Secure computation on incomplete networks

SHAILESH VAYA[†*]

Abstract

Secure multiparty computation of a multivariate function is a central problem in cryptography. It is known that secure multiparty computation can be realized by a set of n parties iff the connectivity of the underlying (authenticated) communication network is more than twice the number of corrupted parties. This impossibility result makes secure multiparty computation far less applicable in practice, as most deployed networks have a much lower degree than $O(n)$ and ideally one would like to tolerate $\theta(n)$ corrupted parties.

This work proposes a new model for secure multiparty computation for settings where authenticated channels are not assumed to be available between every pair of parties, and infact may be available between very few pairs of parties (i.e., networks of low degrees). For such settings, it is clear that not all honest parties can achieve traditional security guarantees of MPC. Such honest parties which neither receive their correct outputs, nor maintain the privacy of their inputs are called *sacrificed* parties. The new formulation of MPC, which allows some honest parties to be "sacrificed", in the manner described above, is called almost everywhere secure computation.

We show how to adapt standard protocols for unconditional secure MPC, that assume authentication channels between all pairs of parties, so that they can execute on incomplete networks, with special properties. Instrumental to our adaptation is a protocol for establishing secure channels between distant nodes of an incomplete network, using some infrastructure support from the incomplete network. The challange of designing such a multiparty protocol, can be abstracted as a two party secret key agreement problem using public broadcast channel.

Key words: Secure multiparty computation, Unconditional secure multiparty computation, Byzantine Agreement, almost everywhere agreement, almost everywhere secure computation, expander graphs, bounded degree networks, privacy amplification, secret key agreement.

[*]Department of Computer Science and Engineering, Indian Institute of Technology, Madras, Chennai, India - 600036. E-mail: vaya@cse.iitm.ernet.in. Most of the research in this work was done while the author was a graduate student at UCLA.

1 Introduction

In secure multiparty computation n players jointly compute the value of an arbitrary n-variate polynomial time computable function on their corresponding inputs. The strong guarantee for secure MPC is that even if a fraction of the nodes are controlled by a malicious adversary, all the honest parties should obtain their correct outputs, without leaking any more information about their individual inputs, than revealed by the outputs received by the corresponding parties. Solutions for this problem were given by [Yao82] for two parties, followed by [GMW87] for n parties.

In [BGW88] and [CCD88], it has been shown how to achieve secure multiparty computation in the information theoretic regime, as long as as least $\lfloor \frac{n}{3} \rfloor + 1$ of the participating parties are guaranteed to be honest. This result assume the existence of point to point authentication channel between every pair of parties. It is natural to wonder how realistic is this assumption? Since *reliable*, let alone *private* channels are too expensive to assume, networks with direct and private channels between every pair of parties are hard to realize in practical setting. What is the possibility of achieving secure multiparty computation on such settings?

The above question was first posed by [Dol83] and further explored by [DDWY93], who showed that if there can be at most t corrupted players, then connectivity of $2 * t + 1$ is both necessary and sufficient to achieve information theoretically secure multiparty computation. Since the number of corrupted parties can be a constant fraction of the total number of parties, to reflect many practical settings, the connectivity (by way of authentication channels) of the underlying network demanded to achieve general secure MPC should also be $O(n)$.

Since the results of [DDWY93] rule out achieving secure multiparty computation or even agreement for networks of low degree, [DPPU88], pose a weaker question on agreement: *An honest party could be surrounded by a set of parties that are all controlled by the adversary. Clearly, this honest party cannot even communicate reliably with the rest of the honest parties, so even achieving byzantine agreement on such networks is not a possibility. Can we still do anything meaningful? What about the possibility of most of the honest parties agreeing on one value?*. [DPPU88] proposes a notion, which allows some of the honest parties to not reach the agreement value. This type of agreement is called *almost everywhere secure agreement*. [DPPU88] makes the following type of statements: $g(n,t)$-agreement protocol is one whose execution results in all, but $g(n,t)$ of the honest parties reaching the agreement, for some function $g(.,.)$, when at most t parties are corrupted.

In this work we continue this important line of research. We propose a formulation of secure multiparty computation, relevant to the setting of incomplete networks, which allows a subset of the honest parties to not achieve the guarantees of secure multiparty computation. Such honest parties are called *sacrificed* parties. In this new formulation of secure multiparty computation, called almost everywhere secure multiparty computation, the sacrificed parties are not required to preserve the privacy of their inputs or to receive their correct outputs, however, all the other honest parties should still be able to achieve the standard guarantees of secure multiparty computation.

We propose definitions for realizing secure multiparty computation, on incomplete networks, that possess special properties, and show how to realize these definitions on several different families of networks. Section 2 is devoted to the discussion of results in this work. Here we highlight the contributions briefly:

Contributions of this work:

1. We propose a new stand-alone model for the problem of privacy amplification using public broadcast channel. We argue the inapplicability of the previous techniques developed for this problem and propose a solution for this problem in the new setting.

2. We show how to realize unconditionally secure multiparty computation on partially connected networks. For this, we formally define several new notions, terminologies and concepts.

   We define what it means for a multiparty protocol to execute on a partially connected network. We propose corruption of parties as well as channels by modeling channels as entities, just like parties. While a corrupted party can behave arbitrarily, as is allowed traditionally, a corrupted channel is allowed limited deviations in the behavior. While at one end these deviations capture a large class of adversarial behaviors, at the other, these deviations are not (necessarily) part of the final definition and do not affect realization of secure multiparty computation on arbitrary partially connected networks, with special properties.

   We define general adversary structures $\mathcal{T}$ which capture the limitations of an adversary: Namely, to choose to corrupt only from a given quadruplets of subsets of parties and channels, actively as well as passively. We define $\mathcal{T}$-authentic and $\mathcal{T}$-secure channels between a pair of parties p_u and p_v of the network N. A multiparty protocol realizes $\mathcal{T}$-secure channel between p_u and p_v if for all $\vec{C} \in \mathcal{T}$ corrupted by the adversary, the multiparty protocol establishes a secure channel between p_u and p_v.

   We propose a definition of unconditionally secure multiparty computation in the stand alone model. Then we propose a definition along the same lines, for a multiparty protocol to $\mathcal{T}$-securely evaluate function f on a partially connected network, which captures the notion of sacrificed honest parties as described above.

   Finally, we show how to adapt any multiparty protocol that realizes the first definition to a multiparty protocol that realizes the definition for partially connected networks and show how to achieve unconditionally secure multiparty computation on partially connected networks with special communicability properties.

The intuition behind our terminologies, definitions and proofs has been discussed in the Section 2, while all definitions and proofs have been relegated to the appended appendix.

Comparison with results in [Vay06]: Following are the salient differences in this work with respect to [Vay06]:

1. The definitions presented here are modular. This has been achieved by abstracting important conceptual elements from the definition, defining them separately and referring to them in definition under consideration. Secondly, all notations are defined in the Preliminaries and followed consistently throughout the work. This has simplified the presentation.

2. In [Vay06], we present a definition for unconditionally secure multiparty computation in which the input commitment phase is necessitated to be via *verifiable secret sharing*. We find that this may be a little restrictive. Here, we present a slightly different definition by associating a function $reveal()$ with the input commitment phase of the multiparty protocol and characterizing this function. Our

characterization of correctness of the input commitment phase, in this way is sufficient for us to refer to the vector of input values committed to by the parties, on which the computation is done.

3. We model corruption of channels by associating with them identities like parties, similar to the notion of intermediate nodes/parties in [Vay06]. We describe the functionality of these channels for passive/active/no corruption. We introduce generic adversary structures to model passive and active corruption of parties and channels throughout the presentation. We also introduce the notion of *real* and *virtual* edges/channels. Behaviorally, these channels are same except that when a message is sent by a sender along these channels, it takes *different number of rounds* to reach the receiver.

4. In [Vay06], we define a notion of executability of a multiparty computation Π on a partially connected network $\mathcal{N}$, by making it users responsibility to verify that a party does not send a message along a channel that does not exist in the partially connected network in any execution. Here, we relinquish this concern to implementation details and say that if a message to be sent to a given party p_i is put in the out-buffer of a party p_j, to be sent through channel (p_j, p_i), while no such channel exists, then the message is simply dropped.

To sum it up, the results in this work and [Vay06] are statistically indistinguishable, the negligible gap being due to the difference in the presentation (which only a computationally unbounded adversor could have found). A more elaborate comparison with [GO07] would be drawn once the manuscript becomes available to us.

1.1 Related works

A more elaborate discussion of related literature is provided in [Vay06] (and the extended abstract). Here we discuss works that are closely related.

The problem of privacy amplification via public discussion has been studied extensively in [BBR88], [BBCM95]. Unconditional secure key agreement in this setting was introduced by Bennett, Brassard and Robert in [BBR88] and generalized by Ahlswede and Csiszar, [RA93] and [Mau93], who introduced the general information theoretic model. In [BBCM95], [BS94], [CM94], it has been shown that the assumption, that Eve's channel ought to be noisier than Bob's, can be relaxed if a public discussion channel is available to the parties. In all these works the parties are provided access to mutually correlated random data and a public discussion channel. Most recently, [DS04], give results on the problem of correcting errors in a jointly shared secret string without leaking partial information, with very strong guarantees using strong randomness extractors.

The problem of *perfect secure message transmission* over partially connected networks was introduced by Dolev, Dwork, Waarts and Yung in their classic work, [DDWY93]. In [DDWY93] tight bounds are established on the necessary and sufficient conditions on the connectivity of the communication network for perfectly secure communication channels to exist between every pair of nodes, for directed and undirected networks. In [DW98], the authors consider the general case when certain number of paths are directed in one direction and another number of directed paths are directed in another direction. Further improvements/Trade-offs on message size and round complexity of protocols, of [DDWY93], have been studied by [SAA96] and [SNR04]. In [OR96], authors allow a negligible probability of failure in transmission of the message. In [GGL91], [FY95] and [FW98], initiate the study of secure communication

and secure computation in the multicast environment in information theoretic model, for passive as well as active adversaries. Further results are given on this problem in [WD00].

Secure computation on partially connected networks has been discussed at some length in the introduction. We mention a few other works that realize almost everywhere secure agreement on partially connected networks, as defined by [DPPU88]. [BG89], [BG90] improve the efficiency of the protocols for almost everywhere agreement presented in [DPPU88]. In [Upf92], it is shown how to realize almost everywhere agreement on constant degree networks, namely LPS expanders, tolerating a linear number of faults. The round complexity of the protocol of [Upf92] is polynomial, though the algorithm to be executed by each party runs in super-polynomial time. In [OR96], networks of bounded degree and a fully polynomial almost everywhere agreement protocol which tolerates, randomly located faulty processors, where processors fail independently with constant probability, with high probability are presented.

Assuming that more than $\frac{2}{3}$ parties are honest, it has been shown that it is possible to securely compute any functionality, [BGW88], [CCD88], [RBO89] in the information theoretic regime. In the computational model, the results have been given in [GMW87], [Yao82]. Adversary structures, extensively used in this work, were studied in [FHM99], [HM97], [HM00], [HMP00].

2 Discussion of the results

In this section we shall provide the intuition behind the new notations and definitions proposed in this work. Subsequently, we shall describe how these definitions are realized in the new setting and provide intuition behind the proofs of security. Full details have been relegated to the appendix.

All our results are for the stand alone model in the synchronous regime. Our protocols achieve statistical correctness only.

2.1 Some new terminologies and definitions

We model corruption of parties as well as channels. For this we model *channels as entities* just like parties and provide a behavioral description of the channels, when they are secure or authentic (passively corrupted) or tamperable (actively corrupted). These descriptions limit how the behavior of the channels can deviate under adversarial corruption but is sufficient for us to realize our ultimate goals for partially connected networks i.e., is necessary for us to keep our proofs modular but only vestigial in the ultimate scheme. Having, said that, our final definitions of secure computation on partially connected networks, do allow such corruptions.

We model different types of corruptions by defining a *generic adversary structure* $\mathcal{T}$, which is a set of quadruplets of subsets of the following types: subsets of parties corrupted passively, subsets of parties corrupted actively, subsets of channels corrupted passively and subsets of channels corrupted actively. Such an adversary is called a $\mathcal{T}$-limited adversary. By a multiparty protocol that $\mathcal{T}$-securely computes some value is essentially qualifying the adversary which it can resist.

2.1.1 Secure communication on partially connected networks

We say that a multiparty protocol, executed on network N, realizes a $\mathcal{T}$-authentic channel between two parties if for all $\vec{C} \in \mathcal{T}$ corrupted by the adversary, the two parties can exchange messages authentically by the multiparty execution of such a protocol on partially connected network N. Similarly, we say that a multiparty protocol realizes a $\mathcal{T}$-secure channel between two parties if for all $\vec{C} \in \mathcal{T}$ corrupted by the adversary, the two parties can exchange message securely by the multiparty execution of such a protocol on partially connected network N.

2.1.2 Secure computation on partially connected networks

We first propose a definition of unconditionally secure multiparty computation in the stand alone setting (in synchronous regime). For this we define two phase computation as follows: In first phase, parties commit to their input values and, in the second phase parties compute the function f on the committed input values. We characterize the correctness of the input commitment phase by associating a function $reveal()$ with the first phase. The function $reveal()$ takes as input the transcripts of the parties generated so far. It should reveal the input values committed to by all the parties, as long as the transcripts of the honest parties are genuine, and irrespective of what transcripts are substituted for the corrupted parties. Furthermore, the honest parties are able to commit to their original input values. Note, that this characterization is used only for defining the correctness of computation. The correctness of computation is defined by requiring the distribution of output values to be indistinguishable from the distribution of evaluations of function f for honest parties. For the privacy property, it is enough to demonstrate a simulator that takes as input the input and out values of the corrupted parties and simulates their view so that the resulting distribution is indistinguishable from the one generated in real execution of the multiparty protocol. This definition is called **Traditional MPC**.

Just a slightly different characterization is required for the input commitment phase of secure computation on partially connected networks, where some honest nodes are sacrificed in the following sense: The sacrificed nodes may not be able to commit to their initial input values, may not preserve the privacy of their input values and may not obtain their correct output values. The characterization of the input commitment phase for this definition is adapted along the same lines. In particular the characterization ensures correctness of commitment and computation are guaranteed only for the non-sacrificed honest parties, which should be greater than $\lfloor \frac{2*n}{3} \rfloor + 1$ in number. For the privacy condition, the simulator is provided with the input and output values of both the corrupted and sacrificed parties. This bounds the amount of knowledge leaked by the computation process, which appropriately models the privacy requirements of secure computation on partially connected networks. Lastly, only statistical correctness of commitment and computation is ensured. This definition is referred to as $\mathcal{T}$-secure evaluation on partially connected network N, where $\mathcal{T}$ is an adversary structure.

2.2 Secure key agreement with public broadcast channel

The problem description is as follows: N channels are given between two parties, at least one of which is known to be secure, but it not known which one of these channels is secure. The two parties also share a public broadcast channel. The two parties need to agree upon a secret random string using this

infrastructure. We give a simple protocol to achieve this goal: Alice sends blocks of uniformly chosen random bits over each of the N channels and reveals the entire blocks to Bob, except for hiding one bit, using the public broadcast channel. Any faulty channel is caught with high probability. Since, at least one of the N channels is secure, one bit is completely hidden from the adversary. The message bit is sent by XOR'ing it with bits chosen from blocks found to be uncorrupted, and sending the resulting bit to Bob via the public broadcast channel. Ofcourse, a bad channel may not necessarily be located this way, but we are able to securely transmist a bit with high probability and achieve statistically correct transmission channel between Alice and Bob. Further, optimizations can be done to achieve better message complexity by using error correcting codes, but that is not the focus of the current work.

2.3 (almost) secure channels on partially connected networks, with special properties

A network is said to have a special property $\mathcal{T}_{p_u,p_v,\beta,c}$ if for every choice of quadruplet $\vec{C} \in \mathcal{T}$ corrupted by $\mathcal{A}$, there exists a path of secure channels connecting uncorrupted notes of length at most $c * lg\_2n$ and multiparty protocol $\beta(p_u, p_v, N, k, l)$ realizes $\mathcal{T}$-authentic channel between p_u and p_v. It can be seen that if a partially connected network N possesses such a property, then one can abstract the infrastructure of bit transmission protocol, as discussed above. [Note, that we shall be using this special property only for the case when there is no (directed/undirected) edge (p_u, p_v) in the partially connected network.]

Thus, using this infrastructure we describe a multiparty protocol, $\gamma(p_u, p_v, N, k, l)$ to be executed on a partially connected network N, to establish $\mathcal{T}$-secure channel (which achieves statistical correctness) from p_u to p_v, along the same lines as the bit transmission protocol.

The proof of privacy is also along the same lines as the bit transmission protocol. Here, we demonstrate a simulator irrespective of whether $\vec{C} \in \mathcal{T}$ or $\vec{C} \notin \mathcal{T}$. When $\vec{C} \notin \mathcal{T}$, the channel established may be just authentic or even tamperable. The simulator would be provided with the message being transmitted, depending on the particular case/sub-case. The simulator for these other cases (i.e. when the channel established is authentic or tamperable), is needed for technical reasons, to carry out higher levels proof of security, namely that of multiparty computation on partially connected networks.

2.4 Realizing secure multiparty computation on partially connected networks

We start by assuming that there exists a multiparty protocol that satisfies the traditional definition for unconditionally secure multiparty computation. Then, we extend the result to unconditionally secure multiparty computation with corruption of parties as well as channels. The extension is straightforward and just requires to demonstrate an appropriate adversary attacking the former execution, given an adversary attacking the multiparty computation execution for the latter case, for both of which the same subset of honest nodes H of size greater than $\lfloor \frac{2*n}{3} \rfloor + 1$ achieve the multiparty computation results.

For $\mathcal{T}$-secure computation on a partially connected network N, we require that network N possesses a certain $\mathcal{T}$-communicability property, which ensures that a large fraction of honest node ($\geq \lfloor \frac{2*n}{3} \rfloor + 1$) are able to securely communicate with each other, irrespective what quadruplet is corrupted by the adversary from adversary structure $\mathcal{T}$.

At the high level, construction of a multiparty protocol that $\mathcal{T}$-securely evaluates function f, on partially connected network N goes as follows:(1) We add (directed) virtual edges between every pair of nodes for which there does not exist any edge in the partially connected network N. This network is called C_N and it provides a straightforward infrastructure for all pairs of parties to communicate with each other. The *virtual* and *real* channels are distinguished in the following sense: It takes α rounds to send a message along a real channel, while it takes $f_\gamma(\alpha, N)$ rounds for a message sent along the virtual channel to reach the other party, for some function f_γ. Along the same lines we construct an enhanced adversary structure $\mathcal{T}_{C_N}$ from $\mathcal{T}$, by adding (passive/active) corruption of virtual edges, depending on the choice of quadruplet $\vec{C}$ corrupted by the adversary $\mathcal{A}$, the family of multiparty protocols $\gamma(p_u, p_v, N, k, c)$ used for establishing secure channels on the network N. Now, we adjust the multiparty protocol for the traditional case, so that each step of it is essentially elongated to $f_\gamma(\alpha, N)$ rounds. All the messages sent in a given step are received - either via the real channel in α rounds or through the virtual channel in $f_\gamma(\alpha, N)$ rounds, but before the execution of the next step of the multiparty protocol commences.

The inductive hypothesis starts with a triplet intermediate network N_i, adversary structure $\mathcal{T}_{N_i}$, and multiparty protocol Π_{N_i} and constructs the new triplet by removing a virtual channel from the network N_i and replacing it with a virtual channel by a channel simulated by some multiparty protocol γ (existence of which follows from the communicability property of the network). It takes at most $f_\gamma(\alpha, N)$ rounds for the parties to send a message by executing the $\mathcal{T}$-secure channel. It can be verified that the *correctness* and *privacy* property of the multiparty protocol are satisfied for the corresponding subset of honest nodes H_i at each step.

The I. H. is repeated till all virtual channels have been replaced by simulated channels. Thus, we have a multiparty protocol that $\mathcal{T}$-securely evaluates function f on the partially connected network N for adversary structure $\mathcal{T}$ discussed above.

The adversary structure $\mathcal{T}$ can be limited to corrupt only t parties maliciously and we can then use the result by observing that communicability property for the resulting adversary structure holds for appropriate classes of networks.

Thus, we achieve t-secure multiparty computation on families of incomplete networks, as considered in [Upf92], [DPPU88].

Full details of notions, definitions and proofs of security are appended in appendix.

3 Discussion about simulation based definition versus KKMO type input indistinguishability

I quote from [**?**] the objection raised by Garay and Ostrovsky about simulation based definition:
"Besides several other issues that would require technical improvement, a salient difference (shortcoming) in [Vay06] is that the simulation based security definition, [**?**], where the ideal world adversary (the simulator), besides having access to the inputs of the corrupted parties also has access to the inputs of the corrupted players that are given up; such a strong assumption makes the proof security relatively straightforward and gives simulator an unfair advantage compared to the real world execution."

A second paragraph from Section 5 is as follows: "However, in the setting of almost everywhere secure computation, the simulation based approach encounters the following problem: it is not clear how to define in a meaningful and network independent way, the simulation and adversarial view of the doomed players or indeed how to deal with this in a dynamically growing way. It is clear that these nodes are not part of the nodes for which we guarantee a correct output, but it is not clear what view of these nodes an adversary gets. Indeed, for some of the doomed honest nodes the adversary could learn all the information and be able to change their inputs, while for others the adversary would only get partial control."

From the above I gather that the objection of Garay and Ostrovsky in essence is about some *assumption*, namely that where from is the simulator made available the inputs and outputs of sacrificed nodes. It seems that Garay and Ostrovsky raise these (kinds of other) objections because they are seeing simulator and the entire set up of providing the inputs and outputs of corrupted and sacrified parties as something that should "be achievable" in real life. However, demonstrating a *simulator* and the associated set up is only a way to prove a bound on the amount of knowledge leaked in a protocol execution. Its subtle: There is no assumption being made here - just a way of "arguing" that the adversary never gains any more knowledge from the execution of the protocol, then the inputs and outputs of the sacrificed and corrupted parties - said another way one could at best have reconstructed the input and output values of these parties from the view of the adversary. I think Juan and Rafi are struggling to understand, [**?**], that simulator is just an *abstract mental construct* used to prove some properties of the MPC protocol and hence state that *simulator is given an unfair advantage over the real world*. Infact, everything is just fine even if the adversary does not even possess or is aware of the existence of such a simulator![1]

Acknowledgments: I'd like to thank Rafail for providing me with financial support for a year, during which a part of this work was done. I had several useful discussions related to this work with Rafi and Juan. In particular, my thesis adversor, Rafi, called to our attention the application of cut-and-choose technique, [Wik00], used in the bit transmission protocol. I also had a few discussions with Rafi and Juan, on whether to use [KKMO94] style indistinguishability based definition of security, used in [GO07], or simulation based definition of security used in this work, [Vay06]. I am grateful to Chandrashekhar Pandurangan for invaluable advice about the presentation of this submission. Finally, I would like to express my gratitude towards Prof. P. L. Dhar without whose moral support and encouragement this work would not be possible.

Remarks on Acknowledgements, in [Vay06]: Some comments have been made in [**?**] about the accuracy of acknowledgements made in [Vay06]. For this I would like to bring to light the history of this work:

Some previous versions and intermediate copies of [Vay06] (available with me) which had UC based definitions contain the same attributions/acknowledgements which have remain unchanged in all these copies. After UC formulations were ruled out, we had a few discussions about the choice of simulation paradigm versus KKMO criterion for the stand alone model and went separate ways according our choices [In that sense I have equal contributions to the definitions of [**?**] as Rafail and Juan to [Vay06], which they have also forgotten to acredit me]. Logs of the emails exchanged with Rafail and Juan and intermediate documents are available with me (all exchanges took place through email while I was in India) and I

[1] Hint: How real is *reality*? Does *reality* really exist? I do not know - may be it is all happening in the mind.

would be happy to share these with any interested third party, to estimate these contributions. In any case, from the recent doubts raised by [**?**], that "it is unfair to provide the simulator inputs/outputs of sacrificed parties", it is clear that Juan and Rafi do not understand that demonstrating a simulator is a way of bounding the amount of relevant knowledge (see above) that the adversary can obtain from its view (which can be reconstructed from this relevant knowledge in polynomial time) and goes to show the extent to which they contributed to the "central" "ideas" in [Vay06].

May be I forgot and was not courteous enough to mention how many months Rafi spent formatting, [Vay06], before filing it with UCLA library - that to me was Rafi's most important contribution to [Vay06] and I do feel guilty to not have mentioned this in the previous version.

References

[BBCM95] C.H. Bennett, G. Brassard, C. Crepeau, and U. M. Maurer. Generalized privacy amplification. *IEEE Transactions on Information Theory*, 41, November 1995.

[BBR88] C. H. Bennett, G. Brassard, and J. M. Robert. Privacy amplification by public discussion. *Siam Journal of Computing*, 17(2), April 1988.

[BG89] P. Berman and J. Garay. Asymptotically optimal distributed consensus. In *International Colloquium in Automaton, Languages and Programming, ICALP*, pages 80–94. Springer Verlag, 1989.

[BG90] P. Berman and J. Garay. Fast consensus on networks of bounded degree. In *International Workshop on Distributed Algorithms*, pages 321–333, 1990.

[BGW88] M. BenOr, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of Twentieth annual Symposium of Theory of Computation, STOC*, Chicago, Illinois, May 1988. Association for Computing Machinery.

[BS94] G. Brassard and L. Salvail. Secret key reconcilation by public discussion. In *Advances in Cryptology, EUROCRYPT*, Lecture Notes in Computer Science, pages 410–423. Springer Verlag, 1994.

[Can05] R. Canetti. Universal composable security: A new paradigm of cryptptographic protocols. *Eprint cryptographic archives, http://eprint.iacr.org/2000/067*, January 2005.

[CCD88] D. Chaum, C. Crepeau, and I. Damgard. Multiparty unconditionally secure protocols. In *Proceedings 20th Annual Symposium on Theory of Computing, STOC*, Chicago, Illinois, May 1988. Association for Computing Machinery.

[CM94] C. Cachin and U. Maurer. Linking information reconcilation and privacy amplification. In *Advances in Cryptology, EUROCRYPT*, May 1994.

[DDWY93] D. Dolev, C. Dwork, O. Waarts, and M. Young. Perfect secure message transmission. *Journal of ACM, JACM*, 1993.

[Dol83] D. Dolev. The byzantine generals revisited. *Journal of Algorithms*, 1(3), 1983.

[DPPU88] C. Dwork, D. Peleg, N. Pippinger, and E. Upfal. Fault tolerance in networks of bounded degree. *SIAM Journal on Computing*, 1988.

[DS04] Y. Dodis and A. Smith. Correcting errors without leaking partial information. In *Foundations of Computer Science, FOCS*. IEEE, November 2004.

[DW98] Yvo Desmedt and Yongge Wang. Perfectly secure message transmission revisited. In *Advances in Cryptology, EUROCRYPT*, Lecture Notes in Computer Science, ESP00, FINLAND, June 1998. Springer Verlag.

[FHM99] M. Fitzi, M. Hirt, and U. Maurer. General adversaries in unconditional multi-party computation. In *Advances in Cryptology, ASIACRYPT'99*, Lecture Notes in Computer Science, Singapore, November 1999.

[FW98] M. K. Franklin and R. N. Wright. Secure communications in minimal connectivity models. In *Advances in Cryptology, EUROCRYPT*, Lecture Notes in Computer Science, pages 346–360, ESPOO, FINLAND, June 1998. Springer Verlag.

[FY95] M. Franklin and M. Yung. Secure hypergraphs: privacy from partial broadcast. In *Symposium of Theory of Computation, STOC*, Las Vegas, NV, USA, June 1995. Association for Computing Machinery.

[GGL91] O. Goldreich, S. Goldwasser, and N. Linial. Fault-tolerant computation in the full information model (extended abstract). In *Proceedings of Thirty Second Symposium on Foundations of Computer Science*, pages 447–457, 1991.

[GM82] S. Goldwasser and S. Micali. Semantic security. In *Foundations of Computer Science, FOCS*. IEEE, 1982.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proceedings of nineteenth annual Symposium of Theory of Computation, STOC*. ACM, May 1987.

[GO07] J. Garay and R. Ostrovsky. Almost everywhere secure computation. In *manuscript in preparation*, 2007.

[HM97] M. Hirt and U. Maurer. Complete characterization of adversaries tolerable in secure multi-party computation. In *ACM Proceedings of Sixteenth Annual Symposium in principles of Distributed Computing*, 1997.

[HM00] M. Hirt and U. Maurer. Player simulation and general adversary. *Structures in Perfect Multiparty Computation, Journal of Cryptology*, 2000.

[HMP00] M. Hirt, U. Maurer, and B. Przydatek. Efficient secure multi-party computation. In *Advances in Cryptology, ASIACRYPT*, Lecture Notes in Computer Science, 2000.

[KKMO94] J. Kilian, E. Kushilevitz, S. Micali, and R. Ostrovsky. Reducibility and completeness in multi-party private computations. In *Foundations of Computer Science, FOCS*, 1994.

[Mau93] U. Maurer. Secret key agreement by public discussion from common information. In *IEEE Transaction on Information Theory*, volume 39, May 1993.

[OR96] M. Ben Or and D. Ron. Agreement in the presence of faults, on networks of bounded degree. *Information Processing Letters, IPL*, 57(6):329–334, March 1996.

[RA93] I. Csiszar R. Ahlswede. Common randomness in information theory and cryptography- part i: secret sharing. *IEEE Transaction on information theory*, 39:1131–1132, 1993.

[RBO89] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocol with honest majority. In *ACM Symposium in Theory of Computing, STOC*, 1989.

[SAA96] Hasan Md. Sayeed and H. Abu-Amara. Effecient perfectly secure message transmission in synchronous networks. *Information and Computation*, 1(126):53–61, 1996.

[SNR04] K. Srinathan, A. Narayanan, and C. Pandu Rangan. Optimally perfect secure message transmission. In *Advances in cryptography, CRYPTO*, Lecture Notes in Computer Science, Santa Barbara, California, USA, August 2004. Springer Verlag.

[Upf92] E. Upfal. Tolerating linear number of faults in networks of bounded degree. In *Symposium on Principles of distributed computing, PODC*, Vancouver, British Columbia, Canada, 1992. ACM.

[Vay06] S. Vaya. *Almost everywhere secure computation*. PhD thesis, University of California Los Angeles, Los Angeles, California, USA, December 2006.

[WD00] Y. Wang and Y. Desmedt. Secure communication in multicast channels: The answer to franklin and wright's question. *Journal of Cryptology*, 2000.

[Wik00] Wikipedia. http://www.everything2.com/index.pl?node=blindIn *Web Encylopaedia*, 2000.

[Yao82] A. Yao. Protocols for secure computation. In *Proceedings of twenty third annual Symposium on Foundations of Computer Science, FOCS*. IEEE, 1982.

A Model and Definitions

We will review some standard terminologies, notations and definitions and follow them up with the new terminologies and definitions.

A.1 Preliminaries.

A function $\delta : N \rightarrow [0,1]$ is called *negligible* if for all $c > 0$ and for all large enough $k \in N$ we have $\delta(k) < k^{-c}$.

A **distribution ensemble** $X = \{X(k,a)\}_{k \in N, a \in \{0,1\}^*}$ is an infinite set of probability distributions, where $X(k,a)$ is associated with each $k \in N$ and $a \in \{0,1\}^*$. A distribution ensemble is called binary if it consists only of distributions over $\{0,1\}$.

Definition A.1 *The statistical distance, $SD(Z_1, Z_2)$, between two distributions, Z_1 and Z_2 is defined as* $SD(Z_1, Z_2) = \frac{1}{2}\sum_a |Prob(Z_1 = a) - Prob(Z_2 = a)| < \delta(k)$.

Definition A.2 (Statistical indistinguishability) *Distribution ensembles X and Y have a* **statistical distance** *δ, if for all sufficiently large k and a, $SD(X(k,a), Y(k,a)) < \delta(k)$. If $\delta(.)$ is a negligible function, ensembles X and Y are called* **statistically indistinguishable** *and this is denoted as $X \approx Y$.*

$Power - Set(F)$ refers to the set of all subsets of a set F.

A mixed network N is referred to as a triplet (V, E, E_d), where V refers to a set of vertices, and E refers to the set of undirected edges and E_d refers to the set of directed edges. Whenever not specified, N will denote a mixed network. Note that $E \cap E_d$ may not be ϕ, but we shall always consider networks where $E \cap E_d = \phi$.

Power of the adversary We shall focus on static adversaries only, which corrupt a subset of parties before the execution of the protocol.

If adversary $\mathcal{A}$ corrupts a party *actively*, it gains complete control of it. In particular, the adversary gains complete control of the party, namely its input tape, its random tape, its program and is free to send arbitrary messages on the behalf of the party, while also receiving all the messages sent to the party by other parties.

If $\mathcal{A}$ corrupts a party *passively* it just obtains the privilege to receive all the inputs, outputs and messages exchanged by the party with other parties during the execution of the protocol.

Computational versus Information Theoretic setting We assume that the adversary is computationally unbounded.

Statistical versus Perfect Correctness The protocols in this work, achieve statistical correctness only. This allows the protocols to fail to achieve the correctness of their outputs with at most a negligible probability, $\mu(n)$, for some negligible function $\mu(.)$.

Adversary structures An adversary $\mathcal{A}$ is said to be *t-limited* if it can corrupt at most t honest parties.

We consider corruptions of parties as well as channels, passively as well as actively. For this we use the notion of *adversary structure*. Define adversary structure, $\mathcal{T} \subset \{(\mathcal{X}_p, \mathcal{X}_a, \mathcal{Y}_p, \mathcal{Y}_a) | \mathcal{X}_p, \mathcal{X}_a \subset V, \mathcal{Y}_p, \mathcal{Y}_a \subset V * V$, and $\mathcal{X}_p \bigcap \mathcal{X}_a = \phi, \mathcal{Y}_p \bigcap \mathcal{Y}_a = \phi\}$, to denote a set of quadruplets of of the following four types of subsets: $\mathcal{X}_p$, a subset of parties corrupted passively, $\mathcal{X}_a$, a subset of parties corrupted actively, $\mathcal{Y}_p$, a subset of channels corrupted passively and $\mathcal{Y}_a$, a subsets of channels corrupted actively by $\mathcal{A}$.

An adversary which is restricted to adversary structure, $\mathcal{T}$, is called $\mathcal{T}$-limited adversary or in short $\mathcal{T}$-adversary.

Let $\vec{C} \in \mathcal{T}$ be a quadruplet of corruptions by $\mathcal{A}$.

Definition A.3 *Let $P, \mathcal{A}, \vec{C}$ be defined as above. $\vec{C}$ is a feasible vector of corruptions, if there exists a subset of honest parties $H \subset P - \vec{C}[0] - \vec{C}[1]$, such that the following two conditions hold true:*

1. $|H| \geq \lfloor \frac{2n}{3} \rfloor + 1$
2. $\forall P_u, P_v \in H : (P_u, P_v) \notin \vec{C}[2] (P_u, P_v) \notin \vec{C}[3]$

A *feasible adversary structure* $\mathcal{T}$ is defined along the same lines.

Modeling a partially connected network of communication channels We recognize communication channels as entities, similar to parties which may be passively or actively corrupted by the adversary and behave differently, as a result.

Authentic or Secure channels could be established between a given pair of parties by cryptographic means, but could also be available due to physical infrastructure of the system. For example, an email originating from the domain of a deemed institute, is an authenticating message. Similarly, secure channels could also be realized by Fiber optic cables etc.

Let $N = (V, E, E_d)$ be a network, where V refers to the set of vertices. Each vertex models a party aka player. E refers to the set of undirected edges in the network, where each edge models a *bi-directional secure communication channel* between the corresponding pairs of vertices and E_d refers to the set of directed edges in the network, where each edge models a *directed secure communication channel*. We formalize such channels as follow:

Let r be a constant, greater than or equal to 3, which will be specified later. Let $\mathcal{A}$ corrupt $\vec{C}$, as described above, A.1. The behavior of entity $F^r(S, R, edge_{id})$ modeling a bi-directional channel between S and R for different types of corruptions is defined below:

Definition A.4 *Let $F^r(S, R, edge_{id})$, denote a bi-directional channel between parties S and R, with unique identity $edge_{id}$, in the synchronous setting as an entity which executes as follows:*

If ($edge_{id} \in \vec{C}[3]$) // Passive corruption 1. If message $(S, R, mesg - id, m)$ is received from party S in round i, $F^r(S, R, edge_{id})$ forwards message $(S, R, mesg - id, m)$ to $\mathcal{A}$ in round $i + 1$, and to party R in round $i + r$, else

2. If message $(R, S, mesg - id, m)$ is received from party R in round i, $F^r(S, R, edge_{id})$ forwards message $(R, S, mesg - id, m)$ to $\mathcal{A}$ in round $i + 1$, and to party S in round $i + r$. else if ($edge_{id} \in \vec{C}[3]$) // Active corruption 1. If $(S, R, mesg - id, m)$ is received from S in round i, $F^r(S, R, edge_{id})$ records the message (and round number etc.) and forwards message $(S, R, mesg - id, m)$ to $\mathcal{A}$ in round $i + 1$, else

2. If $(S, R, mesg - id, m')$ is received from $\mathcal{A}$, $F^r(S, R, edge_{id})$ checks (validity etc.) if received before (or same as) round $i + r - 1$ with previous records (and round numbers etc.) and forwards $(S, R, mesg - id, m')$ to R in round $i + r$, else

3. If $(R, S, mesg - id, m)$ is received from R in round i, $F^r(S, R, , edge_{id})$ records this message (and, round number etc.) and forwards message $(R, S, mesg - id, m)$ to $\mathcal{A}$ in round $i + 1$, else

4. If $(R, S, mesg - id, m')$ is received from $\mathcal{A}$, $F^r(R, S, edge_{id})$ checks (validity etc.) if received before round $i+r-1$ with previous records (and round numbers etc.) and forwards message $(R, S, mesg-id, m')$ to R in round $i + r$. else // Secure channel 1. If message $(S, R, mesg - id, m)$ is received from party S in round i, $F^r(S, R, edge_{id})$ forwards message $(S, R, mesg - id, \perp, |m|)$ to $\mathcal{A}$ in round $i + 1$, and message $(S, R, mesg - id, m)$ to party R in round $i + r$, else

2. If message $(R, S, mesg - id, m)$ is received from party R in round i, $F^r(S, R, edge_{id})$ forwards message $(R, S, mesg - id, \perp, |m|)$ to $\mathcal{A}$ in round $i + 1$, and message $(R, S, mesg - id, m)$ to party S in round $i + r$.

The entity $F^r(S, R, edge_{id})$ modeling a directed channel from S to R is defined along the same lines as above (where a message is considered by channel $edge_{id}$ only if received from party S and is to be sent to party R).

Remark A.5 1. In the proof of the main Theorem, we use networks that have two different types of channels: *real* and *virtual* channels. Messages sent along either of the channels may takes different number of rounds to reach the other party. The need for this distinction (and the exact number of rounds associated with each channel) is made more explicit at a later stage, Section D.5.

2. All *real* channels are undirected and all *virtual* channels are directed. With each channel is associated a unique identity, denoted by $edge_{id}$. If the set of parties is $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$, edge identity $edge_{id}$ associated with a real channel between p_u and p_v is (p_u, p_v) where $u < v$.

View of a party generated by execution of multiparty protocol Π Let Π be a multiparty protocol executed by $\mathcal{P}$. We define the *View* of a player as the set of inputs, random bits used by the player and all the messages received by the player during the execution of the protocol. Likewise, the *View* of the adversary is the vector of views of the players, corrupted by the adversary, constituted on execution of the multiparty protocol Π. Further, we define a distribution of the views of the players/adversary as the distribution of the corresponding views constituted from executing the multiparty computation protocol over different random choices made by the players and the adversary. This distribution is defined for a vector of inputs given to the parties engaging in a multiparty computation. Formally,

Let multiparty computation protocol Π be executed by a set of players $\mathcal{P}$. $View_{p_j}^{\Pi,P,\mathcal{A}}(\vec{C}, \vec{I})$ refers to the random variable denoting the view of p_j, when multiparty protocol Π is executed by the set of players P with input vector $\vec{I}$, when adversary $\mathcal{A}$ corrupts quadruplet $\vec{C}$. Correspondingly, the random variable $\overrightarrow{View}_X^{\Pi,P,\mathcal{A}}(\vec{C}, \vec{I})$ denotes the vector of views of subset of players X, constituted from executing Π amongst P with input vector $\vec{I}$.

Similarly, we define distributions over these random variables as $\mathbf{View}_{\mathbf{p_j}}^{\mathbf{\Pi,P},\mathcal{A}}(\vec{\mathbf{C}}, \vec{\mathbf{I}})$ and $\vec{\mathbf{View}}_{\vec{\mathbf{C}}}^{\mathbf{\Pi,P},\mathcal{A},\vec{\mathbf{C}}}(\vec{\mathbf{C}}, \vec{\mathbf{I}})$.

Synchronous versus Asynchronous protocols All the parties are assumed to be synchronized with respect to a global clock. Computation and Communication is structured in rounds. In each round, each party does some computation based on the history of messages received so far, its own input, randomness etc. and sends some messages to other parties on the basis of its computation, which is received by other parties before the beginning of the next round.

Distinction between *Round* and *Step* The term *step* is used to denote (possibly) multiple *rounds* of execution of a multiparty protocol during which some computation is carried out, with which we may typically associate some semantic.

For example, consider the BGW protocol, [BGW88]. Each *step* of the BGW protocol, when executed on a complete network or clique, can consist of a single 'round' in which the parties compute certain values and sends these values to the other parties over secure channels. In the setting of *partially connected networks*, two distant parties may communicate by simulating a 'secure channel' using the rest of the infrastructure of the partially connected network. So for the case of partially connected networks a 'step' may consist of several rounds during which a message is securely transmitted between two nodes.

Execution of a multiparty protocol over a partially connected network N When defining multiparty protocols to be executed over partially connected networks one encounters the issue whether the protocol can at all be executed over a partially connected network or not in the following sense: Suppose a multiparty protocol invokes message transmission function for a message to be sent from party u to party v, while there is no communication channel from party u to party v in the underlying partially connected network. How do we handle this?

This issue can be handled at various levels. We handle it at the level of message transmission on the communication channel: Namely, if during the execution of a multiparty protocol Π, a message transmission function is invoked by party p_u to transmit a message to another party p_v, in round r, while there is no underlying (physical) channel from p_u to p_v, then the message is pushed in the outgoing queue of the messages to be transmitted, in round r, by the multiparty protocol Π. When the messages are dequeued from the outgoing queue and sent on the physical links, the message to be sent on the link from p_u to p_v is dropped as there is no communication channel in the network, on which to send the message.

A.2 Secret key agreement with a public broadcast channel

We shall formally describe a new model for the problem of secret key agreement. The comparison to previous models, the applicability of previous results/techniques and solution are all relegated to the next section.

We are given the following set up:

Given $N+1$ channels $\{F^r(Alice, Bob, id_1), F^r(Alice, Bob, id_2), F^r(Alice, Bob, id_3), \ldots, F^r(Alice, Bob, id_N$ for some constant r, where $\exists i \in [N] : id_i \notin \vec{C}[2], id_i \notin \vec{C}[3]$, where $\vec{C}[2]$ is the subset of channels passively corrupted and $\vec{C}[3]$ is the subset of channels actively corrupted by $\mathcal{A}$. Also, $id_{N+1} \in \vec{C}[2]$. Here we use $\vec{C}$ to denote quadruplet of subsets corrupted, for uniformity of presentation, though $\vec{C}[0] = \vec{C}[1] = \phi$.
To design a protocol to simulate a secure channel or an (almost) secure channel between Alice and Bob availing of the above infrastructure.

We shall formalize what it means for a multiparty protocol to simulate an authentic/secure channel.

Let $\Pi(k) = (\Pi^S(k), \Pi^R(k))$ be a PPT protocol that is executed by sender S and receiver R, to transmit a k bit string. Let $\vec{I} = (i_s, \perp, \perp)$ denote the vector of inputs, of which element $\vec{I}[S]$ denotes the input given to S, $\vec{I}[R]$ denotes the input given to party R and $\vec{I}[\mathcal{A}]$ denotes the input given to $\mathcal{A}$ at the beginning of the protocol. Here, specifically $i_s \in \{0,1\}^k$ is a k-bit string given to sender S. Let $\mathcal{A}$ be an adversary attacking $\Pi(k)$.

Output Specification of protocol $\Pi(k)$: After execution of $\Pi(k)$ the parties output their respective output values, as per the specification of $\Pi(k)$, and $\mathcal{A}$ outputs its entire view generated from the execution of $\Pi(k)$.

Let $Exec_{\Pi(k),S,R,\mathcal{A},l}(\vec{I})$ denote the vector of outputs of the parties when protocol $\Pi(k)$ is executed by S and R with input $\vec{I}$ with adversary $\mathcal{A}$, where l is some security parameter. In particular, $Exec_{\Pi(k),S,R,\mathcal{A},l}(\vec{I})[S]$ refers to the output of party S, $Exec_{\Pi(k),S,R,\mathcal{A},l}(\vec{I})[R]$ refers to the output of party R and $Exec_{\Pi(k),S,R,\mathcal{A},l}(\vec{I})[\mathcal{A}]$ refers to the output view of adversary $\mathcal{A}$.

$\mathbf{Exec_{\Pi(k),S,R,\mathcal{A},l}}(\vec{\mathbf{I}})$ denote the distribution of r. v. $Exec_{\Pi(k),S,R,\mathcal{A},l}(\vec{I})$ taken over the random choices made by S, R and $\mathcal{A}$.

A protocol simulates an authentic channel if its execution between the sender and the receiver, results in receiver outputting the same string as the input given to the sender. The more formal definition is,

Definition A.6 *Let $\vec{I}$, $Exec_{\Pi_k,S,R,\mathcal{A},l}(\vec{I})$, $\Pi(k)$, $\mathcal{A}$ be defined as above.*

Protocol $\Pi(k)$ simulates an authentic channel to send a k bit string, from S to R, if the following holds true:

$$\forall \vec{I} : \vec{I}[S] = Exec_{\Pi(k),S,R,\mathcal{A},l}(\vec{I})[R] \tag{1}$$

Similarly, we can define a secure channel, which will have the additional property of the privacy with respect to the adversary. the formal definition is,

Definition A.7 *Let $\vec{I}$, $Exec_{\Pi_k,S,R,\mathcal{A},l}(\vec{I})$, $\Pi(k)$, $\mathcal{A}$ be defined as above.*

Protocol $\Pi(k)$ simulates a secure channel to send a k bit string, from S to R, if the following conditions hold true:

1. *Correctness:*

$$\forall \vec{I} : \vec{I}[S] = Exec_{\Pi(k),S,R,\mathcal{A},l}(\vec{I})[R]. \tag{2}$$

2. *Privacy:*

$$\forall \vec{I_1}, \vec{I_2} : \vec{I_1}[S] \neq \vec{I_2}[S] : \mathbf{Exec_{\Pi(k),S,R,\mathcal{A},l}}(\vec{\mathbf{I_1}})[\mathcal{A}] \approx \mathbf{Exec_{\Pi(k),S,R,\mathcal{A},l}}(\vec{\mathbf{I_2}})[\mathcal{A}] \tag{3}$$

A relaxation to the above definition is to consider protocols that are allowed to fail to achieve the *Correctness* condition with at most a negligible probability of error, $\mu(l)$, where l is a security parameter. We call the corresponding channels *(almost) authentic* and *(almost) secure* channels.

A.3 Secure computation on partially connected networks

First, we formalize unconditionally secure multiparty computation on partially connected networks. Our main definition will admit mixed networks, however the results we achieve are only for undirected networks.

A.3.1 Secure multiparty computation

The two requirements for security of multiparty computation are *Correctness* and *Privacy*. They are formalized as follows:

The *correctness* requirement is formalized by describing an ideal process in which all the participating parties submit their inputs to a trusted third party T_f which evaluates a multiparty computation function f on these inputs and returns back output values to the parties. A multiparty computation protocol π correctly evaluates multiparty function f if the outputs obtained by the honest parties from running the multiparty computation protocol are same as what they would be from submitting their inputs to the trusted third party T_f and obtaining the outputs from it.

The *privacy* requirement is formalized as follows: A multiparty computation protocol π is private if there exists a PPT simulator S with access to the adversaries program $\mathcal{A}$, input I_C and output values O_C of the subset of corrupted parties T_f, that generates a distribution D_S of the views of the adversary $\mathcal{A}$ that is indistinguishable from the distribution D_π of the views of the adversary generated from real executions of protocol π in which the adversary controls the subset of parties C. We require that the distributions D_S and D_π are perfectly or statistically indistinguishable.

Handling input values in unconditionally secure multiparty computation: In the UC framework, [Can05], an ideal functionality, $\mathcal{F}_{SEC}$ has been proposed for secure function evaluation for the asynchronous case. This functionality consists of the input commitment phase and the evaluation phase. Once all the uncorrupted parties have submitted their inputs to T_f, (i.e., essentially committed to their input values), output is available upon request by any of the parties. The distribution of output values from $\mathcal{F}_{SEC}$ can change significantly depending upon whether, and how many corrupted parties committed to their input values, when an uncorrupted party requests its output. But this ambiguity with respect to defining correctness of evaluation of f, is inherent to the asynchronous regime.

Fortunately, this can be handled cleanly in the synchronous regime. Here, the inputs of some of the parties may be fixed a few rounds earlier than others (till a corrupted party is caught cheating, for example), but is bound to happen before a certain step for all the parties. Defining correctness of computation of function f: First, parties commit to their input values, then the parties compute f on the committed input values. This formulation does not loose any generality.

In [Vay06], the definition presented requires the parties to commit to their input values via some verifiable secret sharing scheme. This is sufficient, as all known multiparty protocols use the VSS scheme, but the description *looks* (if quite interestingly is actually not) somewhat restrictive. Here, we strive for a generic presentation of the definition for the input commitment phase. We characterize the correctness of the input commitment phase, by associating a function $reveal(.,\ldots,.)$ to it and characterizing it. The privacy requirement is handled separately for the entire protocol.

Correctness of the Input Commitment Phase of a multiparty computation protocol

$\Pi = (\Pi^1, \Pi^2)$ refers to a two phase multiparty protocol in which, the parties execute Π^1 to commit to

their individual input values, and then execute Π^2 to evaluate function f on the input values committed to by Π^1.

Party P_i is given input y_i, such that $\forall i \in [n] : y_i \in \{0,1\}^*$. Let $\vec{y} = (y_1, y_2, \ldots, y_n)$ represent the input values given to the parties $\mathcal{P}$.

Let $\Pi^1(\mathcal{P}, \vec{y}, \vec{r^1}, \mathcal{C}, \mathcal{A})$ refers to the vector of input values committed to by $\mathcal{P}$ on the execution of Π^1, starting with some vector of input values $\vec{y}$, randomness $\vec{r}$, when $\mathcal{A}$ corrupts subset $\mathcal{C} \subset \mathcal{P}$.

The input commitment phase should have the following properties (1) All the honest parties are able to commit to their input values y_i they start with, irrespective of the behavior of the adversary $\mathcal{A}$, (2) After the input commitment phase, none of the parties can modify the input values committed to, irrespective of how the corrupted parties behave. In particular, the distribution on the output values is immediately defined after the input commitment phase.

The above requirements are captured as follows: If the input values of all the honest, as well as dishonest parties have been committed to after execution of Π^1, then *there must exist a function $reveal_{\Pi^1}$, associated with Π^1, which when applied on the transcripts of the computations of the parties, reveals the input values committed to by the parties - with the additional property that these committed values are the same as the initial inputs of the honest parties, and may be different from the initial values of the corrupted parties, but unmodifiable from after here on, irrespective of the behavior of the corrupted parties. Furthermore, this should hold even if arbitrary transcripts (different from the ones generated) are substituted for the malicious parties*. Note, that this characterization does not specify which of the inputs of $reveal$ are from the honest parties (But just that $\lfloor \frac{2*n}{3} \rfloor + 1$ of the input values should be from honest parties). The domain of function $reveal()$ is the vector of all feasible transcripts of the parties, and the range is the vector of all feasible input values on which function f is to be evaluated. Formal description follows.

Definition A.8 *$\Pi = (\Pi^1, \Pi^2)$ is a two phase multiparty protocol, to be executed by n parties $\mathcal{P}$, while adversary $\mathcal{A}$ corrupts $\mathcal{C}$, of which Π^1 denotes the input commitment phase, if there exists an n-variate function $reveal_{\Pi^1} : \{\{0,1,\perp\}^*\}^n \rightarrow \{\{0,1\}^*\}^n$, associated with Π^1, which satisfies the following:*

Let $\overrightarrow{Trans} = \Pi^1_{Trans}(\mathcal{P}, \vec{y}, \vec{r^1}, \mathcal{C}, \mathcal{A})$, denote the vector of transcripts of parties $\mathcal{P}$, generated by execution of Π^1, as defined above.

1. *Let $\vec{x} = reveal_{\Pi^1}(\overrightarrow{Trans})$. $\forall P_i \notin \mathcal{C} : x_i = y_i$.*
2. *$\forall \overrightarrow{Trans'} : [(\forall P_i \notin \mathcal{C} : \overrightarrow{Trans'}[i] = \overrightarrow{Trans}[i]) \rightarrow (reveal_{\Pi^1}(\overrightarrow{Trans'}) = reveal_{\Pi^1}(\overrightarrow{Trans}))]$.*

Further, $\Pi^1(\mathcal{P}, \vec{y}, \vec{r^1}, \mathcal{C}, \mathcal{A}) = reveal_{\Pi^1}(\overrightarrow{Trans})$, denotes the vector of input values committed to by Π^1.

Remark A.9 1. The function $reveal()$ formalized above has the salient property that it allows the adversary to delete its transcript generated from execution of Π^1, generate arbitrary transcript on its own and yet not be able to affect the input values committed to by the transcripts of the honest values. (This allows the adversary to behave arbitrarily during the input commitment phase, as well as the computation phase, as long as the adversary is limited to corrupting at most $\frac{n}{3}$ parties.

2. Take for example the BGW protocol. Depending on the specification of the protocol, the cheating parties could be made to commit to d_0 or $d_1 \neq d_0$ etc.. Depending on these specifications, different distributions of the transcripts would be generated by the execution of the input commitment phase of the BGW protocol. However, the function $reveal_{\Pi^1}()$ for the BGW protocol is independent to such specification.

We now present a formal definition of unconditionally secure multiparty computation protocol in the stand alone model, for the synchronous setting.

Let $f : (\{0,1\}^*)^n \rightarrow (\{0,1\}^*)^n$ be an n-ary functionality, where $f_i(z_1, \ldots, z_n)$ denotes the i^{th} element of $f(z_1, \ldots, z_n)$. For $\mathcal{C} = \{i_1, \ldots, i_t\} \subset [n]$, we let $f_{\mathcal{C}}(z_1, \ldots, z_n)$ denote the subsequence $f_{i_1}(z_1, \ldots, z_n), \ldots, f_{i_t}(z_1, \ldots, z_n)$.

Let $\vec{I} = (i_1, i_2, \ldots, i_n)$ be the vector of input values, where input value i_j is given to party P_j.

Definition A.10 *Let f be an n-ary function as defined above. Let $\Pi = (\Pi^1, \Pi^2)$ be a two phase multiparty protocol, as characterized in A.8. Then, Π* **securely evaluates** *f, if the following holds true $\forall \mathcal{C} \subset P$ of parties corrupted by $\mathcal{A}$, such that $|P - \mathcal{C}| \geq \lfloor \frac{2*n}{3} \rfloor + 1$:*

1. *Correctness: Let $\vec{x}$ be the vector of input values committed to by execution of Π^1.[2]. Then,*
   $\Pi^2(\vec{x}, \mathcal{A})_{P-\mathcal{C}} \approx f(\vec{x}, r)_{P-\mathcal{C}}$
2. *Privacy: There exists a PPT simulator S, that runs in time polynomial in the complexity of $\mathcal{A}$, which takes as input the subset $\mathcal{C}$, $\vec{y}_{\mathcal{C}}$ and $f(\vec{x}, r)_{\mathcal{C}}$, adversary program $\mathcal{A}$, and generates the view of $\mathcal{A}$, such that the distribution of the views of $\mathcal{A}$ generated from real execution of Π is indistinguishable from the distribution of the views of $\mathcal{A}$ generated by S:*
   $S^{\mathcal{A}}(\mathcal{C}, \vec{y}_{\mathcal{C}}, f(\vec{x}, r)_{\mathcal{C}}) \approx \vec{View}_{\mathcal{C}}^{\Pi,\mathcal{A}}(\mathcal{C}, \vec{x}, \vec{y})$

for all feasible adversaries $\mathcal{A}$[3].

A.4 Secure multiparty computation with corruption of parties and channels on a partially connected network

As an intermediate step to defining secure multiparty computation over partially connected network we propose a model for secure multiparty computation, in which the adversary can corrupt a subset of parties, as well as a subset of secure channels connecting these parties. The channels between the parties may be corrupted passively or actively. If the channel between two parties is corrupted passively, then the new channel is behaviorally equivalent to an *authentic channel*. If the channel between two parties is corrupted actively, then the new channel is behaviorally equivalent to a *tamperable channel*.

[2]As defined above, the vector of input values committed to by the parties is specified by vector $\vec{x} = \overrightarrow{\Pi^1(\mathcal{P}, \vec{y}, \vec{r^1}, \mathcal{C}, \mathcal{A})} = reveal_{\Pi^1}(\overrightarrow{Trans})$

[3]The variable $\overrightarrow{View}_{\mathcal{C}}^{\Pi,\mathcal{A}}(\mathcal{C}, \vec{x}, \vec{y})$ is as defined above, at the beginning of the section, and used rather canonically in cryptographic literature

How to model the corruption of channels? Let $\vec{C}$ denote a quadruplet of corruption as according to adversary structure, Definition A.1.

As described above, if there exists a secure channel, with identity $sec - id$, between parties P_u and P_v, represented by $F^r(P_u, P_v, sec - id)$, then passive corruption of this channel makes it behaviorally equivalent to $F_r(P_u, P_v, sec - id)$, with $sec - id \in \vec{C}[2]$ and active corruption of this channel makes it behaviorally equivalent to $F_r(P_u, P_v, sec - id)$, with $sec - id \in \vec{C}[3]$. Since we are only interested in static corruptions, the behavior of the channels is fixed once the adversary has corrupted a subset of nodes and edges at the beginning of the protocol.

We note that for this setting not all honest parties are guaranteed to receive their correct output values or are able to preserve the privacy of their input values. Honest parties, for which this is not ensured, are called *sacrificed*. Informally, we say a multiparty protocol Π executed on network N, $\mathcal{T}$-securely evaluates function f, if for every feasible $\vec{C}$, corrupted by $\mathcal{A}$, there exists a subset of honest nodes H for which we achieve the guarantees of multiparty computation i.e., all the parties in H receive their correct output values, while maintaining the privacy of their input values. At least $\lfloor \frac{2n}{3} \rfloor + 1$ honest parties should be able to communicate via pairwise secure communication channels for this to happen.

For this, we are required to realize secure communication channels between distant pairs of nodes of a partially connected network, when the network has some special properties.

A.4.1 Secure communication over partially connected networks

In a partially connected network of low degree it may not be possible to establish secure channels between every pair of honest parties. For example, consider a partially connected network in which an honest node is surrounded by corrupted parties. This honest node is segregated from the rest of the network. Obviously, it is not possible to establish a secure channel between this honest node and any other honest node of the network. Considering the limitations of our setting we relinquish such hopes and focus on achieving statistical guarantees which may be sufficient for our purpose.

Our focus is not on the identities of the nodes which may belong to the clique of honest nodes H, such that each pair of parties belonging to H shares a *secure communication channels*, but on bounding the size of subset H for choices of quadruplet $\vec{C}$ of parties corrupted by $\mathcal{A}$. For this we first focus on establishing secure channels between a given pair of nodes of the network, as long as the corrupted parties and channels belong to a certain subset.

Multiparty protocol is said to realize $\mathcal{T}$-secure channel between u and v, if it realizes a secure channel between u and v, for all $\vec{C} \in \mathcal{T}$, corrupted by $\mathcal{A}$. There are no guarantees made about the type of channel realized when $\vec{C} \notin \mathcal{T}$.

There are two properties of a secure channel: *Correctness* and *Privacy*. Here, we allow the protocol to fail to achieve correctness with a negligible probability of error and target (almost) correctness instead of perfect correctness. For privacy condition we demonstrate a simulator Sim, running in time polynomial in the running time of adversary $\mathcal{A}$, such that the distribution of views of $\mathcal{A}$ generated by real executions of the protocol is indistinguishable from the distribution of the views generated by the simulator Sim. We demonstrate such a simulator even if the channel realized by the multiparty protocol is not private or authentic - this is for technical reasons as the simulator may be invoked while proving security of higher level protocols. Message being transmitted between the parties is made available to the simulator in case

the channel realized is authentic or tamperable. Only the size of the message transmitted is made available in case the channel realized is secure.

Now, we formalize the above setting and define a $\mathcal{T}$-secure channel on a partially connected network N.

A.4.2 Definition of $\mathcal{T}$-secure channel

Let adversary be restricted to adversary structure $\mathcal{T}$.

Let $N = (V, E)$ be a partially connected network. Let $\vec{I} = (\perp, \perp, \ldots, m_u, \perp, \ldots, \perp)$ and $\vec{O} = (\perp, \perp, \perp, \ldots, m_v, \perp, \ldots, \perp)$ denote corresponding vectors of inputs and outputs.

Let $\overrightarrow{View}_C^{\gamma,P}(\vec{I})$ be defined as above.

Definition A.11 *($\mathcal{T}$-secure channel) Let $\vec{I}, \vec{O}$ and N be as defined above. Let l be a security parameter.*

Multiparty protocol $\gamma(p_u, p_v, N, s, c)$ executed on network N, realizes $\mathcal{T}$-secure channel between nodes u and v, if for all input vectors $\vec{I}$, quadruplet $\vec{C}$ corrupted by $\mathcal{A}$, such that $\vec{C} \in \mathcal{T}$, the following conditions hold true:

1. *Correctness: $I[u] = O[v]$ with probability $\geq 1 - \mu(l, |I[u]|)$, for all sufficiently large l, for some negligible function $\mu()$.*
2. *Privacy: There exists a simulator Sim that takes as input, the network topology N, $\vec{C}$, the program of adversary $\mathcal{A}$, size of input message $|I[u]|$, and runs in time polynomial in running time of $\mathcal{A}$ and generates a distribution of views of $\mathcal{A}$ such that:*

   $Sim^{\mathcal{A}}(\vec{C}) \approx \overrightarrow{View}_{\vec{C}}^{\Pi,P}(\vec{I})$.

A.4.3 Definition of $\mathcal{T}$-authentic channel over partially connected networks

The definition of a multiparty protocol that realizes an authentic channel over a partially connected network $N = (V, E)$ is formalized along similar lines. For the authentic channel, we relax the privacy condition of the definition of $\mathcal{T}$-secure channel.

We also add a *simulatability* condition which says that the view of the adversary should be simulatable for any protocol realizing the authentic channel on the partially connected network. Since the simulator is given as input the message being transmitted, the simulatability condition follows trivially for this case. We would invoke this simulator, while proving security of higher level protocols.

Definition A.12 *($\mathcal{T}$-authentic channel) Let $\vec{I}, \vec{O}$ and N be as defined above and l be a security parameter.*

Protocol $\gamma(p_u, p_v, N, s, c)$ executed on network N, realizes $\mathcal{T}$-authentic channel between p_u and p_v, if for all input vectors $\vec{I}$, all quadruplets $\vec{C}$ corrupted by $\mathcal{A}$, such that $\vec{C} \in \mathcal{T}$, the following conditions hold true:

1. *Correctness:* $\vec{I}[p_u] = \vec{O}[p_v]$.

2. *Simulatability: There exists a simulator* Sim *that runs in time polynomial in the running time of* $\mathcal{A}$ *and generates a distribution of views of* $\mathcal{A}$ *that is indistinguishable from the distribution of views of* $\mathcal{A}$ *generated from a real execution of* γ*. Furthermore,* Sim *is given input* $inp = \vec{I}[p_u]$ *iff* $p_u \notin \vec{C}[0] \bigcup \vec{C}[1]$*, else* $inp = \perp$*. Namely,*

   $Sim^{\mathcal{A},p_u,p_v,N}(\vec{C}, inp, l, c) \approx \overrightarrow{View}_{\vec{C}}^{\Pi,\mathcal{P}}(\vec{I})$.

A.4.4 Unconditional secure multiparty computation with corruption of parties as well as channels

We now formalize a definition of unconditionally secure multiparty computation which models corruption of parties, as well as channels, passively as well as actively. As discussed above, we model this by *sacrificing* a few honest nodes in the following manner:

1. Correctness of the outputs of the sacrificed honest parties is not guaranteed.

2. Privacy of the inputs as well as outputs of the sacrificed honest parties is not required to be preserved. For this we show a simulator, which when given the input and output values of all the corrupted parties as well as the sacrificed honest parties is able to simulate the view of $\mathcal{A}$ generated from real execution of the multiparty protocol. Essentially, this puts a bound on the amount of knowledge on the input values of the honest parties, that can be leaked during the computation process in the worst case.

First, we give a relaxed description of the two-phase multiparty computation which incorporates the notion of the sacrificed honest parties as described above. We present a weaker characterization of the correctness of commitment of input values, than given above to incorporate sacrificing of honest parties. The new characterization is that (1) there exists a subset of honest parties $\mathcal{H}$ of size $\geq \lfloor \frac{2*n}{3} \rfloor + 1$, which should be able to commit to their initial input values, and which together share the committed input values of all the parties. (2) the phase may fail with negligible probability.

Let $\mathcal{A}$ be an adversary restricted to adversary structure $\mathcal{T}$, Definition A.1.

Definition A.13 $\Pi = (\Pi^1, \Pi^2)$ *is a two phase multiparty protocol, executed on network* N *by set of parties* $\mathcal{P}$*, while* $\mathcal{A}$ *controls* $\vec{C}$*, where* Π^1 *denotes the input commitment phase, if there exists an* n*-variate function* $reveal_{\Pi^1} : \{\{0,1,\perp\}^*\}^n \rightarrow \{\{0,1\}^*\}^n$*, associated with* Π^1*, for which* Π^1 *and* $reveal_{\Pi^1}$ *satisfy the following:*

Let $\overrightarrow{Trans} = \Pi^1_{Trans}(\mathcal{P}, \vec{y}, \vec{r^1}, \vec{C}, \mathcal{A})$*, denote the vector of transcripts of parties* $\mathcal{P}$*, generated by the execution of* Π^1*, as defined above. There exists a subset of honest parties* $\mathcal{H} \subset \mathcal{P}$*,* $|\mathcal{H}| \geq \lfloor \frac{2*n}{3} \rfloor + 1$ *for which the following two conditions hold true:*

1. *Let* $\vec{x} = reveal_{\Pi^1}(\overrightarrow{Trans})$.
   $\forall P_i \in \mathcal{H} : x_i = y_i$, *with probability greater than* $1 - \mu(n)$, *for some negligible function* $\mu(.)$.

2. $\forall \overrightarrow{Trans'} : (\forall P_i \in \mathcal{H} : \overrightarrow{Trans'}[i] = \overrightarrow{Trans}[i]) \rightarrow (reveal_{\Pi^1}(\overrightarrow{Trans'}) = reveal_{\Pi^1}(\overrightarrow{Trans}))$.

Further, let $\Pi^1(\mathcal{P}, \vec{y}, \vec{r^1}, \vec{C}, \mathcal{A}) = reveal_{\Pi^1}(\overrightarrow{Trans})$, *denote the vector of input values committed to by* Π^1.

We allow the input commitment phase of the multiparty protocols executed on partially connected networks to fail with a negligible probability. This relaxation is a must as we only establish (almost) secure channels on the partially connected network, which may fail to deliver their correct messages with at most negligible probability, and hence the input commitment phase may also fail with a negligible probability.

We formalize unconditionally secure multiparty computation on partially connected networks.

Let $f : (\{0,1\}^*)^n \rightarrow (\{0,1\}^*)^n$ be an n-ary functionality, where $f_i(z_1, \ldots, z_n)$ denotes the i^{th} element of $f(z_1, \ldots, z_n)$. For $C = \{i_1, \ldots, i_t\} \subset [n] = \{1, \ldots, n\}$, we let $f_C(z_1, \ldots, z_n)$ denote the subsequence $f_{i_1}(z_1, \ldots, z_n), \ldots, f_{i_t}(z_1, \ldots, z_n)$.

Definition A.14 *Let* $f, \vec{y}, P, \mathcal{A}, \vec{C}$ *be defined as above.*

$\Pi = (\Pi^1, \Pi^2)$ *is a two-phase multiparty computation protocol, A.13, executed on network* N, $\mathcal{T}$*-securely evaluates* f *if for all* $\vec{C} \in \mathcal{T}$, *A.1, corrupted by* $\mathcal{A}$, *there exists a subset of parties,* $H \subset P - \vec{C}[0] - \vec{C}[1]$, $|H| \geq \lfloor \frac{2n}{3} \rfloor + 1$, *for which the following condition holds true:*

1. *Correctness: Let* $\vec{x}$ *be the vector of input values committed to, by the parties, after the execution of input commitment phase* Π^1.[4] *Then,*

   $\Pi^2(\vec{x}, \vec{C}, \mathcal{A}, \vec{r^2})_H \approx f(\vec{x}, \vec{r})_H$

2. *Privacy: There exists a simulator* S, *that runs in time polynomial in the complexity of* $\mathcal{A}$, *which takes as input* $\vec{C}$, $\vec{y}_{\vec{C}}$, $f(\vec{x})_{\vec{C}}$ *and adversary program* $\mathcal{A}$ *and generates distribution of views of* $\mathcal{A}$, *such that:*

   $S^{\mathcal{A}}(\vec{C}, \vec{y}_{P-H}, f(\vec{x}, \vec{r})_{P-H}) \approx \overrightarrow{View}^{\Pi,\mathcal{A}}_{\vec{C}}(\vec{C}, \vec{y}, \vec{r^1} \circ \vec{r^2})$, *where* $\vec{r^1}, \vec{r^2}$ *refer to the sequence of random bits used during phase* Π^1 *and* Π^2, *respectively.*

for all feasible $\mathcal{A}$.

[4] $\vec{x} \leftarrow \Pi^1(\vec{y}, \vec{C}, \mathcal{A}, \vec{r^1})$, as defined above

B Secret key agreement by public discussion - revisited

We shall propose a solution to the problem of secret key agreement via public discussion, proposed in sub subsection A.2.

B.1 (almost) secure channel using an authentic channel

We present a protocol that realizes an (almost) secure channel from Alice to Bob in the model proposed in subsection A.2. Alice and Bob share the following $N+1$ channels:

$$\{F^r(Alice, Bob, id_1), F^r(Alice, Bob, id_2), F^r(Alice, Bob, id_3), \ldots, F^r(Alice, Bob, id_{N-1}), F^r(Alice, Bob, id$$

for some constant $r \geq 3$, where $\exists i \in [N] : id_i \notin \vec{C}[2] \bigcup \vec{C}[3]$ and $id_{N+1} \in \vec{C}[2]$ i.e., Channel id_{N+1} is given to be authentic and at least one of the N channels is known to be secure.

We describe a protocol for transmitting a single bit message m from Alice to Bob (almost) correctly. Protocol $\Pi_1 = (\Pi_1^{Alice}, \Pi_1^{Bob})$ is described below. $Alice \rightarrow Bob$: refers to the step in which Alice does computation and sends some messages to Bob.

1. $Alice \rightarrow Bob$: Along each channel $F^r(Alice, Bob, id_j)$, $\forall j \in [N]$, Alice sends a block of uniformly chosen sequence of random bits of length $k(N, s)$.

   Here $k(N, s)$ is some polynomial in N, s, specified later.

2. $Alice \rightarrow Bob$: For each channel $F^r(Alice, Bob, id_j)$, $\forall j \in [N]$, Alice chooses an integer $i_p \in \{1, \ldots, k(N, s)\}$ uniformly at random and sends the set of these integers to channel $F^r(Alice, Bob, id_{N+1})$.

3. $Alice \rightarrow Bob$: $\forall j \in [N]$, Alice hides i_p^{th} bit of the sequence sent through channel $F^r(Alice, Bob, id_j)$ in previous Step and reveals the rest to Bob by sending to channel $F^r(Alice, Bob, id_{N+1})$.

4. $Bob \rightarrow Alice$: Bob identifies the ***faulty*** channels as follows:

   Channel id_j is identified as ***faulty*** if the subsequence sent along the i^{th} channel in Step 3, is different from the corresponding sequence received from Alice (excepting the hidden bit specified in Step 2) along i^{th} channel in Step 1.

   Bob reveals the identities of the channels identified as ***faulty***, by sending them to channel $F^r(Bob, Alice, id_{N+1})$.

5. $Alice \rightarrow Bob$: Alice excludes all the faulty channels and chooses the hidden i_p^{th} bits of the good channels and computes an exclusive-OR of them with the bit m to be sent and sends the resulting bit c to channel $F^r(Alice, Bob, id_{N+1})$

6. Bob : Let the bit received by Bob in the previous step be c. Bob computes the exclusive-OR of the hidden bits at its own end and extracts $c \oplus b^*$ as the bit meant for it.

7. Alice and Bob repeat the above steps parallely $l(s)$ times. Bob accepts the majority value as its output value.

We shall prove that protocol $\Pi(1)$ described above realizes an (almost) secure channel for transmitting a single bit message from Alice to Bob, as according to the definitions given in subsection A.2.

Theorem B.1 *Protocol $\Pi(1)$ simulates an (almost) secure channel between Alice and Bob.*

Proof: We have to prove the following two conditions:

1. Privacy: To prove that the communication channel, simulated between Alice and Bob by the above protocol, is private we need to show that the distribution of the views of the adversary when bit $m =' 0'$ is sent through the channel is indistinguishable from the distribution of the views of the adversary when bit $m =' 1'$ is sent i.e.,

   $$\mathbf{Exec}_{\mathbf{\Pi(1),Alice,Bob},\mathcal{A},\mathbf{l}}(\overrightarrow{('0', \perp)})[\mathcal{A}] \approx \mathbf{Exec}_{\mathbf{\Pi(1),Alice,Bob},\mathcal{A},\mathbf{l}}(\overrightarrow{('1', \perp)})[\mathcal{A}]$$

   First, observe that Step 5 is the only step when Alice sends a message to Bob that is dependent on input bit m. More specifically, the message sent by Alice to Bob in Step 5 is $m \oplus b_{i_j} \oplus \ldots$. Since it is known that one of the N channels is secure with respect to $\mathcal{A}$, one such random bit b_{i_j} used in Step 5 to hide the message bit in the last step is uncorrelated with respect to the view of the adversary. Hence, $m \oplus b_{i_j} \oplus \ldots$ is uncorrelated with the rest of the view of the adversary. Thus, the following two distributions are (perfectly) indistinguishable:

   $$\mathbf{Exec}_{\mathbf{\Pi_k},\mathbf{S},\mathbf{R},\mathcal{A},\mathbf{l}}(\vec{\mathbf{I_1}})[\mathcal{A}] \approx \mathbf{Exec}_{\mathbf{\Pi_k},\mathbf{S},\mathbf{R},\mathcal{A},\mathbf{l}}(\vec{\mathbf{I_2}})[\mathbf{R}]$$

2. (almost) Correctness:

   $$Pr_{r_s,r_r}(R_e(Ch, \Pi_s, r_s, b, \Pi_r, r_r) \neq m) = \mu(s) \tag{4}$$

   for some negligible function $\mu(.)$ and security parameter s.

   We shall show a simple lower bound on the probability of Alice and Bob agreeing on the same bit. We shall do it in two steps. First, we shall prove a lower bound on the probability of the event that all the channels chosen to hide bit 'b' in Step 4 are good i.e., were not manipulated by the adversary. Then, we shall do a standard probability amplification using Chernoff's bound.

   Define X_i to be the event that Channel id_i was faulty and chosen. Correspondingly, define X_i for all $i \in [N]$. Then, the probability of choosing at least one faulty channel is $Pr[X = \bigcup_{i \in N} X_i]$. This is upper bounded by the Union bound, $Pr[X = \cup_{i \in N} X_i] \leq \sum_{i=1}^{i=N} Pr[X_i]$.

   The probability that a faulty channel id_i was chosen for transmission is $Pr[X_i] \leq \frac{1}{k(N,s)}$. Thus, in the worst case we have $Pr[X = \bigcup_{i \in N} X_i] \leq \frac{N}{k(N,s)}$ as the upper bound of choosing at least one faulty channel. The probability that all the channels chosen are good is $Pr[\overline{X}]$, $Pr[\overline{X}] \geq 1 - \frac{N}{k(N,s)}$

   In this case we are assured of a good transmission i.e., the minimum probability of a good transmission is $1 - \frac{N}{k(N,s)}$. If $k(N, s)$ is chosen to be $N * f(s)$ for some function $f(.)$, then this expression evaluates to $Pr[\overline{X}] \geq 1 - \frac{N}{k(s)} \geq 1 - \frac{1}{f(s)}$

   The probability that Bob receives incorrect bit in Step 5, $Pr[X]$, is at most $\frac{1}{f(s)}$. Alice and Bob repeat Step 1 through Step 5, $l(s)$ times and take the majority value as the answer. By Chernoff's bound, the probability that Alice and Bob agree on dissimilar bits is: $Pr(b' \neq b) \leq exp(-\mu * \frac{\delta^2}{2})$, where $\delta = \frac{1}{2} - \frac{1}{2*(f(s)-1)}$ and $\mu = l(s) * (1 - \frac{1}{f(s)})$.

   The transmitted bit is correct with probability at least $1 - \exp^{\frac{-(f(s)-2)^2 * l(s)}{8*f(s)*(f(s)-1)}}$. For $f(s) = 3$ and $l(s) = s$, the probability of transmitting a bit correctly is at least $1 - exp^{O(s)}$.

Protocol $\Pi(1)$ simulates a private and (almost) correct and hence (almost) secure channel to transmit a one bit message from Alice to Bob. □

Remark B.2 The extension to send messages of arbitrary length is straightforward. One can use error correction codes to reduce the bit complexity of the above protocol, but our focus here is not on the efficiency of the protocol but just an existential result, that we can deploy later.

C (almost) Secure channels on partially connected networks

Let the adversary be restricted to some adversary structure $\mathcal{T}$. We give a multiparty protocol that establishes $\mathcal{T}$-secure channels between a pair of nodes of a partially connected network N as long as the adversary corrupts $\vec{C} \in \mathcal{T}$.

Let $N = (V, E)$ represent the communication infrastructure available between the set of parties V, with the following Special Property ($\mathcal{T}_{p_u,p_v,\beta,c}$):

Definition C.1 *Network N possesses Property $\mathcal{T}_{p_u,p_v,\beta,c}$, if the following two conditions hold true:*

1. *For every $\vec{C} \in \mathcal{T}$, corrupted by $\mathcal{A}$, there exists a path of secure channels, connecting uncorrupted nodes, of length at most $c * \lg_2 n$ from p_u to p_v.*
2. *There exists a multiparty protocol $\beta(p_u, p_v, N)$ that realizes $\mathcal{T}$-authentic channel between p_u and p_v. (The channel established by β is also referred to as the additional channel between p_u and p_v.)*

Now we describe a multiparty protocol $\gamma(p_u, p_v, N, s, c)$ for node p_u to send a single bit message to node p_v, securely using the infrastructure of network $N = (V, E)$. Let $l(n)$ be some polynomial in n, which will be defined according to the guarantee of security made for γ.

We make a few highlighting remarks about the features of the multiparty protocol $\gamma(p_u, p_v, N, s, c)$:

1. Recall the definition of (physically realized) authentic and secure channels of network N, in Section A. Each of these channels is parameterized by variable r, which refers to the number of rounds after which a message sent by the sender, along the channel, reaches the receiver.
2. Each *Step* of $\gamma(p_u, p_v, N, s, c)$ consists of multiple rounds. The first step consists of a total number of $(c * \lg_2 n) * r$ rounds, where the constant c depends on network topology N. A message sent over the *additional channel* realized by $\beta(p_u, p_v, N)$ takes $r * r_{\beta,N}$ rounds.
3. The protocol $\gamma(p_u, p_v, N, s, c)$ runs for a total of $c * \lg_n * r + 5 * r_\beta * r + 2 * r$ rounds. No messages are exchanged in the last two rounds. The presence of constant $2 * r$ is due to technical reasons, to carry out higher level proofs of multiparty computation.

MPC protocol $\gamma(p_u, p_v, N, s, c)$: Setup Phase.
Enumerate all paths of length at most $c * \lg_2 n$ between p_u and p_v in network N. Let $PA = \{pa_1, pa_2, \ldots, pa_w\}$ be the set of paths, w in number. The set PA and the path-ids etc. are known to all the parties of the network as a result of the set up phase.

1. $p_u \rightarrow p_v$: Along each path $pa_i \in PA, \forall i \in [n]$ between p_u and p_v, send a block of $k(w, s)$ uniformly chosen random bits, for some polynomial $k(.,.)$ that will be specified later. (After forwarding the block to the next node on the path, the intermediate node deletes the block.)

2. $p_u \rightarrow p_v$: For each of the w paths, choose an integer $i_p \in \{1 \ldots k(w, s)\}$ uniformly at random and send this set of integers to the node p_v over the *additional channel*, realized by multiparty protocol $\beta(p_u, p_v, N)$.

3. $p_v \rightarrow p_u$: Node p_v hides the i_p^{th} bit from the respective block of random bits, and sends the rest of the block to p_u over the *additional channel*.

4. $p_u \rightarrow p_v$: Node p_u identifies the *faulty* paths, as follows: p_u checks for mismatch between the block created after removing the hidden bit (i.e., i_p^{th} bit) from the respective block sent in Step 1, and the block sent in Step 3 (for the same path).
   If all the paths are identified as *faulty*, then p_u sends a message **PROTOCOL-ABORT** to p_v over the *additional channel*. Otherwise, let $b^* = \bigoplus_{j=1}^{g} b_{i_j}$ be the exclusive-or of the hidden bits from all the paths identified as non-faulty. The node p_u sends the following to node p_v over the *additional channel*:

   (a) The subset of the identities of all the paths identified as *non-faulty*.

   (b) If m_o is the bit to be sent, send $b_o = m_o \oplus b^*$ over the *additional channel* to node p_v.

5. p_v: Let b'_o be the bit received in the previous Step. p_v receives the identities of all the "non-faulty" paths and computes the XOR of the "hidden" bits of the blocks sent over each of the paths in previous Step i.e., its own values of b_{i_j}'s and the bit b'_o received in the previous Step.

   Node v extracts the bit for each of the $l(s)$ parallel executions and accepts the majority as its output value.

Simulatability of adversaries view for multiparty $\gamma(p_u, p_v, N, s, c)$. If network N possesses certain

Property $\mathcal{T}_{p_u,p_v,\beta,c}$, C.1, multiparty protocol $\gamma(p_u, p_v, N, s, c)$ realizes a $\mathcal{T}$-secure channel between p_u and p_v. For technical reasons, we demonstrate a simulator that simulates the view of the adversary for all the different cases and not just when $\vec{C} \in \mathcal{T}$. This simulator is invoked while proving the security of higher level protocols.

When the channel simulated by $\gamma(p_u, p_v, N, s, c)$ is to be proved secure, then the simulator will not be given any knowledge about the message being transmitted, except for the length of the message. The simulator is given the message being transmitted on the channel for all other cases. We consider each case separately and qualify it accordingly.

Theorem C.2 *Let multiparty protocol $\gamma(p_u, p_v, N, s, c)$ be executed on network N with Property $\mathcal{T}_{p_u,p_v,\beta,c}$, C.1.*

There exists a simulator, that runs in time polynomial in running time of $\mathcal{A}$ and generates a distribution of views of $\mathcal{A}$ that is indistinguishable from distribution of views of $\mathcal{A}$ generated from executing $\gamma(p_u, p_v, N, s, c)$ on N, for all quadruplets $\vec{C} \in \mathcal{C}$ corrupted by $\mathcal{A}$.

Proof: Let $\mathcal{A}$ corrupt quadruplet $\vec{C} \in \mathcal{T}$. Depending on the subset of nodes corrupted by $\mathcal{A}$, we shall have several cases and the behavior of the simulator may be different for each case.

We shall divide the proof according to the choice of parties and channels corrupted by $\mathcal{A}$, and whether additional channel simulated by $\beta(p_u, p_v, N)$ is authentic or tamperable. In either case, there exists a simulator Sim^{β} that generates a distribution of views of $\mathcal{A}$ that is indistinguishable from the views of $\mathcal{A}$ generated from real execution of multiparty protocol β. (In particular, when p_u is honest, then the simulator is given the input message to be sent by p_u to p_v and if p_u is faulty, then $\mathcal{A}$ itself is the generator of the message).

We consider each case separately:

1. Nodes p_u, p_v are honest and there exists at least one path of non-faulty nodes of length at most $c * \lg_2 n$ between p_u and p_v In this case simulator Sim^{β} is given as input the message to be sent.

   (a) $\vec{C} \in \mathcal{T}$ In this case we demonstrate a simulator that simulates the view of $\mathcal{A}$ but takes no input i.e. is a Zero-Knowledge simulator, thus providing perfect *Privacy* to the channel established.

   The simulator $\mathcal{S}$. The simulator $\mathcal{S}$ is given as input $\vec{C}$ corrupted by the adversary $\mathcal{A}$ and whether the additional channel is authentic or tamperable. Note that for the MPC protocol $\gamma(p_u, p_v, N, s, c)$, only p_u receives an input message and only p_v outputs an output value, the rest of the parties just participate in the protocol $\gamma(p_u, p_v, N, s, c)$ without taking any inputs or producing any outputs. Next, also note that *the input message bit* is used only in the last Step of the protocol - the rest of the steps only involve exchanging uniformly random bits. The simulation of the protocol is quite straightforward from hereon.

   We are going to show the simulation of execution of one sequence of steps of protocol $\gamma(p_u, p_v, N, s, c)$. The extension to $l(s)$ independent parallel repetitions for each different message bit or repetition for amplification is a straightforward extension. Note, that in this case the simulator is given as input the length of the message.

The simulator $\mathcal{S}$ simulates the role of all the honest parties, p_u and p_v as it is. In the first step the simulator chooses a sequence of uniformly chosen random bits for each path and simulates the sending of this sequence of uniformly chosen random bits along the path as follows: When an intermediate party sends a message to an adversary corrupted party, the simulator sends the message to $\mathcal{A}$ which may modify the message before passing it to the next honest party (which is simulated by the simulator).

Then, $\mathcal{S}$ chooses the identity of one bit from each sequence of random bits and invokes the simulator Sim^{β} for simulating sending this set of identities from p_u to p_v through an authentic channel.

In the next step the simulator takes all the sequences, in their entireties except for the hidden bits (in previous step) and invokes Sim^{β} for simulating sending this set of identities from p_u to p_v through an authentic channel simulated by protocol β.

In the next step the simulator identifies the faulty paths based on the corrupted blocks in first Step, and invokes the simulator Sim^{β} for simulating sending the identities of these paths through an authentic channel simulated by protocol β.

Now since we are given that there exists at least one non-faulty path in network N between p_u and p_v we know that one at least random bit is completely hidden from the adversary - there may be more but at least one is completely hidden. The simulator computes the XOR of all of these hidden bits (chosen from all non-faulty paths) with r' and invokes simulator Sim^{β} on this input.

[Simulation for *parallel* repetitions for the same bit or for different repetitions for different message bits is a straightforward extension - the proof follows by use of hybrid argument, la [GM82]]

Observe that Sim^{β} and hence simulator Sim run in time polynomial in the running time of the adversary $\mathcal{A}$.

Proof of indistinguishability of $\mathcal{A}$'s view. Intuitively, the argument is as follows: During the simulation of the first few steps of the protocol only uniformly chosen random bits are used(exchanged between) the parties. Now, for the choice of $\vec{C} \in \mathcal{T}$ there is one path secure from the adversary. The uniformly chosen random bit from this path remains secure from $\mathcal{A}$ and is used to hide the message bit used in the last step of γ. But the XOR of a random bit with the message bit creates a ("random" looking) bit that is indistinguishable from a random bit. Thus, the view of $\mathcal{A}$ generated by the simulator or in the real execution of γ is indistinguishable. Details follow.

We show that the distribution of the views of $\mathcal{A}$ generated by the above simulator, which doesn't take any input, is indistinguishable from the distribution of the views of the adversary generated from execution of $\gamma(p_u, p_v, N, s, c)$ with some arbitrary message m.

For this we show that this is the case after simulation of every step of γ. First, notice that the adversary plays an active role in formulating its view only in the first step - after the first step of protocol $\gamma(p_u, p_v, N, s, c)$ the adversary is passive and just passively collects all messages (since the additional channel simulated by β is authentic). We are going to show indistinguishability of the distribution of adversaries view after every step of $\gamma(p_u, p_v, N, s, c)$.

In the first step notice that the *probability* with which the adversary chooses to corrupt the blocks of uniformly chosen random bits sent on the faulty paths is the same whether its a real execution of $\gamma(p_u, p_v, N, s, c)$ or the simulation. In particular, the *probability* with which the adversary chooses to corrupt the blocks, sent on some particular paths, at some particular corrupted nodes and at some particular positions is exactly the same for the simulation of the ideal world as it is in the real execution of $\gamma(p_u, p_v, N, s, c)$ given that the blocks were chosen uniformly at random. Thus, the distribution of views of the adversary $\mathcal{A}$ generated after the real execution of first step of $\gamma(p_u, p_v, N, s, c)$ or after the simulation of the first step of $\gamma(p_u, p_v, N, s, c)$ by the simulator is exactly the same or a Distinguisher may be constructed using the adversaries program which can distinguish between two sources of uniformly random bits - contradiction.

Now notice that the adversary doesn't contribute actively in the constitution of its view, after the first step of the protocol, as it just receives all the messages exchanged between p_u and p_v on $\mathcal{T}$-authentic channel.

Fix a view V_w of the adversary constituted up till Step 1. We know that the probability that this has been picked from such views generated from real execution of protocol $\gamma(p_u, p_v, N, s, c)$ or from ideal world simulation is equal i.e., $\frac{1}{2}$.

Observe also that for protocol $\gamma(p_u, p_v, N, s, c)$ every view of the honest parties generated till Step 1, that is compatible to the same view of the adversary generated till Step 1, is equally likely to be from real life execution of $\gamma(p_u, p_v, N, s, c)$ or from ideal world simulation. Thus we can fix a given view of the honest parties as well as the view of the adversary till Step 1 and this is equally likely to be from real world execution of protocol $\gamma(p_u, p_v, N, s, c)$ or from ideal world simulation.

Observe that once the view of the honest parties and the adversary is fixed for Step 1, if the same random choices are made for nodes p_u and p_v during Step 2 through Step 5 in real execution of β as well as in simulation by the simulator Sim^{β}, the view of the adversary generated is identical.

So the distribution of the views of the adversary generated so far (i.e., up to Step 5 of $\gamma(p_u, p_v, N, s, c)$), constituted from real execution or ideal world simulation are identical.

The Step 6 needs a slightly different analysis because in this step messages are sent from node p_u to p_v which depend on the actual message m. This message m is available to the node p_u in the real world execution of $\gamma(p_u, p_v, N, s, c)$ but not to the simulator because the simulator is not given any input. Observe that for this case, that since there exists at least one path of non-faulty nodes between nodes p_u and p_v along which a block of uniformly random bits was sent in Step 1, therefore at least one random bit is secure from the adversary. This random is also used to hide the "message bit" in Step 6. Let this random bit be r_0. The adversaries view for this step would be $r_0 \oplus m_i$ for real world execution and a uniformly chosen random bit r' for the simulation. However since r_0 is a uniformly chosen random bit completely hidden from the adversary (just like r' is uniformly chosen random bit uncorrelated to the rest of the view of the adversary) and hence uncorrelated to the rest of the view of the adversary $r_0 \oplus m_i \approx r'$. This completes the argument for the indistinguishability of the distribution of the views of $\mathcal{A}$ generated from execution of one sequence of Steps 1 through Step 6, of protocol

$\gamma(p_u, p_v, N, s, c)$.

To take into account the view of $\mathcal{A}$ constituted from $l(s)$ different parallel executions repeated, for each of the r different message bits, the construction of a hybrid argument, a la [GM82], is straightforward.

Thus the distribution of the views of the adversary generated from the real world execution of $\gamma(p_u, p_v, N, s, c)$ is indistinguishable from the distribution of the views of the adversary generated from the ideal world simulation by the simulator.

(b) $\vec{C} \notin \mathcal{T}$ i.e., Channel simulated by β between p_u and p_v is not authentic. In this case the channel established between parties p_u and p_v by protocol $\gamma(p_u, p_v, N, s, c)$ is not secure but may be arbitrarily *tamperable*. This can be seen as follows: In Step 1, p_u may be able to send a block of uniformly chosen random bits to p_v, using the path of non-faulty nodes between p_u and p_v, which may be perfectly hidden from the adversary. However, in the subsequent steps the adversary, which has full control over the additional channel, not only receives the messages sent by p_u but has full control for the rest of the steps of the protocol. The adversary may be able to get p_u and p_v to agree on arbitrary message bit.

In this case since the message m that p_u intends to send to p_v is completely revealed to the adversary $\mathcal{A}$, we provide the simulator $\mathcal{S}$ with the message m that p_u intends to send to p_v. Once the simulator possesses the message m itself, it can simulate all the honest parties, ditto as if they were participating in real execution of protocol γ and can thus perfectly simulate the view of $\mathcal{A}$.

2. Node p_u is corrupted. Recall that only p_u is supposed to receive an input message for $\gamma(p_u, p_v, N, s, c)$. In this case p_u itself is controlled by $\mathcal{A}$. In particular, the $\mathcal{A}$ itself possesses the message that p_u intends to transmit to p_v, which can be honest or corrupted.

   First note that the simulator is only given the quadruplet $\vec{C}$ corrupted by $\mathcal{A}$ for this case. All we need to note is that the honest nodes only make uniformly random choices when/if they are required to make such choices during the execution of $\gamma(p_u, p_v, N, s, c)$. Otherwise, they are just limited to forwarding messages exchanged on the authentic channel. Thus the simulator does not need any information or message for simulating the role of the honest nodes in this case - it is the adversary who needs the message and it also possesses the message. The adversary controlled p_u may not execute the protocol as required and abort or send unexpected/arbitrary messages but all this is perfectly simulated.

3. Otherwise. This means that (a) p_u is honest and, (b) That either, (1) p_v is corrupted or, (2) There exists no path of non-faulty nodes of length at most $c * \lg_2 n$ between p_u and p_v in network N or, (3) The channel realized by β is not authentic.

   In either of the three cases *we will not be required to prove that the protocol $\gamma(p_u, p_v, N, s, c)$ realizes a secure/private channel between nodes p_u and p_v for this case*. Since, we are not required to prove that the channel realized by γ is not private it is enough for us to demonstrate a simulator that takes as input the message m that p_u intends to send to p_v. Depending on the other conditions full or no information about the message m may get leaked to the adversary just as it would in the execution of protocol $\gamma(p_u, p_v, N, s, c)$.

For this case, we demonstrate a simulator that is given as input the tuple $\{m, \vec{C}\}$, which will be sufficient for the the higher level simulator.

But construction of such a simulator is quite straightforward, as noted above, because once the simulator possesses the only input any party is given in the execution of γ, *the simulator just simulates the role of all honest parties "exactly" as in the real execution of protocol* γ, while sending the messages to be sent to the corrupt parties to the adversary and vice-versa receiving too. If the simulator makes the same random choices for the honest parties and the adversary makes the same random choices for the corrupted parties, for the ideal world simulation as well as real world execution of protocol $\gamma(p_u, p_v, N, s, c)$, then the adversary is going to end up with exactly the same view. Hence the distributions of the views of the adversary for the two cases would also be indistinguishable. Otherwise, a contradiction would result due to the existence of a distinguisher that distinguishes between two sources of uniformly chosen random bits.

□

We are going to show that protocol $\gamma(p_u, p_v, N, s, c)$ described above realizes a $\mathcal{T}$-secure channel between p_u and p_v.

Theorem C.3 *Let $N = (V, E)$ be a partially connected network with Property C.1 defined above. Then, protocol $\gamma(p_u, p_v, N, s, c)$, executed on N, realizes a $\mathcal{T}$-secure channel between p_u and p_v.*

Proof: Recall the definition of $\mathcal{T}$-secure and $\mathcal{T}$-authentic channels from section A. We shall show that channel realized by $\gamma(p_u, p_v, N, s, c)$ is secure:

1. Privacy: We demonstrate a simulator that runs in time polynomial in the running time of adversary $\mathcal{A}$, such that for every input vector $\vec{I}$, such that $\forall j \in [n], j \neq u : I[j] = \phi$ and $I[u] = m \in \{0,1\}^r$ (here m is the message to be sent from p_u to p_v), subset C of nodes corrupted by the adversary $\mathcal{A}$ such that $C \in T$, the simulator Sim takes as input the subset C, network topology N and generates a distribution of views of $\mathcal{A}$ that is indistinguishable from the distribution of views of $\mathcal{A}$ generated from real execution of $\gamma(p_u, p_v, N, s, c)$.

   The existence of such a simulator follows as a corollary to Theorem C.2. In particular we are considering case 1a, for which the simulator does not take $\vec{I}$ as the input. This completes the proof of privacy.

2. (almost) Correctness: It is given to us that Protocol $\beta(p_u, p_v, N)$ realizes a $\mathcal{T}$-authentic channel from p_u to p_v.

   We have to prove that the channel realized by $\gamma(p_u, p_v, N, s, c)$ is (almost) correct when $\mathcal{A}$ chooses to corrupt quadruplet $\vec{C} \in \mathcal{T}$. For this just observe that N possesses the property, C.1: Thus for each $\vec{C} \in \mathcal{T}$, the protocol $\gamma(p_u, p_v, N, s, c)$ is equivalent to the bit transmission protocol, B.1.

   The analysis for correctness condition is the same as that of the bit transmission protocol in Section B, with $N = w, s = l$ channels and other suitable parameters for security. From Section B, we have that the channel realized by $\gamma(p_u, p_v, N, s, c)$ between p_u and p_v is correct, except for negligible probability of error, $\mu(l)$, for security parameter l and hence almost correct.

From the above we have that multiparty protocol $\gamma(p_u, p_v, N, s, c)$ realizes a $\mathcal{T}$-secure channel between p_u and p_v.

□

How to extend the protocol and analysis to send messages of arbitrary lengths? To send message $m = (m_0, m_1, \ldots, m_o, \ldots, m_{|m|})$ (Where m_i represents the i^{th} bit of m) from node p_u to node p_v, execute Step 1 through Step 5, *parallely* for each bit, $l(s)$ times parallely.

The o^{th} bit computed in this manner is accepted as the m_o^γ bit of message m.

The analysis of the correctness condition of the protocol remains (more or less) the same. More specifically, the message is sent correctly over the channel iff each bit of the message is sent correctly. To ensure that the entire message gets sent correctly, we ensure that each bit of the message is sent correctly.

Suppose with message complexity $O(l)$ we are able to ensure that the message m is sent correctly with probability $1 - \mu(l)$, for some negligible function $\mu(.)$. The probability of making an error, while sending the entire message m over the channel is at most $1 - (1 - \mu(l))^{|m|} \leq |m| * \mu(l)$, which is also negligible. [To reduce the probability of error, back to $\mu(l)$, the number of parallel executions of Step 1 to Step 6 of the bit-transmission protocol may be increased by an additive factor of $O(\lg |m|)$ rounds.]

D Almost everywhere secure computation

In this section we shall show how to realize the notion of *almost everywhere secure computation* on a class of partially connected networks that possess special properties.

D.1 Realizing almost everywhere secure computation on partially connected networks with special properties

The proof for realizing almost everywhere secure computation on networks with special properties involves several smaller components. We shall break it into two parts: (1) Assuming unconditionally secure multiparty computation, we show how to realize almost everywhere secure computation on complete networks with parties as well as channel corruptions (2) Assuming (1), we show how to realize almost everywhere secure computation on partially connected networks with special properties.

D.1.1 Corruption of parties as well as channels

We shall prove that if there exists an unconditionally secure multiparty protocol in the vanilla model, which allows only for corruption of parties, according to Definition A.10, then there exists an unconditionally secure multiparty computation protocol executed on a complete network in the model that allows for corruption of parties as well as channels, actively as well as passively, according to Definition A.14.

Let f be an n-variate function and Π be an unconditionally secure multiparty protocol for evaluating function f as long as at least $\lfloor \frac{2*n}{3} \rfloor + 1$ parties are honest. The following claim is well known:

Claim D.1 *There exist multiparty computation protocol that securely evaluates n-variate function f according to Definition A.10.*

The three distinguishing features of Definition A.14 with respect to Definition A.10 are that in Definition A.14 : (1) Adversary $\mathcal{A}$ is allowed to corrupt parties as well as channels, passively as actively, so an adversary structure $\mathcal{T}$ is introduced, A.1 (2) The multiparty computation protocol assumes only a partially connected communication network that connects the parties. (3) Guarantees only statistical correctness and not perfect correctness.

Let N_C be a complete undirected network.

Theorem D.2 *There exists a multiparty computation protocol Π' that $\mathcal{T}$-securely evaluates function f according to Definition A.14, on a complete network N_C for all adversaries $\mathcal{A}$ restricted to a feasible adversary structures $\mathcal{T}$.*

Proof: Let Π be a multiparty protocol as per Claim D.1. We show that $\Pi' = \Pi$ executed on N_C $\mathcal{T}$-securely evaluates f, as according to Definition A.14, for all feasible adversary structures $\mathcal{T}$.

Let $\vec{C} \in \mathcal{T}$. Since $\vec{C}$ is a feasible quadruplet of corruptions, Definition A.3, then there exists a subset of honest parties $H \subset P$, $|H| \geq \lfloor \frac{2*n}{3} \rfloor + 1$, such that $\forall p_u, p_v \in H : (p_u, p_v) \notin \vec{C}[2] \bigcup \vec{C}[3]$.

We need to make sure that executing protocol Π' on N_C, while the adversary $\mathcal{A}'$ corrupts $\vec{C}$, achieves the correctness and privacy condition of Definition A.14.

For Correctness: It is enough to show that for every strategy of $\mathcal{A}'$ that corrupts quadruplet $\vec{C}$ as above, on execution of protocol Π' on N_C, there exists a corresponding $\mathcal{A}$ that corrupts parties in subset $P - H$ while parties execute Π, such that the (distribution of the) view of the honest parties is indistinguishable for the two cases. This strategy of adversary $\mathcal{A}$ is as follows: The adversary $\mathcal{A}$ internally simulates $\mathcal{A}'$ for all maliciously corrupted parties (i.e., in subset $\vec{C}[0]$), simulates the protocol for honest parties on the rest of the corrupted parties (i.e., in subset $P - H - \vec{C}[0]$) and appropriately simulates $\mathcal{A}'$ for passively and actively corrupted channels between the parties in subset H and $P - H$ and the internal channels between the parties in subset $P - H$, before the messages are (received from or) sent to the honest parties in H, while executing multiparty protocol Π.

Correctness is straightforward or contradiction is achieved about the correctness of Π, which satisfies Definition A.10 as in Claim D.1.

For Privacy: For the privacy condition it is enough to demonstrate an appropriate PPT simulator. The simulator is given the input and output values of all the parties in subset $P - H$. Clearly, by Definition A.10, there exists a simulator that simulates the view of the parties in subset $P - H$ (which is of size at most $\lfloor \frac{n}{3} \rfloor$) indistinguishably. The same simulator (slightly modified as described next) is invoked here as well. Clearly, the view generated by channel corruptions and passive corruptions of the parties, in the new case, "looks" enhanced but is just a trivial extension because the simulator is given the input and output values of all the parties in subset $P - H$.

If the distributions of views of $\mathcal{A}'$ generated by the above simulator, and the distribution generated during the real execution of protocol Π' have a non-negligible difference, then it translates to a non-negligible difference in the corresponding distributions generated for Definition A.10 - a contradiction.

Furthermore, it is easily verified that if Π is a two phase protocol as according to Definition A.10, then so is Π' as according to Definition A.14 (Basically, the function $reveal()$ translates as such, the difference is only in weakening the requirement from all honest parties being able to successfully commit to their initial input values, to at least $\lfloor \frac{2*n}{3} \rfloor + 1$ honest parties being able to commit to their initial input values, except for negligible probability). □

D.1.2 Almost everywhere secure computation

The definition of almost everywhere secure computation has been provided in the Section A on Models and Definitions. In this section we shall show how to realize this definition on networks which have certain special properties.

Now we present the main theorem of this work. The proof of this theorem is somewhat long. In general, a formal proof of a claim or a lemma has been preceded by intuitive discussion.

First, we describe a $\mathcal{T}_{\beta_N,c}$-Communicability Property of an undirected network $N = (V, E)$. Essentially, it says that for every choice of quadruplet of corruptions $\vec{C} \in \mathcal{T}$, there exists a large enough number of honest nodes that can communicate securely with each other, so that secure multiparty computation can be carried out. This is formalized as follows:

Definition D.3 *Let $\mathcal{A}$ be a $\mathcal{T}$-limited adversary, as defined in A.1. A partially connected network N has a $\mathcal{T}_{\beta_N,c}$-Communicability Property, iff the following condition holds true:*

$\forall \vec{C} \in \mathcal{T} : \exists H \subset V : |H| \geq \lfloor \frac{2*n}{3} \rfloor + 1 : \forall p_u, p_v \in H : [(p_u, p_v) \notin E \bigcup E_d \rightarrow$ *Network N possesses Property $\{\vec{C}\}_{p_u,p_v,\beta_N,c}$, C.1].*

The next theorem says that if N possesses, $\mathcal{T}_{\beta_N,c}$-communicability property, described above, then there exists a secure multiparty computation protocol Π_N that $\mathcal{T}$-securely evaluates f on N:

Theorem D.4 *If network N possesses $\mathcal{T}_{\beta_N,c}$-Communicability Property, D.3, then there exists a multiparty computation protocol Π_N, that $\mathcal{T}$-securely evaluates function f, on network N, as according to Definition A.14.*

Proof: The proof of this theorem is somewhat long and delicate for the primary reason that we need a protocol to realize secure channels between distant nodes of the partially connected network N, using the infrastructure of N, but are not allowed to use any composition theorems (Please refer to discussion in the section on Model and Definitions for this).

The higher level construction of the protocol and the proof of its security proceed as follows:

1. Let $C_N = (V, E, E_{C_N})$ be the network on vertex set V constructed by connecting all the pairs of parties not already connected in N, by *virtual* edges (Note, that since the virtual edges are directed, so for each $(u, v) \in V * V - E$ two directed edges are added to E_{C_N}).

   We fix an order on all the (directed) virtual edges, in subset E_{C_N}, and proceed by induction as follows:

   (a) Base case: We start by describing an adversary structure $\mathcal{T}_{C_N}$ for a complete network C_N as a function of adversary structure $\mathcal{T}$, network N, and $\mathcal{T}_{\beta_N,c}$-communicability property, D.3.
   Then, we show that there exists a multiparty protocol Π_{C_N}, A.13, to be executed on the complete network C_N, that $\mathcal{T}_{C_N}$-securely evaluates f, as according to Definition A.14.

(b) By induction hypothesis, we are given an intermediate network N_i for which we are given a multiparty protocol Π_{N_i}, such that protocol Π_{N_i}, $\mathcal{T}_{N_i}$-securely evaluates f, on network N_i, as according to Definition A.14.

We are required to give a construction of multiparty protocol $\Pi_{N_{i+1}}$ to be executed on network N_{i+1} (where network N_{i+1} is network N_i with the i^{th} virtual edge e_i removed from the network N_i) and adversary structure $\mathcal{T}_{N_{i+1}}$ (which is basically adversary structure $\mathcal{T}_{N_i}$ except that edge e_i is not present in adversary structure $\mathcal{T}_{N_{i+1}}$ because there is no e_i in N_{i+1}), such that multiparty protocol $\Pi_{N_{i+1}}$ executed on network N_{i+1}, $\mathcal{T}_{N_{i+1}}$-securely evaluates function f, as according to Definition A.14.

(c) Lastly, note that inductive hypothesis will be applied till there are no virtual edges in the network that can be replaced by simulated channels, i.e., till $N_j = N$ and $\mathcal{T}_{N_j} = \mathcal{T}$, the original adversary structure which allows network N to possess certain communicability property. The multiparty protocol $\Pi_N = \Pi_{N_j}$ will $\mathcal{T}$-securely evaluate function f on network N, as according to Definition A.14.

Basically, at every inductive step we shall replace a virtual channel, by a simulated channel that does precisely the same function[5] using the infrastructure of network N i.e., is *authentic* if the original corruption makes it *authentic*, is *secure* if the original corruption is *secure* and is *tamperable* if the original edge is *tamperable*. Loosely speaking, the I. H. says that if we can realize $\mathcal{T}_{N_i}$-secure computation on partially connected network N_i with r real, v virtual and s simulated channel, then we can realize $\mathcal{T}_{N_{i+1}}$-secure computation on network N_{i+1} with r real, $v-1$ virtual and $s+1$ simulated edges (where adversary structure $\mathcal{T}_{N_{i+1}}$ is identical to $\mathcal{T}_{N_i}$, except that the *virtual* edge replaced in the last step, is not part of adversary structure $\mathcal{T}_{N_{i+1}}$ for it does not exist in network N_{i+1}).

Following will be the outline of the rest of the proof:

Outline of the proof:

1. Ordering of edges: Next, we shall define an ordering of the missing edges of the network N.
2. Round structuring protocols
3. Construction of the sequence of networks N_i
4. Adversarial structures $\mathcal{T}_{N_i}$
5. Statement and Proof of the Base Case
6. Inductive hypothesis discussion and formalization
7. Proof of Inductive hypothesis

[5] Actually, this is *almost precisely*, because with negligible probability the $\mathcal{T}'$-secure channel may fail to deliver the message and the originally perfect edge is replaced by a statistically perfect edge.

An ordering of *virtual channels*: Consider the set of edges, $\overline{E_d}$ which is null for the network N, $\overline{E_d} = \{(p_u, p_v) | \forall u, v : (p_u, p_v) \notin E\}$. Consider an arbitrary ordering over this set of virtual edges, $e_0 = \phi$, $e_1 = (p_{u,0}, p_{v,0}), e_2 = (p_{u_1}, p_{v_1}), \ldots, e_l = (p_{u_l}, p_{v_l})$. Note, that the original network N contains only the *real* edges (which are all undirected, btw).
Construction of network $N_i, \forall i \in \{0, \ldots, l-1\}$: Network $N_0 = C_N$. Network N_i is defined as follows: $N_i = (V, E, E_i)$, where $V = P$, $E_i = E_{i-1} - e_i$.
Adversary structure $\mathcal{T}_{N_i}$: The intuition behind the construction of adversary structures for each i is as follows:

Let $\gamma(p_u, p_v, N, s, c)$ be a multiparty protocol used to establish a channel from p_u to p_v for $(p_u, p_v) \notin E_i$ using the infrastructure of network N. Depending on the choice of $\vec{C}$ corrupted by the adversary, the channel established by $\gamma(p_u, p_v, N, s, c)$ may be (almost) secure, (almost) authentic or tamperable.

Thus, the pair $(\gamma(p_u, p_v, N, s, c), \vec{C})$ essentially determines the type of channel established between p_u and p_v on network N. An alternative way to look at this is to let edge (p_u, p_v) always exist in the network, but let the adversary structure be enhanced depending on the type of channel realized by $\gamma(p_u, p_v, N, s, c)$ from p_u to p_v. So, if on corruption of quadruplet $\vec{C}$, $\gamma(p_u, p_v, N, s, c)$ realizes an (almost) secure channel, then the new adversary structure contains the same $\vec{C}$. However, if on corruption $\vec{C}$, $\gamma(p_u, p_v, N, s, c)$ realizes an (almost) authentic channel from p_u to p_v, then this is equivalent to assuming a secure channel between p_u and p_v, passively corrupted by $\mathcal{A}$ i.e., in the new adversary structure $\vec{C}$ is enhanced to include (p_u, p_v) i.e., $(p_u, p_v) \in \vec{C}'[2]$ in the new adversary structure (p_u, p_v). If the channel realized by $\gamma(p_u, p_v, N, s, c)$ is tamperable, then (p_u, p_v) is added to $\vec{C}[3]$, for each $\vec{C}$. This is how the adversary structure is constructed for the base case network C_N, which has all the virtual edges, starting from the adversary structure $\mathcal{T}$ for network N.

Now, we work backwards: The adversary structure $\mathcal{T}_{N_i}$, for network N_i is constructed from the adversary structure $\mathcal{T}_{N_{i-1}}$ for network N_{i-1} by just removing the channel e_i from $\vec{C}[2]$ or $\vec{C}[3]$ for each $vecC \in \mathcal{T}_{N_{i-1}}$.
Round restructuring: Our goal is to adapt a multiparty computation protocol that satisfies Definition A.14, for a complete network to a multiparty computation protocol that is to be executed on a partially connected network, with special properties, that satisfies Definition A.14.

Even if we assume that there exists a multiparty protocol that establishes (almost) secure channel between some pair of nodes of the partially connected network there arises the following *synchronization* issue with respect to using it for realizing secure function evaluation: Suppose it takes α rounds to send a message from party p_u to party p_v along a channel (realized through may be physical infrastructure) of the partially connected network. To send messages between distant nodes of the network, multiparty protocol $\gamma(p_u, p_v, N, s, c)$ is executed for this purpose and may run in $f_\gamma(\alpha, N)$ rounds, for some function f, to deliver a message from party p_u to party p_v.

Consider the multiparty protocol Π_{ec} from Theorem D.2, which is restructured to run on the complete network, C_N described above, as follows: Each Step of protocol Π_{ec} is now expanded to $f_\gamma(\alpha, N)$ rounds. If in protocol Π_{ec} the message was delivered between two parties in one round, now it is delivered in α rounds or $f_\gamma(\alpha, N)$ rounds depending on whether the message is sent along a *real channel* or a *virtual channel*. This is explained below:

Definition D.5 *For a real channel, $F^r(S, R, edge - id)$, $r = \alpha$.*

For a virtual channel, $F^r(S, R, edge - id)$, $r = f_\gamma(\alpha, N)$, for some function f_γ (which will roughly be the maximum number of rounds taken by multiparty protocol γ over all pairs of p_u,p_v for network N).

Base case for network $N_0 = C_N$: For the base case we have the complete network N_0. As discussed above, all the edges in N_0 which are also present in partially connected network N denote *real* channels, while the rest denote *virtual* channels.

The adversary $\mathcal{A}$ is restricted to adversary structure $\mathcal{T}_{C_N}$, as described above.

We are given that network N possesses the $\mathcal{T}_{\beta_N,c}$-Communicability Property, D.3. Because of the Communicability Property D.3 of network N, the adversary structure $\mathcal{T}_{C_N}$ is feasible.

By Theorem D.2, there exists a multiparty protocol Π_{ec}, that $\mathcal{T}_{C_N}$-securely evaluates function f, on network C_N, as according to Definition A.14, for every feasible adversary structure $\mathcal{T}_{C_N}$.

We adapt multiparty protocol Π_{ec} to the round structure D.1.2, as described above[6]. It follows that the new multiparty protocol Π_{N_0}, executed on N_0 (with round structure, D.5), $\mathcal{T}_{N_0}$-securely evaluates function f, as according to Definition A.14 (where $\mathcal{T}_{N_0} = \mathcal{T}_{C_N}$).

This completes the proof for the Base Case.

Induction Hypothesis. Let Π^i be a multiparty protocol, to be executed on network N_i, that $\mathcal{T}_{N_i}$-securely evaluates f, as according to Definition A.14.

Then, there exists a multiparty protocol Π^{i+1}, to be executed on network N_{i+1}, that $\mathcal{T}_{N_{i+1}}$-securely evaluates function f, as according to Definition A.14 (where construction of network N_{i+1} and adversary structure $\mathcal{T}_{N_{i+1}}$ has been discussed above).

Proof of Inductive Hypothesis: First note that the only difference between network N_i and N_{i+1} is that in N_i there is a virtual edge/channel $e_i = (p_u, p_v)$ between p_u and p_v, which is not present in network N_{i+1}. Similarly, the only difference in adversary structure $\mathcal{T}_{N_i}$ and $\mathcal{T}_{N_{i+1}}$ is that edge e_i is not *passively* or *actively* corrupted in adversary structure $\mathcal{T}_{N_{i+1}}$ (as it does not even exist in N_{i+1}).

We shall show how to adapt Π^i to multiparty protocol Π^{i+1} to be executed on N_{i+1}.

Description of Π^{i+1}: The multiparty protocol Π^{i+1} to be executed on N_{i+1} is exactly the same as Π^i except for the following difference: In Π^i when some message m is sent from party p_u to p_v along the virtual channel (p_u, p_v) of N_i, while in Π^{i+1} the message is sent from p_u to p_v, by execution of $\gamma(p_u, p_v, N, s, c)$ on network N. (It is easily verified that both protocols Π^i and Π^{i+1} follow the round structure, D.1.2).

We need to only make sure that the new protocol Π^{i+1} satisfies the correctness and privacy condition, as according to Definition A.14[7].

Claim D.6 *Multiparty protocol Π^{i+1}, to be executed on network N_{i+1}, $\mathcal{T}_{N_{i+1}}$-securely evaluates f, as according to Definition A.14.*

Proof: Fix any quadruplet $\vec{C}_{i+1} \in \mathcal{T}_{N_{i+1}}$ corrupted by $\mathcal{A}_{i+1}$. We shall prove that execution of multiparty protocol Π^{i+1}, achieves the correctness and privacy condition, Definition A.14. For this we invoke the Inductive Hypothesis.

[6]Basically, this implies that for any execution of multiparty protocol, following such a round structure D.1.2, on network N, a message is transmitted on virtual channel from party p_u to party p_v only in round $\equiv 0 \bmod f_\gamma(\alpha, N)^{th}$ and reaches the other party in next round $\equiv 0 \bmod f_\gamma(\alpha, N)^{th}$

[7]Here the correctness condition includes both the correctness of input commitment phase, as well as the evaluation phase

First, let $\vec{C}_i$ corrupted by $\mathcal{A}_i$ be defined from $\vec{C}_{i+1}$ as follows: $\vec{C}_i[0] = \vec{C}_{i+1}[0], \vec{C}_i[1] = \vec{C}_{i+1}[1]$.

If $\gamma(p_u, p_v, N, s, c)$ simulates an authentic channel between p_u and p_v for corrupted quadruplet $\vec{C}_i$, then $\vec{C}_i[2] = \vec{C}_{i+1}[2] \bigcup e_i$, else if $\gamma(p_u, p_v, N, s, c)$ simulates a tamperable channel between p_u and p_v, then $\vec{C}_i[3] = \vec{C}_{i+1}[3] \bigcup e_i$, else $\vec{C}_i[2]$ and $\vec{C}_i[3]$ are identical to $\vec{C}_{i+1}[2]$ and $\vec{C}_{i+1}[3]$, respectively. Clearly, $\vec{C}_i \in \mathcal{T}_{N_i}$.

1. Correctness: For the correctness condition we demonstrate an adversary $\mathcal{A}_i$ that attacks MPC protocol Π^i's execution on network N_i, that corrupts $\vec{C}_i$, such that for the same initial vector of inputs (and uniformly chosen randomness) given to the honest parties and the adversary - the distribution of outputs of the honest parties $H_{i+1} = H_i$ (where H_i is as determined according to the definition A.14) would be indistinguishable[8] from the case when all the parties were executing multiparty protocol Π^{i+1} on N_{i+1} with adversary $\mathcal{A}_{i+1}$ corrupting $\vec{C}_{i+1}$.

   This would prove the correctness condition.

   Consider the following $\mathcal{A}_i$ that attacks protocol Π^i: $\mathcal{A}_i$ internally simulates $\mathcal{A}_{i+1}$ by providing it all the messages that corrupted parties receive from the honest parties during the execution of protocol Π^i and sending messages generated by it to the appropriate honest parties (which are all simulated by $\mathcal{A}_i$). It can be seen that this can be done since the topology of the networks N_i and N_{i+1} are almost identical except for one difference: In network N_i there is a channel (p_u, p_v) between p_u and p_v (which may be secure, authentic or tamperable), while for network N_{i+1}, the same channel is simulated by γ to send a message from p_u to p_v. We shall show a perfect internal simulation of $\mathcal{A}_{i+1}$ by adversary $\mathcal{A}_i$, taking into account this difference.

   We shall prove the following two conditions are simultaneously true by induction on the number of Steps of MPC protocol Π^i, Π^{i+1} executed:[9]

   (a) *The view of adversary $\mathcal{A}_{i+1}$, internally simulated by adversary $\mathcal{A}_i$ attacking protocol Π^i's execution is indistinguishable from the view of $\mathcal{A}_{i+1}$ attacking protocol Π^{i+1} execution after every step.*

   (b) *The view of the honest parties H_i, generated from execution of protocol Π^i is indistinguishable from the view of the honest parties H_{i+1} generated from execution of protocol Π^{i+1} after every step.*

   To prove this let us assume that the above conditions are true up to execution of Step j of protocol Π^i,Π^{i+1} i.e., $View_{H_i,\Pi^i,j} \approx View_{H_{i+1},\Pi^{i+1},j}$[10] and $View_{\mathcal{A}_i,\Pi^i,j} \approx View_{\mathcal{A}_{i+1},\Pi^{i+1},j}$. Then we shall prove that the above conditions hold true even after the execution of Step $j+1$ i.e.$View_{H_i,\Pi^i,j+1} \approx View_{H_{i+1},\Pi^{i+1},j+1}$ and $View_{\mathcal{A},\Pi^i,j+1} \approx View_{\mathcal{A},\Pi^{i+1},j+1}$.

[8]Note, this indistinguishability would only be statistical because the channel simulated by γ are (almost) secure and not perfectly secure, replacing a perfectly secure channel by an (almost) secure channel would allow negligible probability of error in correctness conditions.

[9]The reader may recall the distinction on Step and rounds previously made, D.5 i.e., each Step consists of $f_\gamma(\alpha, N)$ rounds.

[10]As noted above $H_{i+1} = H_i$

Internal Simulation of adversary $\mathcal{A}_{i+1}$ by $\mathcal{A}_i$: First we demonstrate the internal simulation of adversary $\mathcal{A}_{i+1}$ by $\mathcal{A}_i$. All the messages sent to adversary $\mathcal{A}_i$ controlled parties by other parties are forwarded by $\mathcal{A}_i$ to $\mathcal{A}_{i+1}$ as it is. In addition, $\mathcal{A}_i$ runs the simulator for protocol $\gamma(p_u, p_v, N, s, c)$ demonstrated in C.2 with $\mathcal{A}_{i+1}$. It has been noted in Theorem C.2 that this simulator does not take any input till the last and final step of protocol $\gamma(p_u, p_v, N, s, c)$, when the actual message is hidden by random bits and sent from p_u to p_v along the additional channel. So up till the last step the simulation can be carried by $\mathcal{A}'$ without any issues. For the simulation of the last step of $\gamma(p_u, p_v, N, s, c)$ we have the following cases: The virtual edge (p_u, p_v) is *passively* or *actively* corrupted by the adversary $\mathcal{A}_{i+1}$ or the virtual edge (p_u, p_v) is secure. In case the (p_u, p_v) is corrupted, in anyway, then the adversary $\mathcal{A}_{i+1}$ would have received the message sent to channel (p_u, p_v), after the first round of this Step itself (Please refer to the definitions of various channels in the Second section) and so the simulation can be carried out for the rest of the protocol $\gamma(p_u, p_v, N, s, c)$. (Of course, this message that the corrupted channel (p_u, p_v) receives i.e., is received by $\mathcal{A}_i$ in the second round is not revealed to adversary $\mathcal{A}_{i+1}$ till it is necessary to do so during the simulation of protocol $\gamma(p_u, p_v, N, s, c)$.) In case the virtual channel (p_u, p_v) is (almost) secure[11], but party p_v is corrupted, then too adversary $\mathcal{A}_{i+1}$ would have received the message from (p_u, p_v) at the end of the second round of this Step. Hence for all cases the adversary $\mathcal{A}_{i+1}$ can successfully carry the simulation of $\mathcal{A}_i$ for protocol $\gamma(p_u, p_v, N, s, c)$.

Now let us verify that the view of the honest parties of subset $H_{i+1} = H_i$ and the adversary $\mathcal{A}_{i+1}$ after Step $j+1$ of protocol Π^i and Π^{i+1}.

We have the following cases to analyze:

p_u is an honest node: First, review the important feature D.1.2 of MPC protocol Π^{i+1} described above: The message m to be sent by honest party p_u to party p_v in Step $j+1$ by execution of γ is fixed at the end of Step j. Since $(View_{H_i,\Pi^i,j}, View_{\mathcal{A},\Pi^i,j}) \approx (View_{H_i,\Pi^{i+1},j}, View_{\mathcal{A},\Pi^{i+1},j})$, therefore message m to be sent from node p_u to p_v during Step $j+1$ for protocol Π^i and Π^{i+1} is also indistinguishable.

Protocol Π^{i+1} is just protocol Π^i and γ executed simultaneously but which do not influence each other i.e., the view of the honest parties generated from one doesn't influence the view of the honest parties generated from another (once the message m to be sent is fixed at the beginning of Step $j+1$).

For this case, we have the following sub-cases:

(a) Virtual channel (p_u, p_v) is corrupted. For this case the simulator of protocol γ, C.2, executed by adversary $\mathcal{A}'$ doesn't need the message m till step 5 of γ (much more than $c * \lg_n$ rounds). However the message m is received by the adversary $\mathcal{A}_i$ in the first round of Step $j+1$ from p_u.

[11] This is just modeling the case of when the adversary receives the message for otherwise it is meaningless to have a secure channel with p_v corrupted

The views of the honest parties $H_{i+1} = H_i$, constituted from execution of $\Pi^i \subset \Pi^{i+1}$ (i.e., excepting execution of $\gamma(p_u, p_v, ..)$) and adversary $\mathcal{A}_{i+1}$ are indistinguishable after every round for both N_i and N_{i+1}. The view of adversary $\mathcal{A}_{i+1}$ is indistinguishably simulated by $\mathcal{A}_i$ because $\mathcal{A}_i$ indistinguishably simulates γ and forwards all messages received from other honest parties and hence outputs indistinguishable messages to be sent to honest parties. Honest parties view in turn is indistinguishably augmented for networks N_i and N_{i+1} (excluding execution of γ) which output indistinguishable messages to be sent in the next round. Moreover since at the last round of Step $j+1$ the honest parties have deleted all extra messages (and random blocks etc.) from execution of γ the views of the honest parties are indistinguishable for Π^i executed on N_i and Π^{i+1} executed on N_{i+1}.

(b) Or else party p_v is corrupted. In this case the adversary $\mathcal{A}_i$ receives the message m from channel (p_u, p_v) in the second round of Step $j+1$, and can hence carry the internal simulation of $\mathcal{A}$ as required. The rest of the analysis of this case is identical to the case above.

(c) Or else p_u is honest, and (p_u, p_v) is a secure virtual channel. In this case the protocol γ sends message m from node p_u to node p_v perfectly securely on network N. Thus the adversary $\mathcal{A}'$ can simulate the view of adversary $\mathcal{A}$ perfectly without access to message m at all. The rest of the analysis for this case is identical to the first case.

p_u is corrupted. In this case the adversary $\mathcal{A}_{i+1}$ attacking Π^{i+1} has large number of rounds (and, flexibility) before which it must commit to a message to be sent to node p_v. It is also possible that the message committed or sent to node p_v is completely bogus. The adversary $\mathcal{A}_i$ shall extract the message it intends to send to p_v by internally simulating $\mathcal{A}_{i+1}$ and, then sends the message to the virtual channel (p_u, p_v) (or if it too is corrupted, then to node p_v).

For this the argument is as follows: The view of $\mathcal{A}_{i+1}$ and the honest parties in subset H_{i+1} is indistinguishable after each round of Step $j+1$ for Π^i's execution on network N_i, or Π^{i+1}'s execution on network N_{i+1}. After some rounds (the number of which may depend on how the adversary $\mathcal{A}_{i+1}$ behaves for corrupt node p_u), the adversary $\mathcal{A}_i$ is successful in extracting the message that $\mathcal{A}_{i+1}$ intends to send to node p_v. Then it sends the extracted message to virtual channel (p_u, p_v) or directly to node p_v if virtual (p_u, p_v) is also corrupted, in the following round. (Note, each honest party has computed the message it intends to send to other parties at the beginning of the Step and this remains unchanged by whatever messages it receives during the Step.)

The same message is also extracted by party p_v during execution of γ of Π^{i+1}. At the end of Step $j+1$ all the extra messages, due to the execution of γ are deleted by the honest nodes - leaving a clean state - this is the last step of the protocol γ. Thus, for this case also the distribution of views of the honest parties and the adversary $\mathcal{A}_{i+1}$ are indistinguishable after Step $j+1$. This completes the proof of Induction Hypothesis.

Putting together, we have that the distribution views of the set of honest parties H_i when Π^i is executed on network N_i with $\mathcal{A}_{i+1}$ attacking the execution, is indistinguishable to the (distribution) of views of honest parties H_i, when Π^{i+1} is executed on network N_{i+1} with $\mathcal{A}_{i+1}$ attacking the execution.

Hence, the distribution of outputs generated by the honest parties $H_{i+1} = H_i$ for the two cases are indistinguishable. Thus, if Π^i satisfies the correctness conditions, as described in definition A.14, then so does Π^{i+1}.

2. Privacy: For this we demonstrate a simulator that simulates the views of $\mathcal{A}_{i+1}$, attacking Π^{i+1}'s execution on network N_{i+1}, such that the distribution of views of $\mathcal{A}_{i+1}$ generated by the simulator is indistinguishable from the distribution of the views of $\mathcal{A}_{i+1}$ generated from real execution of protocol Π^{i+1}.

   We shall employ induction hypothesis for the following adversary $\mathcal{A}_i$, as described in the Correctness condition proved above, that attacks Π^i, while internally simulating $\mathcal{A}_{i+1}$. By induction hypothesis, there exists a simulator Sim_i that takes as input the identities, input and output values of the parties in subset $P - H_i$, adversary program $\mathcal{A}_i$, network N_i, $\vec{C}_i$ and generates a view (or more specifically distribution of views) of $\mathcal{A}_i$ that is indistinguishable from the view (or specifically distribution of views) of $\mathcal{A}_i$ generated from real execution of Π^i on network N_i with same parameters. Let Sim_i be such a simulator.

   Simulator Sim_{i+1} for $N_{i+1}, \Pi^{i+1}, \mathcal{A}_{i+1}$ simulates adversary $\mathcal{A}_{i+1}$ as follows: First note that the simulator Sim_{i+1} is given the same set of input and output values of party set $P - H_{i+1} = P - H_i$ as Sim_i (except that $\vec{C}_i$ may contain the virtual channel (p_u, p_v) unlike $\vec{C}_{i+1}$).

   In general at the end of an execution, adversary $\mathcal{A}_i$ outputs its input tape and read tapes as the output and this is considered as the output of the adversary. In the adversary program $\mathcal{A}_i$ described above, $\mathcal{A}_i$ (indistinguishably) internally simulates $\mathcal{A}_{i+1}$ for generating the messages to be sent to the honest parties etc.. We make another enhancement to this adversary program: On termination $\mathcal{A}_i$ outputs, whatever is outputted by $\mathcal{A}_{i+1}$ (instead of its own read tapes, input tapes). Since, we have shown that in the proof of the correctness condition above, $\mathcal{A}_i$ perfectly simulates the view of $\mathcal{A}_{i+1}$ and, hence simulator Sim_i is going to output the view of $\mathcal{A}_{i+1}$, generated in the process, whose distribution will be perfectly indistinguishable from the distribution of views of $\mathcal{A}^{i+1}$ generated from real execution of Π^{i+1}. (The picture looks like this: Earlier $Sim_i \leftrightarrow \mathcal{A}_i(\mathcal{A}_{i+1})$ and now $Sim_{i+1} = Sim_i(\mathcal{A}') \leftrightarrow \mathcal{A}$.)

   The proof of correctness of simulator Sim_{i+1} follows from the proof of correctness of simulator Sim_i, and the proof of correctness of internal simulation of adversary $\mathcal{A}_{i+1}$ by $\mathcal{A}_i$ given above, otherwise it would contradict the correctness of either of them.

This proves the induction hypothesis. □

The I. H. is invoked only till there exists a virtual channel in the network that can be replaced by a simulated channel. So, successive applications of I. H. results in network $N_l = N$ which is the original partially connected network N under consideration.

Putting together with the base case, we have that if N is a network, that possesses $\mathcal{T}_{\beta_N,c}$-Communicability Property, D.3, then there exists a multiparty computation protocol Π_N, to be executed on N, that $\mathcal{T}$-securely evaluates function f, as according to Definition A.14. □

E Almost everywhere secure computation on some specific classes of networks

In this section we shall describe a class of partially connected networks and associated adversary structures $\mathcal{T}$, for which the partially connected networks possess $\mathcal{T}_{\beta_N,c}$-Communicability Property, as described in Section D.3. Using the Main Theorem D.4, we shall obtain multiparty protocols that $\mathcal{T}$-securely evaluates (an arbitrary n-ary) function f on the corresponding network N.

We shall consider the (infinite) families of partially connected networks previously considered by [DPPU88], [Upf92] for the almost everywhere agreement problem.

1. In [Upf92], a protocol was given that achieves $O(t)$-agreement, on a class of constant degree expander graphs, namely LPS expanders. The following Theorem E.1 is a (minor) restatement of theorem from [Upf92].

   Theorem E.1 *There exists*

   (a) Constant $\alpha > 0$, μ, q and d, independent of t and n;

   (b) An n-vertex d-regular network G, which can be explicitly constructed;

   (c) A multiparty communication protocol $\beta(p_u, p_v, N)$ with round complexity α.

   Such that for any subset $\tilde{C}$, $|\tilde{C}| \leq k * n$ for some constant k, of faulty nodes in G, communication protocol $\beta(p_u, p_v, N)$ there exists a subset of non-faulty nodes H, where $|H| \geq n - \mu * |\tilde{C}|$, such that $\beta(p_u, p_v, N)$ establishes $\{(\tilde{C}, \phi, \phi, \phi)\}$-authentic communication channel between p_u and p_v. Furthermore, β requires at most $q * \lg_2 n$ communication rounds.

   Two observations can be made from Theorem E.1 and its proof:

   (a) There exists a path of honest nodes between every pair of parties u, v from subset H of length at most $d * \lg_2 n$.

   (b) There exists a multiparty protocol $\beta(p_u, p_v, N)$ that realizes $\{\tilde{C}\}$-authentic channel between parties p_u and p_v for $\forall p_u, p_v \in H$.

   Let $\epsilon = sup_{\{\chi | n - \mu * \chi * n \geq \lfloor \frac{2*n}{3} \rfloor + 1\}}$. Define the following adversary structure: $\mathcal{T} = \{(\tilde{C}, \phi, \phi, \phi) | \tilde{C} \subset V, |\tilde{C}| \leq \epsilon * n\}$.

   We have the following corollary to Theorem E.1:

   Corollary E.2 *Let N be a network from above Theorem E.1. Then N possesses the $\mathcal{T}_{\beta,N}$-Communicability Property D.3, for adversary structure $\mathcal{T}$.*

Putting together Corollary E.2 with Theorem D.4, we have that:

Theorem E.3 *There exists a multiparty protocol Π to be executed on network N (LPS expander with n nodes, as in E.1) which $\mathcal{T}$-securely evaluates function f, as according to Definition A.14.*

Note that since the algorithm of [Upf92], for establishing authentic channel between distant nodes of the network, run in super-polynomial time, therefore in the above multiparty protocol honest parties also run in super-polynomial time.

2. Consider the recursive construction of committee networks of unbounded degree as defined in [DPPU88]. The description for one level of recursive construction is as follows: V contains n^{ϵ} committees, each committee further consists of $m = n^{\epsilon}$ sub-committees (so on and so forth for $1/\epsilon$ recursive steps). Inside, each committee all the sub-committees form a clique i.e., every two sub-committees A_i and A_j are connected. Each such connection represents a meta-edge between some matching of nodes of the sub-committees A_i and A_j. For more details the reader is referred to [DPPU88].

   Let G_{ϵ} be a committee graph from [DPPU88] with n nodes. The following theorem was proved in [DPPU88] for network G_{ϵ}:

   Theorem E.4 *For every $\epsilon > 0$ there exists a constant $c = c(\epsilon)$, graphs $G_{\epsilon} = (V, E)$, $|V| = n$ of degree $O(n^{\epsilon})$ and t-resilient $O(t)$-agreement protocol $t \leq c * n$.*

   As a corollary to the proof of Theorem E.4, from [DPPU88], we have the following:

   Corollary E.5 *There exists a constant d, and a family of multiparty protocols $\beta(p_u, p_v, N)$, to be executed on G, such that for every subset of nodes $\tilde{C} \subset V, |\tilde{C}| \leq c * n$ corrupted by $\mathcal{A}$, there exists a subset S of honest nodes $S \subset V$ such that,*

   (a) $|S| \geq n - O(t)$

   (b) There exists a path of non-faulty nodes of length at most $d * \lg_2 n$ between p_u, p_v for all $p_u, p_v \in S$.

   (c) For all $p_u, p_v \in S$ multiparty protocol $\beta(p_u, p_v, N)$ realizes $\{\{tildeC, \phi, \phi, \phi\}\}$-authentic channel between p_u and p_v.

   Define the following adversary structure: $\mathcal{T} = \{\vec{C} | \vec{C}[0] \subset V, |\vec{C}[0] \bigcup \vec{C}[1]| \leq \epsilon * n, \vec{C}[1] = \vec{C}[2] = \vec{C}[3] = \phi\}$, for network G, which basically captures all subsets, of passive and active corruptions of parties, of size at most $\epsilon * n$, such that $n - \mu * \epsilon * n \geq \lfloor \frac{2*n}{3} \rfloor + 1$.

   We have the following corollary to the above Corollary E.5:

Corollary E.6 *Let N_ϵ be the committee graph of n nodes and $\mathcal{T}$ be an adversary structure as defined above. Network N possesses $\mathcal{T}_{\beta,N}$-Communicability Property D.3, for adversary structure $\mathcal{T}$.*

Putting together with Theorem D.4, we have:

Theorem E.7 *There exists a multiparty protocol Π which $\mathcal{T}$-securely evaluates function f on committee graph N_ϵ with n parties, for adversary structure $\mathcal{T}$ as defined above, as according to Definition A.14.*

3. Results of similar nature, as described above, can be stated for butterfly graphs and constant degree expanders, which can tolerate $\frac{n}{\lg_2 n}$ and n^δ corrupted nodes respectively, using the results in [DPPU88].