models.

The works in this thesis have certainly laid the foundations for building powerful and efficient infrastructures for multiparty computation. It is our strong belief that pushing this research forward is a way towards having large-scale practical multiparty computation truly see the light of day.

Bibliography

[1] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Lukasz Mazurek. Fair two-party computations via bitcoin deposits. In *Financial Cryptography and Data Security - FC 2014 Workshops, BITCOIN and WAHC 2014, Christ Church, Barbados, March 7, 2014, Revised Selected Papers*, pages 105–121, 2014.

[2] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Lukasz Mazurek. Secure multiparty computations on bitcoin. *Commun. ACM*, 59(4):76–84, 2016.

[3] Benny Applebaum, Yuval Ishai, Eyal Kushilevitz, and Brent Waters. Encoding functions with constant online rate, or how to compress garbled circuit keys. *SIAM J. Comput.*, 44(2):433–466, 2015.

[4] Gilad Asharov. Towards characterizing complete fairness in secure two-party computation. In *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, pages 291–316, 2014.

[5] Gilad Asharov, Amos Beimel, Nikolaos Makriyannis, and Eran Omri. Complete characterization of fairness in secure two-party computation of boolean functions. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part I*, pages 199–228, 2015.

[6] N. Asokan, Matthias Schunter, and Michael Waidner. Optimistic protocols for fair exchange. In *CCS '97, Proceedings of the 4th ACM Conference on Computer and Communications Security, Zurich, Switzerland, April 1-4, 1997.*, pages 7–17, 1997.

[7] N. Asokan, Victor Shoup, and Michael Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications*, 18(4):593–610, 2000.

[8] Boaz Barak, Ran Canetti, Yehuda Lindell, Rafael Pass, and Tal Rabin. Secure computation without authentication. In *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, pages 361–377, 2005.

[9] Donald Beaver. Precomputing oblivious transfer. In *Advances in Cryptology - CRYPTO '95, 15th Annual International Cryptology Conference, Santa Barbara, California, USA, August 27-31, 1995, Proceedings*, pages 97–109, 1995.

[10] Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 479–488, 1996.

[11] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 503–513, 1990.

[12] Zuzana Beerliová-Trubíniová and Martin Hirt. Perfectly-secure MPC with linear communication complexity. In *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008*, pages 213–230, 2008.

[13] Amos Beimel, Yehuda Lindell, Eran Omri, and Ilan Orlov. $1/p$-secure multiparty computation without honest majority and the best of both worlds. In *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, pages 277–296, 2011.

[14] Amos Beimel, Tal Malkin, and Silvio Micali. The all-or-nothing nature of two-party secure computation. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, pages 80–97, 1999.

[15] Amos Beimel, Eran Omri, and Ilan Orlov. Protocols for multiparty coin toss with dishonest majority. In *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, pages 538–557, 2010.

[16] Mihir Bellare, Viet Tung Hoang, Sriram Keelveedhi, and Phillip Rogaway. Efficient garbling from a fixed-key blockcipher. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 478–492, 2013.

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

[18] Michael Ben-Or and Dana Ron. Agreement in the presence of faults, on networks of bounded degree. *Inf. Process. Lett.*, 57(6):329–334, 1996.

[19] Eli Ben-Sasson, Serge Fehr, and Rafail Ostrovsky. Near-linear unconditionally-secure multiparty computation with a dishonest minority. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 663–680, 2012.

[20] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, pages 169–188, 2011.

[21] Iddo Bentov and Ranjit Kumaresan. How to use bitcoin to design fair protocols. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, pages 421–439, 2014.

[22] Piotr Berman and Juan A. Garay. Asymptotically optimal distributed consensus (extended abstract). In *Automata, Languages and Programming, 16th International Colloquium, ICALP89, Stresa, Italy, July 11-15, 1989, Proceedings*, pages 80–94, 1989.

[23] Piotr Berman and Juan A. Garay. Fast consensus in networks of bounded degree (extended abstract). In *Distributed Algorithms, 4th International Workshop, WDAG '90, Bari, Italy, September 24-26, 1990, Proceedings*, pages 321–333, 1990.

[24] Dan Bogdanov, Sven Laur, and Jan Willemson. Sharemind: A framework for fast privacy-preserving computations. In *Computer Security - ESORICS 2008, 13th European Symposium on Research in Computer Security, Málaga, Spain, October 6-8, 2008. Proceedings*, pages 192–206, 2008.

[25] Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas P. Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael I. Schwartzbach, and Tomas Toft. Secure multiparty computation goes live. In *Financial Cryptography and Data Security, 13th International Conference, FC 2009, Accra Beach, Barbados, February 23-26, 2009. Revised Selected Papers*, pages 325–343, 2009.

[26] Dan Boneh and Moni Naor. Timed commitments. In *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*, pages 236–254, 2000.

[27] Elette Boyle, Kai-Min Chung, and Rafael Pass. Large-scale secure computation: Multi-party computation for (parallel) RAM programs. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 742–762, 2015.

[28] Elette Boyle, Shafi Goldwasser, and Stefano Tessaro. Communication locality in secure multi-party computation - how to run sublinear algorithms in a distributed setting. In *Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*, pages 356–376, 2013.

[29] Ran Canetti. Security and composition of cryptographic protocols: a tutorial (part I). *SIGACT News*, 37(3):67–92, 2006.

[30] Ran Canetti, Ivan Damgård, Stefan Dziembowski, Yuval Ishai, and Tal Malkin. On adaptive vs. non-adaptive security of multiparty protocols. In *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*, pages 262–279, 2001.

[31] Ran Canetti and Marc Fischlin. Universally composable commitments. In *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, pages 19–40, 2001.

[32] Ran Canetti, Shai Halevi, and Jonathan Katz. Adaptively-secure, non-interactive public-key encryption. In *Theory of Cryptography, Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005, Proceedings*, pages 150–168, 2005.

[33] Ran Canetti, Hugo Krawczyk, and Jesper Buus Nielsen. Relaxing chosen-ciphertext security. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, pages 565–582, 2003.

[34] Nishanth Chandran, Wutichai Chongchitmate, Juan A. Garay, Shafi Goldwasser, Rafail Ostrovsky, and Vassilis Zikas. The hidden graph model: Communication locality and optimal resiliency with adaptive faults. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*, pages 153–162, 2015.

[35] Nishanth Chandran, Juan A. Garay, and Rafail Ostrovsky. Improved fault tolerance and secure computation on sparse networks. In *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part II*, pages 249–260, 2010.

[36] Nishanth Chandran, Juan A. Garay, and Rafail Ostrovsky. Edge fault tolerance on sparse networks. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part II*, pages 452–463, 2012.

[37] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proceedings of the 20th Annual ACM*

Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA, pages 11–19, 1988.

[38] Benny Chor and Eyal Kushilevitz. A zero-one law for boolean privacy (extended abstract). In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 62–72, 1989.

[39] Tung Chou and Claudio Orlandi. The simplest protocol for oblivious transfer. In *Progress in Cryptology - LATINCRYPT 2015 - 4th International Conference on Cryptology and Information Security in Latin America, Guadalajara, Mexico, August 23-26, 2015, Proceedings*, pages 40–58, 2015.

[40] Arka Rai Choudhuri, Matthew Green, Abhishek Jain, Gabriel Kaptchuk, and Ian Miers. Fairness in an unfair world: Fair multiparty computation from public bulletin boards. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 719–728, 2017.

[41] Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 364–369, 1986.

[42] Ran Cohen and Yehuda Lindell. Fairness versus guaranteed output delivery in secure multiparty computation. *J. Cryptology*, 30(4):1157–1186, 2017.

[43] Ivan Damgård and Yuval Ishai. Scalable secure multiparty computation. In *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, pages 501–520, 2006.

[44] Ivan Damgård, Yuval Ishai, and Mikkel Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, pages 445–465, 2010.

[45] Ivan Damgård, Yuval Ishai, Mikkel Krøigaard, Jesper Buus Nielsen, and Adam D. Smith. Scalable multiparty computation with nearly optimal work and resilience. In *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, pages 241–261, 2008.

[46] Ivan Damgård, Joe Kilian, and Louis Salvail. On the (im)possibility of basing oblivious transfer and bit commitment on weakened security assumptions. In *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, pages 56–73, 1999.

[47] Ivan Damgård and Jesper Buus Nielsen. Scalable and unconditionally secure multiparty computation. In *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, pages 572–590, 2007.

[48] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 643–662, 2012.

[49] Varsha Dani, Valerie King, Mahnush Movahedi, and Jared Saia. Brief announcement: breaking the o(nm) bit barrier, secure multiparty computation with a static adversary. In *ACM Symposium on Principles of Distributed Computing, PODC '12, Funchal, Madeira, Portugal, July 16-18, 2012*, pages 227–228, 2012.

[50] Danny Dolev, Cynthia Dwork, Orli Waarts, and Moti Yung. Perfectly secure message transmission. *J. ACM*, 40(1):17–47, 1993.

[51] Danny Dolev, Michael J. Fischer, Robert J. Fowler, Nancy A. Lynch, and H. Raymond Strong. An efficient algorithm for byzantine agreement without authentication. *Information and Control*, 52(3):257–274, 1982.

[52] Danny Dolev and Rüdiger Reischuk. Bounds on information exchange for byzantine agreement. In *ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, Ottawa, CanadaAugust 18-20, 1982*, pages 132–140, 1982.

[53] Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM J. Comput.*, 12(4):656–666, 1983.

[54] Cynthia Dwork, David Peleg, Nicholas Pippenger, and Eli Upfal. Fault tolerance in networks of bounded degree (preliminary version). In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 370–379, 1986.

[55] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, 1985.

[56] Matthias Fitzi, Matthew K. Franklin, Juan A. Garay, and Harsha Vardhan Simhadri. Towards optimal and efficient perfectly secure message transmission. In *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, pages 311–322, 2007.

[57] Matthias Fitzi, Juan A. Garay, Ueli M. Maurer, and Rafail Ostrovsky. Minimal complete primitives for secure multi-party computation. *J. Cryptology*, 18(1):37–61, 2005.

[58] Matthias Fitzi, Nicolas Gisin, Ueli M. Maurer, and Oliver von Rotz. Unconditional byzantine agreement and multi-party computation secure against dishonest minorities from scratch. In *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, pages 482–501, 2002.

[59] Matthias Fitzi, Daniel Gottesman, Martin Hirt, Thomas Holenstein, and Adam D. Smith. Detectable byzantine agreement secure against faulty majorities. In *Proceedings of the Twenty-First Annual ACM Symposium on Principles of Distributed Computing, PODC 2002, Monterey, California, USA, July 21-24, 2002*, pages 118–126, 2002.

[60] Juan A. Garay and Markus Jakobsson. Timed release of standard digital signatures. In *Financial Cryptography, 6th International Conference, FC 2002, Southampton, Bermuda, March 11-14, 2002, Revised Papers*, pages 168–182, 2002.

[61] Juan A. Garay, Jonathan Katz, Chiu-Yuen Koo, and Rafail Ostrovsky. Round complexity of authenticated broadcast with a dishonest majority. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pages 658–668, 2007.

[62] Juan A. Garay, Jonathan Katz, Ranjit Kumaresan, and Hong-Sheng Zhou. Adaptively secure broadcast, revisited. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing, PODC 2011, San Jose, CA, USA, June 6-8, 2011*, pages 179–186, 2011.

[63] Juan A. Garay, Philip D. MacKenzie, Manoj Prabhakaran, and Ke Yang. Resource fairness and composability of cryptographic protocols. *J. Cryptology*, 24(4):615–658, 2011.

[64] Juan A. Garay and Rafail Ostrovsky. Almost-everywhere secure computation. In *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, pages 307–323, 2008.

[65] Juan A. Garay and Carl Pomerance. Timed fair exchange of standard signatures: [extended abstract]. In *Financial Cryptography, 7th International Conference, FC 2003, Guadeloupe, French West Indies, January 27-30, 2003, Revised Papers*, pages 190–207, 2003.

[66] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[67] Oded Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004.

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

[69] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(3):691–729, 1991.

[70] Oded Goldreich and Ronen Vainish. How to solve any protocol problem - an efficiency improvement. In *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings*, pages 73–86, 1987.

[71] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 555–564, 2013.

[72] S. Dov Gordon, Carmit Hazay, Jonathan Katz, and Yehuda Lindell. Complete fairness in secure two-party computation. *J. ACM*, 58(6):24:1–24:37, 2011.

[73] S. Dov Gordon, Yuval Ishai, Tal Moran, Rafail Ostrovsky, and Amit Sahai. On complete primitives for fairness. In *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings*, pages 91–108, 2010.

[74] S. Dov Gordon and Jonathan Katz. Complete fairness in multi-party computation without an honest majority. In *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, pages 19–35, 2009.

[75] S. Dov Gordon and Jonathan Katz. Partial fairness in secure two-party computation. In *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings*, pages 157–176, 2010.

[76] Iftach Haitner. Semi-honest to malicious oblivious transfer - the black-box way. In *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008*, pages 412–426, 2008.

[77] Danny Harnik, Yuval Ishai, and Eyal Kushilevitz. How many oblivious transfers are needed for secure multiparty computation? In *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, pages 284–302, 2007.

[78] Danny Harnik, Yuval Ishai, Eyal Kushilevitz, and Jesper Buus Nielsen. Ot-combiners via secure computation. In *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008*, pages 393–411, 2008.

[79] Danny Harnik, Joe Kilian, Moni Naor, Omer Reingold, and Alon Rosen. On robust combiners for oblivious transfer and other primitives. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, pages 96–113, 2005.

[80] Martin Hirt, Christoph Lucas, and Ueli Maurer. A dynamic tradeoff between active and passive corruptions in secure multi-party computation. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 203–219, 2013.

[81] Yan Huang, Jonathan Katz, and David Evans. Efficient secure two-party computation using symmetric cut-and-choose. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 18–35, 2013.

[82] Yan Huang, Jonathan Katz, Vladimir Kolesnikov, Ranjit Kumaresan, and Alex J. Malozemoff. Amortizing garbled circuits. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, pages 458–475, 2014.

[83] Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 44–61, 1989.

[84] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, pages 145–161, 2003.

[85] Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. On combining privacy with guaranteed output delivery in secure multiparty computation. In *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, pages 483–500, 2006.

[86] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer - efficiently. In *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, pages 572–591, 2008.

[87] Siddhartha Jayanti, Srinivasan Raghuraman, and Nikhil Vyas. Efficient constructions for almost-everywhere secure computation. In *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*, pages 159–183, 2020.

[88] Jonathan Katz. On achieving the "best of both worlds" in secure multiparty computation. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, pages 11–20, 2007.

[89] Joe Kilian. Founding cryptography on oblivious transfer. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 20–31, 1988.

[90] Valerie King and Jared Saia. From almost everywhere to everywhere: Byzantine agreement with $\tilde{o}(n^{3/2})$ bits. In *Distributed Computing, 23rd International Symposium, DISC 2009, Elche, Spain, September 23-25, 2009. Proceedings*, pages 464–478, 2009.

[91] Valerie King and Jared Saia. Breaking the $O(n^2)$ bit barrier: scalable byzantine agreement with an adaptive adversary. In *Proceedings of the 29th Annual ACM Symposium on Principles of Distributed Computing, PODC 2010, Zurich, Switzerland, July 25-28, 2010*, pages 420–429, 2010.

[92] Valerie King, Jared Saia, Vishal Sanwalani, and Erik Vee. Scalable leader election. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, Miami, Florida, USA, January 22-26, 2006*, pages 990–999, 2006.

[93] Valerie King, Jared Saia, Vishal Sanwalani, and Erik Vee. Towards secure and scalable computation in peer-to-peer networks. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, pages 87–98, 2006.

[94] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, pages 486–498, 2008.

[95] Tamás. Kővári, Vera T. Sós, and Pál Turán. On a problem of k. zarankiewicz. *Colloquium Mathematicae*, 3(1):50–57, 1954.

[96] Ranjit Kumaresan, Srinivasan Raghuraman, and Adam Sealfon. Network oblivious transfer. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, pages 366–396, 2016.

[97] Ranjit Kumaresan, Srinivasan Raghuraman, and Adam Sealfon. Synchronizable fair exchange. *IACR Cryptology ePrint Archive*, 2020.

[98] Ranjit Kumaresan, Vinod Vaikuntanathan, and Prashant Nalini Vasudevan. Improvements to secure computation with penalties. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 406–417, 2016.

[99] Eyal Kushilevitz. Privacy and communication complexity. *SIAM J. Discrete Math.*, 5(2):273–284, 1992.

[100] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[101] Enrique Larraia, Emmanuela Orsini, and Nigel P. Smart. Dishonest majority multi-party computation for binary circuits. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, pages 495–512, 2014.

[102] Matt Lepinski, Silvio Micali, Chris Peikert, and Abhi Shelat. Completely fair SFE and coalition-safe cheap talk. In *Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing, PODC 2004, St. John's, Newfoundland, Canada, July 25-28, 2004*, pages 1–10, 2004.

[103] Yehuda Lindell. Fast cut-and-choose based protocols for malicious and covert adversaries. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 1–17, 2013.

[104] Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. *J. Cryptology*, 28(2):312–350, 2015.

[105] Yehuda Lindell, Benny Pinkas, Nigel P. Smart, and Avishay Yanai. Efficient constant round multi-party computation combining BMR and SPDZ. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 319–338, 2015.

[106] Yehuda Lindell and Ben Riva. Cut-and-choose yao-based secure computation in the online/offline and batch settings. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, pages 476–494, 2014.

[107] Alexander Lubotzky, Ralph Phillips, and Peter Sarnak. Explicit expanders and the ramanujan conjectures. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 240–246, 1986.

[108] Hemanta K. Maji, Manoj Prabhakaran, and Mike Rosulek. A zero-one law for cryptographic complexity with respect to computational UC security. In *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, pages 595–612, 2010.

[109] Remo Meier, Bartosz Przydatek, and Jürg Wullschleger. Robuster combiners for oblivious transfer. In *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, pages 404–418, 2007.

[110] Payman Mohassel and Ben Riva. Garbled circuits checking garbled circuits: More efficient and secure two-party computation. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 36–53, 2013.

[111] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA*, pages 448–457, 2001.

[112] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, pages 111–126, 2002.

[113] Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 681–700, 2012.

[114] Rafael Pass, Elaine Shi, and Florian Tramèr. Formal abstractions for attested execution secure processors. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I*, pages 260–289, 2017.

[115] Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980.

[116] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, pages 554–571, 2008.

[117] Benny Pinkas. Fair secure two-party computation. In *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications*

of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings, pages 87–105, 2003.

[118] Michael O. Rabin. How to exchange secrets with oblivious transfer. *IACR Cryptology ePrint Archive*, 2005:187, 2005.

[119] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 73–85, 1989.

[120] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 387–394, 1990.

[121] Rohit Sinha, Sivanarayana Gaddam, and Ranjit Kumaresan. Luciditee: Policy-based fair computing at scale. *IACR Cryptology ePrint Archive*, 2019:178, 2019.

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

[123] Stefan Wolf and Jürg Wullschleger. Oblivious transfer is symmetric. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, pages 222–232, 2006.

[124] Jürg Wullschleger. Oblivious-transfer amplification. In *Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings*, pages 555–572, 2007.

[125] Andrew Chi-Chih 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, 1982.

[126] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 162–167, 1986.

[127] Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, pages 220–250, 2015.

[128] Mahdi Zamani, Mahnush Movahedi, and Jared Saia. Millions of millionaires: Multiparty computation in large networks. *IACR Cryptology ePrint Archive*, 2014:149, 2014.