Grades of Trust in Multiparty Computation*

Jaskaran V. Singh and Nicholas J. Hopper

University of Minnesota,
Minneapolis, MN, USA
singh882@umn.edu
hopper-nj@cs.umn.edu

Abstract. Secure Multiparty Computation involves a protocol between parties with an aim to produce a computed result just as a Trust Party would produce if the parties provided their inputs to it. The Trusted party in conventional computation is replaced by "un-trusted" parties in Secure Multiparty Computation. We first show that this existing Binary definition of Trust is inadequate. Real world is rife with disparities, that which produce a perceivable trust gradient between the participants. Conventional MPC models do not take this into account and rather provide security guarantees based on the threshold of corrupted parties. The thresholds are supposed to cover for some of the parties turning out to be corrupt. Often, with the knowledge of *prior probability* of a party being corrupt, we can do better if we allot weight to each party based on how trusted we perceive it to be. Our paper explores this idea and our contributions towards it are two folds. First, we introduce the Graded Trust model where each party essentially has a *Trust Grade* assigned to it in the protocol based on the prior of it being corrupt.Then, we present a discussion on the philosophy behind graded trust, and the potential benefits for large scale public MPC systems.

Keywords: Multiparty Computation, Trust, Player Simulation

1 Introduction

Secure Multiparty Computation (MPC or SMPC) is a class of protocols that allows participants to compute a function jointly on their respective inputs and obtain an output without revealing the inputs itself.

An MPC solution to a problem is required to have a set of certain properties:- (1) *privacy* (only the end result should be revealed), (2) *correctness* (output generated at the end of the protocol should be correct, at least for the honest parties), (3) *fairness* (output is revealed to the corrupted parties iff output is revealed to honest parties), (4) *independence of inputs* (inputs from any pair of corrupted party and an honest party should be independent of each other), (5) *guaranteed output delivery* (corrupted parties cannot block honest parties from receiving output).

* This research has been funded by the National Science Foundation (Grant No. 1814753 and 1815757)

[Yao86] introduced the world to the general notion of a secure function. [GMW87] presented a cryptographic secure protocol (adversary is considered polynomial bounded) played between N parties with the Adversary controlling τ parties, that tolerates $\tau < N/2$ passively corrupt parties and $\tau < N/3$ actively corrupt parties. Furthermore, [BOGW88] provided a protocol for achieving perfect security (that is, the protocol considers information theoretic setting) with the same bounds for number of corrupted parties as [GMW87]. This conclusion was also independently reached by [CCD88].

MPC has been shown to have potential application in many popular areas of interest like Auctions [NPS99], Machine learning [LJLA17,GSB+17,MZ17] [NWI+13], Genomic Data Processing [CWB18,AAHMA16], contact discovery [LYCL13], satellite collision prevention [HLOW16]. MPC has also been successfully Deployed for Danish sugar beets auction [BCD+09], Boston women's workforce council wage equity study[BLV17], student success [BKK+] and Key Management [ABL+18].

Large Scale Public MPC. In the recent times, there has been research [CGH+18,CGG+,BHZ+18,BGG+20] exploring to use MPC in a setting involving large number of participants, potentially as a collective societal computation for common good. Such a system by design ensures that the door of participation always remains open. That is, parties are free to join and leave the computation as they wish. Barriers are kept low as to ensure more participation. Systems such as Fluid MPC [CGG+] that provide threshold based security guarantees, are vulnerable to a *Sybil attack* [Dou02] − the adversary spawning enough participants to breach a certain threshold and compromise the security guarantees it provides. Recent trends in MPC research hint towards a lot of interest being generated in large scale participation models but any discussion on regulating the effect of potentially adversarial participants has been absent. The challenge we face is to find out if we can do better than this in a setting where (1) we are sure that a one or multiple parties are more likely to be corrupt than others, and (2) we either don't want to reject participation of such parties by design (such as in case of large scale MPC).

1.1 Trust

The common way to describe a Multiparty Computation is to first imagine a Trusted Party which obtains inputs from all the other parties to produce a computed result. This Party is then replaced by *Un-trusted* Parties performing secure computation with each other to produce the final result without knowing what the inputs were (apart from their own, of course). The question to ponder is − Is it right to call those parties Un-trusted? In other words, can we *assign* some trust to them, likely less than the Trusted party we began with, but certainly more than the Un-trusted case?

In a real world scenarios, it is rare to trust (or distrust) every party equally. For example, in the use case of an auction, the auctioning company can be trusted

more than some bidders or the sellers. In the case of a Banking service, the Bank can be trusted more than some of the clients in a computation involving multiple clients and the bank. The underlying factor behind this belief is that Auction companies or banks have more to lose if the knowledge of them cheating leaks out, than they gain from cheating. While other participants are generally under no such restraint. Also, these organizations are usually under formal ethical scrutiny. In the same sense, a person that has been a client of the firm since a long time can be trusted more than a client who has just recently signed contract. This hinges on the thought that a successful long time relationship leads increases trust.

This, however comes with a caveat that a party should not be assumed to be trusted to the extent where extracting secrets and/or manipulating the output is trivial for it. For example, in a setting involving computation between bank and it's many clients, the bank can be trusted more than individual clients, but the trust should not let the bank exceed a threshold that allows it to view the private inputs on it's own, or worse, manipulate the output since then it becomes pointless to have a multiparty computation.

The Graded Trust model. So, the fundamental question that the previous discussion in this paper boils down to is − can there be any MPC model(s) that can take into account the trust differentials between the parties? That is, can a higher trusted party have higher weight in the protocol, enough to prevent a lower several lower trusted parties (acting collusive) to surpass the security threshold (which is also adjusted for additional weight assigned to the parties).

This question is extremely important in the context of Large Scale MPC models where the system is generally designed to accommodate volunteers from the public, with little oversight of who participates. This can prove to be extremely problematic in the case where an adversary with significant resources spawns multiple nodes/parties and overcomes the security thresholds, as discussed previously.

A great example of such a large scale public system is the Tor Network [DMS04] that allows for any entity to join the network and contribute towards privacy preserving and censorship free internet. However, Tor Network's openness towards accepting new volunteers can easily prove to be harmful for it's users if the volunteers turn out to be bad actors, particularly nation-state adversaries with huge resources who can mount Sybil Attacks [WELF16]. Towards this, the network employs consensus weight: a numeric value that assigned to each node that throttles the bandwidth of traffic passing through it, thus preventing Sybil attacks.

Taking inspiration from Tor, we build the Graded Trust Model and our paper presents a key piece in the puzzle towards resilient large scale MPC deployments. We start by introducing the notion of Graded Trust that allows one to lay down policies that if satisfied completely or partially allows the parties to earn Trust Grades. A higher trusted party can then have more weight in the computation than lower trusted party. This makes it possible to neutralize the potential neg-

ative influence of supposed malicious parties, thus allowing the adversary to not obtain the required thresholds to compromise the security guarantees of MPC protocols.

Apart from containing the influence of the Adversary, our model also makes MPC more inclusive by allowing greater number of parties to participate, even those who otherwise wouldn't have been allowed to be a part of the computation. This is in contrast with the traditional MPC models that are binary in their approach towards participation: either you are a participant or you are not. Graded Trust means Graded Participation, which is similar to Tor's census weight mechanism − a volunteer's effective bandwidth is throttled by certain parameters that indicate how trusted they are. More trusted participants get more job in the system.

Our model relies on making a Party simulate one or more virtual parties. This however incurs additional costs, particularly the communication cost. We use the example setting of BGW protocol being played between the parties to present the communication cost associated with our model.

1.2 Background on BGW protocol

One of the most fundamental MPC techniques is the Ben-Or, Goldwasser, Widgerson (BGW) protocol. This protocol builds upon Shamir's Secret Sharing Scheme [Sha79] to provide Secure Multiparty Computation. In the semi-honest (honest but curious) setting with point-to-point channels, BGW provides perfect security as long as the passive adversary controls less than half of the parties (i.e $\tau < N/2$,). If the adversary is malicious (byzantine adversary), perfect security is achieved only if $\tau < N/3$.

The Communication Complexity of the protocol Computing a Circuit with $|C|$ multiplication gates is $O(|C|.N^4)$ in the best case when the parties are expected to stick to the protocol, while in the worst case the communication complexity is $O(|C|.N^6)$ when some number (strictly less than the threshold) of the parties deviate from the protocol due to being malicious or simply malfunctioning. It's important to note that the above complexities are for a setting that only has point to point channels, rather than broadcast channels.

1.3 Outline

We introduce some preliminary ideas and definitions in §2. §3 presents the notions of Trust Grade and Trust Structure and the related ideas. We also present the Trust Structure award protocol. Then, in §4 building upon the ideas of previous section, we present the Graded Trust Model with an example. §5 deals with converting a conventional MPC protocol to protocols in the Graded Trust Model, and it's execution. §6 provides an analysis of the additional communication costs that our model carries. §7 Provides a discussion related to our paper and makes a case of using our model in the real world. §8 deals with discussing the potential Future work based on our paper.

1.4 Related Work

Player Simulation and General Adversary Structure. The notion of Adversary structure [HM00] introduced by Hirt et al involves defining the security of a MPC protocol not in terms of thresholds, but as set of parties that the protocol is secure against if they turn out to be corrupt.

This involves, grouping together a set of participants and having them simulate a single party by executing an MPC within themselves. This allows for various combinations of parties to be grouped together, and the adversarial influence to be compensated for by the non-corrupt parties in the sub-protocol, or if more than a threshold of parties are corrupt in the group, the simulated party is essentially corrupt, but the adversarial influence of many corrupt parties is limited to a few number of simulated parties.

2 Preliminary

Before laying down our model, we first present important concepts and definitions, some of which utilize UC Framework [Can20] introduced by Canetti.

Definition 1. *An interactive turing machine (ITM) μ is an extension of the classical turing machine that allows the machine to interact with it's environment during the protocol execution. Every ITM is denoted by the symbol μ, where $\mu = (ID, C, \tilde{\mu})$. Here, ID is the identity string of the ITM, C is the communication set, i.e, the machines allowed for communication, and $\tilde{\mu}$ is the protocol to be executed by the ITM.*

Our realization of the ITM has the following tapes:

- `Identity tape`*: This tape contains a string that is used to uniquely identify an ITM in a system of ITMs. This tape is* `read only`*, meaning that μ itself cannot change the contents on this tape. The contents of this tape are set externally.*
- `Input tape`*: The* `externally writable` *input tape receives a string of input from other ITMs or the environment.*
- `Output tape`*: μ writes the output message to this tape to be read by other ITMs or the environment.*
- `Subroutine Output Tape`*: This tape consists of the output message generated by the Subroutine or in precise terms - sub-machine.*
- `Subroutine Input Tape`*: The input counterpart of the* `Subroutine Output Tape`*.*
- `Backdoor tape`*: The standard backdoor tape is an* `externally writable tape` *that is used by μ to receive corrupting input from the adversary. This tape is only accessible to the adversary.*
- `Activation tape`*: This tape indicates to the reader whether μ is active or not. Consists of string* `active` *and* `inactive` *when the ITM is active and not active, respectively.*

Operator Party. An Operator Party (or simply, a party) in our model is the machine responsible for carrying out the MPC protocol with other parties to produce the computed result. An Operator Party has the responsibility to operate Virtual Parties (as defined shortly).

Definition 2. *We define an Operator Party p to be essentially an Interactive Turing Machine (ITM) in the UC Framework where an ITM is a tuple $p = (ID, C, \tilde{\mu})$ where ID represents the Identity of the machine, C is the set of communication lines with other machines and $\tilde{\mu}$ is the program that the machine is tasked with executing.*

Virtual Party. A virtual party is essentially a party operated virtually by the Operator Party. Each Operator party can have multiple virtual parties, but a virtual party can have only one operator. A virtual party can be corrupt iff it's operator party is corrupt.

Definition 3. *A Virtual Party v is a machine $v = (ID', C', \tilde{\mu}')$ where μ' is an internal machine of μ. This machine is spawned by the caller μ.*

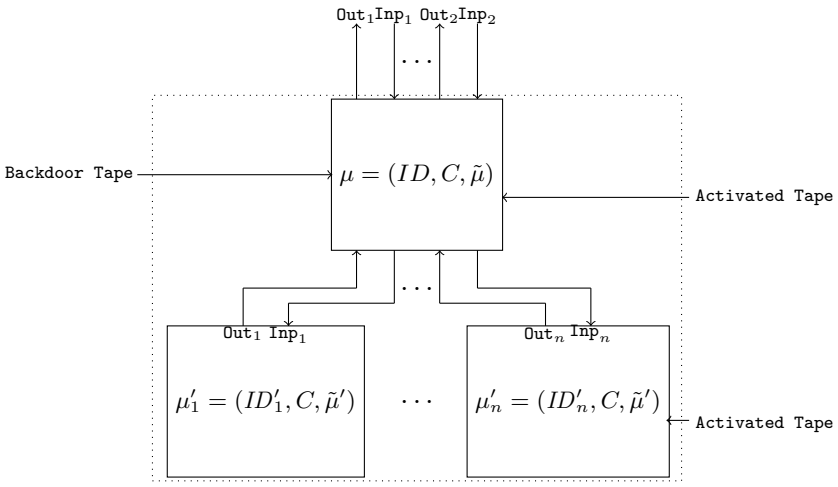

Fig. 1: The interaction between μ representing an Operator Party and it's sub-machines μ' representing it's virtual parties

While we define an ITM with the set of tapes as listed, but not every ITM makes use of all the tapes defined. μ for a player makes use of all tapes, while μ' does not see any use for the backdoor tape (as we discuss shortly).

Definition 4. *A protocol (π) in the UC Framework is defined as the algorithm designed to be executed by all ITMs.*

An Important thing to note here is that the algorithm actually executed by different ITMs may by different according to their roles in the system, but we include all of those sets of algorithms under the very broad notion of a protocol, in some sense like a union of algorithms. Hence, we can say that the there only one protocol π in execution on all ITMs, while the execution of some (or all) ITMs may be distinct.

3 Trust Structure

Although we've mentioned Trust Grade before, we need to formalize it and present *Trust Structure* that builds upon Trust Grades to provide an overall view of how much is each party trusted in comparison to others. Then, we move on to discussing how the Trust Structure is assigned to a set of parties.

Definition 5. ***Trust Grade.*** *The Trust Grade t_i of a party p_i is a Natural Number representing the quantum of trust in that party during the protocol execution.*

We consider the Trust Grade to be a natural number (including zero) but it does not have to be necessarily. Any entity that has comparison and summation operation defined over it can be used in place of a natural number. However, using natural number are more intuitive for this task, and easier to deal with than the alternatives.

If $t_i > t_j$ we say that p_i is more trusted than p_j (or p_j is less trusted than p_i). Likewise if $t_i = t_j$, we say that p_i and p_j are equally trusted.

We reserve $t_i = 0$ for parties that are proven corrupt, as opposed to other parties that are speculatively corrupt for which we reserve $t_i > 0$. Proven corrupt parties should ideally barred from participating in the protocol, which is why a zero Trust Grade would ensure that the party practically has no weight during the protocol execution. It is also easy to see that $t_i = 1$ is the minimum Trust Grade that is assigned to any party that is a part of the protocol.

The upperbound on the Trust Grade for a party is given by the *adjusted* security threshold. We will go into the details of it shortly.

Definition 6. ***Trust Structure.*** *Trust Structure $\mathcal{T}$ is a mapping from a Party to a trust grade. Formally,*

$$\mathcal{T} : p_i \to t_i$$

The Trust Structure can be thought of as a table of Parties and the corresponding trust grades assigned to them.

3.1 Trust Structure Award Mechanism

While we have discussed the idea of Trust Structure, we did not detail how is the Trust Structure going to be assigned to a set of parties. The basic idea is to have a public document designed by a special entity called the Administrator that lists

points that each party could earn on the bases of certain security requirements fulfilled. The parties make their *Trust Parameters* public, which can then be assessed to assigned Trust Structure to that set of parties. The important point to note is that a party or a dealer (which we will describe shortly) can drop out of executing the MPC protocol if they are not satisfied with the Trust Structure assigned.

Trust Policy. The Trust Policy is set of requirements that will eventually be used to assign Trust Grade to a Party if it fulfills a subset of requirements. We assume that the Trust Policy is available publicly as a Document before the start of the MPC protocol. We also envision that in practise, the Trust Policy would be designed after consultation with the all the stakeholders (participants in the protocol).

Trust Parameter. In response to each policy requirement, parties produce a Trust Parameter which is a publicly verifiable piece of information about the Party. The trust parameters are then used to assign Trust Grades to the parties. For example, the policy requirement can allow parties situated in certain regions of the world known for liberal cyber laws (say, Switzerland) to earn higher trust for it. In this example, each party provides it's IP address − a piece of information that is publicly verify-able. This piece of information is a Trust Parameter.

Administrator. We introduce the Administrator $\mathcal{M}$ as an entity that designs and publishes the Trust Policy, assess the Trust Parameters supplied by each of the parties and assigns Trust Structure to them. The job of providing Trust Grades to the parties can easily be automated if the policy requirements can be quantified.

Dealer. A Dealer in a MPC scheme is a special party that provides correlated randomness to other (non-dealer) parties. The dealer also goes by the name client. It supplies shares of it's input to each of the parties, usually via a secret sharing scheme and then those parties are given the responsibility of carrying out the computation to produce result. The set of dealers are denoted by $\mathcal{D}$.

Adversary We consider the Adversay to corrupt the parties statically − The set of corrupt parties are fixed right at the beginning of the protocol. We use $\mathcal{P}^{\mathcal{A}}$ to denote the set of corrupted parties.

Sentient Entities. We can bunch the Administrator, Dealers and Parties into one category that we call sentient entities. Sentience here does not have the same deep connotation that it does in Philosophy, but rather just a way to describe that each of these entities have their own subjective view about the computation and in specific, their own view of the extent of adversarial influence

on the computation. The Sentient Entity will have to make value judgements, particularly assigning Trust Structure to the Parties, and for the Parties and Dealers to see if they are satisfied with the Trust Structure assigned by the Administrator. We denote a sentient entity by $\xi \in \mathcal{M} + \mathcal{P} + \mathcal{D}$.

Definition 7. ***Ideal Setting.*** *We define Ideal Setting to be a setting where no party is more likely to be corrupt than the other. Mathematically,*

$$Pr[p_i \in \mathcal{P}^{\mathcal{A}}] = Pr[p_{j \neq i} \in \mathcal{P}^{\mathcal{A}}]$$

Conventional MPC models assume the Ideal setting. However, different parties owing to the differences in their operational parameters have varied chances of being corrupted by the Adversary. Say, statistically it is known that 3 out of 10 nodes that join a the computation in temporal proximity to one another are corrupt (that is, part of a sybil attack by the Adversary). This means, a node that joins the computation with such a characteristic is going to be corrupt with 30% chance. This perceived truth (by a sentient entity) is based on their statistical observations. It may be less accurate than the *Ground Truth*.

Definition 8. ***Ground Truth.*** *Ground Truth $\mathcal{G}_i$ is the realistic probability of a party p_i being corrupt by the adversary $\mathcal{A}$. Formally, we state:*

$$\mathcal{G}_i = Pr[p_i \in \mathcal{P}^{\mathcal{A}}]$$

Definition 9. ***Perceived Truth.*** *Perceived Truth $\mathcal{C}^{\xi}$ is the probability of a party p_i being corrupt by the adversary $\mathcal{A}$, as perceived by ξ. In formal terms,*

$$\mathcal{C}_i^{\xi} = Pr^{\xi}[p_i \in \mathcal{P}^{\mathcal{A}}]$$

Ground truth is the *actual* probability of a party being corrupt, as opposed to the perceived truth.

Trust Structure is assigned using the Perceived Truth. It is not important here to describe the exact mapping between Perceived Truth and the corresponding Trust Structure. We leave this job for the implementation of this model to decide.

Definition 10. ***Adversary's Gain.*** *The Adversary's Gain is the number of parties (weighted by their trust grades) the Adversary is able to corrupt, as expected by ξ.*

$$G^{\xi} = \sum_{t_i \in \mathcal{T}^{\xi}} t_i \cdot \mathcal{C}_i^{\xi}$$

Threshold. For an MPC protocol to be successful, the number of corrupt parties must stay under a certain threshold. To take an example, consider the BGW protocol where the passive-Adversary should control less than $N/2$ parties for the protocol to be secure against it. Since in our model, we have adjusted the weight of the parties in accordance to how trusted they are, we also need to adjust the security thresholds. So in our model, the passive adversary should control less than $\sum_i t_i/2$. The general rule to transform security thresholds is that the number of parties originally should be replaced with the sum of the Trust Grades.

`SATISFY()`. Each Sentient Entity in our protocol has a personal view of the adversary's capabilities, However, Trust Structure $\mathcal{T}^{\mathcal{M}}$ is assigned to a set of parties by the Administrator. It is natural that some parties and dealers would have a differing view than the Administrator. This is where they execute `SATISFY(`$G^{\mathcal{M}}, G^{\xi}$`)`. Although, we envision the heuristic to be defined by the implementation, roughly it should check if $G^{\mathcal{M}} < G^{\xi}$. That is, the Administrator's Trust Structure is fare's better than the ξ's Structure in containing the Adversary. A weaker heuristic may let the $G^{\mathcal{M}} > G^{\xi}$, but with the condition that $G^{\mathcal{M}} < \tau'$. That is, Administrator's Trust Structure is expected to perform worse than ξ's Trust Structure in containing the Adversary, but at least the corrupted parties do not cross a weaker threshold τ'. The party may also want to check if enough Trust Grade has been assigned to itself.

Protocol 1. Trust Grade Award Protocol π_t

The protocol π_t is executed in presence of an initial Environment ITM $\mathcal{E}$, parties $\mathcal{P}$, Administrator $\mathcal{M}$, and dealer(s) $\mathcal{D}$.

1. $\mathcal{E}$ invokes the machines in set $\mathcal{P}$, $\mathcal{D}$, and $\mathcal{M}$.
2. $\mathcal{M}$ formulates Trust Policy Document and sends it to $\mathcal{P}$ and $\mathcal{D}$.
3. Each party in $\mathcal{P}$ send Trust Parameters to other parties in $\mathcal{P}$, $\mathcal{M}$ and $\mathcal{D}$.
4. $\mathcal{M}$ formulates Trust Structure $\mathcal{T}^{\mathcal{M}}$ and sends it to $\mathcal{P}$ and $\mathcal{D}$.
5. Each $\xi \in \mathcal{P} + \mathcal{D}$ execute `SATISFY(`$G^{\mathcal{T}}$`,` G^{ξ}`)`. If `SATISFY` outputs $\perp$, the ξ outputs $\perp$ and exit the protocol. Otherwise, each one of $\mathcal{P}$, $\mathcal{D}$ and $\mathcal{M}$ would have $\mathcal{T}^{\mathcal{M}}$ at the end of the protocol.

4 Graded Trust Model

Consider an MPC scheme with Parties $\mathcal{P} = \{p_1, p_2, ...p_n\}$ that is $\mathcal{A}$-secure for $|\mathcal{P}^{\mathcal{A}}| \leq c$. That is, the scheme is $\mathcal{A}$-secure only if the cardinality of the set of corrupt parties $\mathcal{P}^{\mathcal{A}}$ is less than the threshold c.

Conventional MPC models do not differentiate between parties, in the sense that all parties are supposed to be functional and honest with equal probability, but in case some turn out to be corrupt, there are thresholds to take care of them. So as long as the number of corrupt parties remain under the threshold, the computation is guaranteed to be secure except for incurring higher computational or communication costs. One way to look at this from the prism of the Graded Trust Model is that for each Party p_i, $t_i = 1$. This egalitarian state can be called *ideal*. What makes it ideal is that no party is less trusted than other. This is not considering the entities barred from participation on the ground of not being trusted at all. The Adversary is supposed to be able to corrupt all parties with equal chances (alternatively, the parties shall turn out to be honest with equal chances), though the total parties corrupt shall still stay below a certain threshold as guaranteed to be secure by the scheme.

Like most things ideal in thought, this scenario is far from real life. With prior information hinting at a party *probably* being corrupt, we can do much better.

We begin by assigning Trust Structure $\mathcal{T}$ to the set of parties. That is, each Party $p_i \in \mathcal{P}$ gets a corresponding Trust Grade t_i obtained by executing π_t.

A setting in which one or more parties are trusted less than the others is considered by us as a non-ideal state, and Graded Trust Model attempts to correct for this giving lower trusted Parties lesser weight in the protocol. Towards this purpose we introduce a set of Virtual Parties $\mathcal{V}^i$ for a Party p_i, where $|\mathcal{V}^i| = t_i$. The higher the Trust Grade, the more Virtual Parties a Party is allowed to operate. Since higher trusted parties are allowed to operate higher number of virtual parties, it means that the protocol would eventually have more higher trusted Parties, which would counter the negative influence of the lower trusted Parties.

An Example. Consider an instance of BGW protocol being played between 4 parties: p_1, p_2, p_3 and p_4, which is actively-secure for under a threshold $N/2 = 2$. Say, we have prior probabilities of corruption shown in the table below. The Administrator then assigns Trust Grades to the parties as also shown in the table below.

Party	Probability of Corruption	Trust Grade
p_1	0.2	2
p_2	0.5	1
p_3	0.1	2
p_4	0.4	1

The Parties and the Dealers execute the `SATISFY` functionality to decide if they want to continue with the proposed Trust Structure. If they continue, they are allowed to operate a number of virtual parties in accordance with the Trust Structure. Here, p_1 and p_3 (which are assigned 2 virtual parties each) are trusted more than p_2 and p_4 (which are assigned one virtual party each). In the original setting the threshold was $N/2$, meaning that at most one party can be corrupt and yet the protocol be successful. After assigning Trust Structure, the threshold is updated to be 3, which allows for the adversary to corrupt parties in any one element of the set $\mathcal{Z} = \{\{p_1\}, \{p_2\}, \{p_3\}, \{p_4\}, \{p_1, p_2, \}, \{p_3, p_4\}, \{p_2, p_3\}, \{p_1, p_4\}\}$, without affecting the protocol since the combined weight of parties in any one element of this set does not exceed 2. As shown in Fig.2, each party p_i operates a set of t_i virtual parties $\{v_1^i, v_2^i, ...\}$

5 Protocol Transformation and Execution

This section describes the transformation of a standard protocol μ_0 with equal trust for each party to a protocol μ in our model for a Trust Structure $\mathcal{T}$. Roughly, the transformation takes into consideration the Trust Grade of each

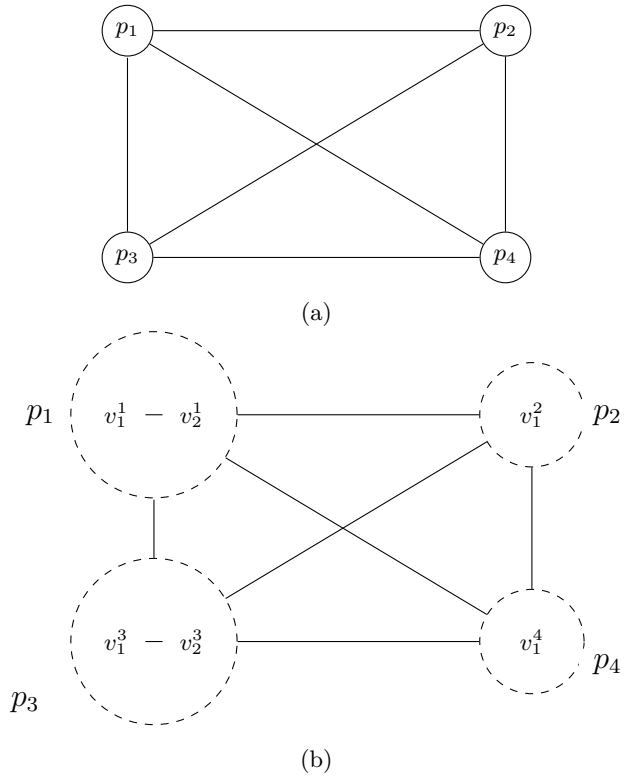

Fig. 2: (a) represents a classical MPC model of a protocol between 4 parties, each of which is trusted equally. (b) shows our model of graded trust between the parties. Not shown in the figure for the sake of simplicity is the fact that each Virtual Party has a one to one connection to every other Virtual Party, indirectly through their respective Operator Parties.

Party and allows it to simulate a number of Parties equal to it's Trust Grade. Then, we describe the protocol execution.

5.1 Setting

We consider a protocol π with the set of all Parties $\mathcal{P} = \{p_1, ...p_i...p_n\}$ and a Trust Structure $\mathcal{T}$. The set $\mathcal{V}^i$ is the set of virtual parties of p_i with trust grade t_i.

We assume that either the Trust Structure for the particular setting is agreed upon beforehand by all the Parties, Dealers and the Administrator in the protocol or they together execute π_t to obtain it.

We assume that either the Parties communicate with each other through a Broadcast Channel or a pairwise channels authenticated with Public Key Infrastructure (PKI). As far as communication between an Operator and their Virtual Parties is concerned, that does not necessarily require any special setup since Virtual Parties can be operated on the same machine as the Operator Party.

5.2 Party Translation

All communication in our Model happens between Virtual Parties which are in some cases operated by different Operator Parties, and in other cases operated by the same Operator Party. We need a way to route messages between these Virtual Parties. The first step towards this is to divide up Parties in domains we term as *Party Space*. We then need to map Parties in one space to the Parties in the other space, and towards this we make use of the *Party Map*.

Party Space. Each party (virtual or not) in our protocol operates in an abstract container called the Party Space, denoted by $\mathcal{S}$. Essentially, a party space $\mathcal{S}$ is the set of all Parties operating in that domain. All operator parties exist in Party space $\mathcal{S}_p$, while all virtual parties operated by p_i, including p_i itself exist in $\mathcal{S}_i^v$. Notice that, p_i exists in the intersection of $\mathcal{S}_i^v$ and $\mathcal{S}_p$. It's easy to see that,

$$\mathcal{S} = \mathcal{S}_p \cup \bigcup_i \mathcal{S}_i^v$$

$$\{p_i\} = \mathcal{S}_p \cap \mathcal{S}_i^v$$

Definition 11. ***Party Map.*** *Party Map ψ is a map from a Party in S_p and a string* `destination-id` *to a Party in S_i^v. Formally,*

$$\psi : S_p \times \texttt{destination-id} \rightarrow S_i^v$$

In a similar fashion we define,

Definition 12. ***Inverse Party Map.*** *Inverse party space map ψ^{-1} maps a Party in S_i^v and a string* `destination-id` *to a Party in S_p. That is,*

$$\psi^{-1} : S_i^v \times \texttt{destination-id} \rightarrow S_p$$

Both Party Map and Inverse Party Map are essentially implemented as routing tables contained with the Operator parties. Whenever an Operator Party receives a message from it's Virtual Party (a subroutine), it looks up who the message is addressed to (the `destination-id` string) and applies ψ to it to obtain the Operator Party that this message needs to be relayed to. In the same way, when this message reaches the respective Operator Party, it uses ψ^{-1} to send the message to the correct Virtual Party.

5.3 Protocol Translation

The process of Protocol Translation involves parsing through the standard protocol specification and inserting the role of the Virtual Parties. Essentially, the protocol translator reads the protocol specification line by line and replaces the instructions with references to a Party with instruction(s) having references to the respective Virtual Parties of that Party if they exist.

Definition 13. ***Protocol Translation.*** *Protocol Translation Γ is a mapping defined as,*

$$\Gamma : \pi_0 \times \mathcal{P} \times \mathcal{T} \rightarrow \pi \times \mathcal{P} \times \mathcal{V} \times \psi$$

We input the original protocol along with a set of parties and the Trust Structure into the protocol translation and get a modified protocol with the same set of parties, set of virtual parties operated by each of those parties and the a mappings ψ and ψ^{-1} between parties and their virtual parties.

We use the notation π_0 to denote the standard MPC protocol, while translated protocol is denoted by π which overall performs the same functionality but accounts for the added virtual parties.

We present an example in Fig.3 where a simple MPC protocol μ_0 with equal trust is translated into a protocol μ with graded trust and roles for virtual parties. Each instruction concerning a particular party is translated into a list of multiple instructions of the same kind but involving the virtual parties of that party.

5.4 Execution of a Operator Party

In §3 we described party and virtual party in our model using UC Framework. While Virtual parties are a subroutine of their respective parties, they differ subtly from traditional notion of subroutine in UC Framework in that they are not expected to only communicate with their operator party. They can indirectly communicate with any party, including all the virtual parties, even those being operated by other operator parties.

The operator party is supposed to sit between it's virtual parties and the outside world. We introduce a modification to the conventional write operation of the ITMs. In addition to the `message` field, the ITMs are required to also supply `source-id` and `destination-id` which are the IDs of the sender and

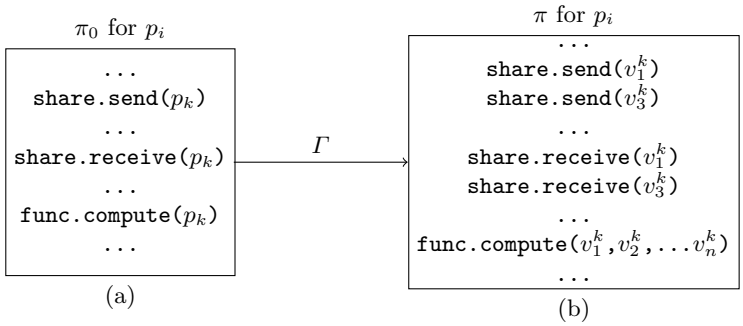

Fig. 3: (a) shows a traditional MPC protocol for p_i with equal trust. (b) shows the protocol for p_i translated into our model.

receiver ITMs respectively. Operator parties effectively play the role of routers in our model.

We confine the Adversary to only corrupt the operator party. That is, if an adversary wants to corrupt a virtual party, it needs to corrupt it's corresponding operator party. In the same way, if any adversary is able to corrupt the an operator party, it can then corrupt all of it's virtual parties. Since in practice we envision virtual parties as processes on the operator party's machine, any Adversary that is able to corrupt the operator party should be able to corrupt the virtual parties.

FUNCTIONALITY 1. Operator Party $\mathcal{F}_i^{\mathcal{P}}$

The protocol π is executed in presence of an Adversary $\mathcal{A}$ and an initial Environment $\mathcal{E}$. Let $\mathcal{F}_i^{\mathcal{P}}$ be the functionality of p_i. $\mathcal{P}^{\mathcal{A}}$ is the set of parties corrupted by $\mathcal{A}$.
Common Input: Trust Structure $\mathcal{T}$

1. $\mathcal{E}$ each party in $\mathcal{P}$, $\mathcal{V}$ and $\mathcal{F}_i^{\mathcal{P}}$. $\mathcal{E}$ sets the control function of these ITMs to π, and writes a unique ID to the identity tape for each of them. $\mathcal{E}$ inserts `active` on the `activation tape` for $\mathcal{P}$, $\mathcal{V}$ and $\mathcal{F}_i^{\mathcal{P}}$.
2. $p_i \in \mathcal{P}$ reads string `(destination-id, message)` from the `subroutine output tape` of v_i.
3. If $p_i \in \mathcal{P}^{\mathcal{A}}$, p_i reads string on `backdoor tape`. Either p_i halts outputting $\perp$, or this string replaces that from v_i.
4. $p_i \in (\mathcal{P} - \mathcal{P}^{\mathcal{A}})$ obtains p_j through ψ(`destination-id`) and writes `(destination-id, message)` to `input tape` of p_j. $p_i \in \mathcal{P}^{\mathcal{A}} writes$`(destination-id, message)` to p_j and/or v_i.

6 Communication Complexity

As in before, Consider that BGW protocol is being played out between Operator Parties $\mathcal{P} = \{p_1, p_2, ...p_n\}$. Each Operator Party p_i operates a set of Virtual Parties $\mathcal{V}_i$. Let $|\mathcal{V}| = \sum_i |\mathcal{V}_i|$ be the total number of virtual parties in operation. The setup only considers point to point authenticated channels between the operator parties. The circuit C evaluated by the parties has $|C|$ number of multiplication gates.

The case in which each party adheres to the protocol is called an *Optimistic case* and the cost associated with it is *Optimistic Cost*. If any some parties (bounded by the respective threshold) deviate from the protocol, we call that a *Pessimistic case* and the cost associated with it *Pessimistic cost*. The latter requires extra communication to correct for the deviation from the protocol, but otherwise produces the same result as the former.

It is known that the cost[AL17] of BGW protocol consisting of N parties is $O(N^4)$ and $O(N^6)$ in the optimistic and pessimistic case respectively. One may think of our model as $|\mathcal{V}|$ independent machines communicating with each other through communication lines, towards realizing the goal of the BGW protocol. It is simple to see that the communication cost for our model would be adjusted for the amount of new (virtual) parties we add, as summarized by the table below.

Cost	π_0	π
Optimistic Cost	$O(\|C\|.\|\mathcal{P}\|^4)$	$O(\|C\|.\|\mathcal{V}\|^4)$
Pessimistic Cost	$O(\|C\|.\|\mathcal{P}\|^6)$	$O(\|C\|.\|\mathcal{V}\|^6)$

While the above mentioned complexity holds true in theoretical sense, but we envision that practically the virtual parties that would belong to a particular Operator Party wouldn't necessarily have to communicate over communication line like they do for Virtual Parties not operated by their own Operator Party. For example, p_i might be simulating the Virtual Parties on it's machine. In that case, the cost of communication between two virtual parties being simulated would be zero.

Cost	π_0	π
Optimistic Cost*	$O(\|C\|.\|\mathcal{P}\|^4)$	$O(\|C\|.(\|\mathcal{V}\|^4 - (\sum_i \|\mathcal{V}_i\|^4))$
Pessimistic Cost*	$O(\|C\|.\|\mathcal{P}\|^6)$	$O(\|C\|.(\|\mathcal{V}\|^6 - (\sum_i \|\mathcal{V}_i\|^6))$

7 Discussion

As we have mentioned earlier, we envision that our system will likely find it's use in large scale (public) MPC deployments described in §1. In some ways, such a system shares it's philosophy with democracy. A democratic system has the goal of making collective decisions with the use of (anonymous) voting. This democratic process of decision making and conventional MPC schemes are similar in that they adhere to the principle of *one person one vote* owing to egalitarianism.

It is easy to see why Democracy like most other ideas has had it's fair share of criticism from the time of ancient Greeks to modern day. A popular argument is that Democracy is vulnerable to to careful social engineering of atleast a majority of population, particularly those who are uninitiated in matters that require voting. In this aspect, democracy again shares this characteristic with traditional MPC schemes in that the Adversary has to target a certain threshold count of the most vulnerable parties, since MPC schemes offer security guarantees only upto a certain threshold.

A common solution prescribed for the perils of traditional democracies is the distribution of votes based on certain prerequisites. While, this may be a naive idea considering the common held ethics of voting, in §8.1 and §8.2 we show that allowing parties to have weights (or Trust Grades, as we call them) based on fulfilment of certain criteria is a step in the right direction.

7.1 Graded Trust MPC as an instrument of inclusive computation/decision making

Although, MPC protocols are designed to be resilient to at most a certain number of parties (less than the threshold), but there has been no notion of a particular party *probably* being corrupt. It has been left onto the MPC administrator to decide on who gets to be a participant.

We believe that this practice excludes entities who are willing but cannot participate due to constraints (such as lack of latest *secure* hardware, proprietary operating system, etc). Instead of barring them to participate, arrangements can be made for them in the MPC system where they are allowed to participate, but have less weight compared to others in the computation owing to lower trust. This is an effective middle ground between inclusiveness and security/privacy.

7.2 Graded Trust MPC as a incentive in the right direction

If a party's Trust Grade is tied to the number of requirements it fulfils, such a practice in a public MPC deployment can have a positive side effect that gaining trust within the system can be an incentive and encouragement for the public parties to comply with the latest security practices (laid down by the Administrator).

8 Future Work

In this paper, while we have presented the Graded Trust Model, we shy away from constructing an MPC system based on it. A challenge remains to chose worthy Trust Policies and deriving appropriate Grades from the Trust Parameters supplied in answer to those Trust Policies.

With the emerging models of Large scale (public) MPC systems, graded trust can also prove to be very useful there, as discussed before. Such large scale systems require parties to have the ability to connect and disconnect with the

ongoing computation, without effecting the end result. This produces a challenge for the threshold based security definitions and an opportunity to employ our model, but work needs to be done to figure out how exactly to re-calculate trust after every epoch.

Our paper does not discuss the Malicious Administrator scenario. The Trust Policy in our model is set by the Administrator. This is troublesome it provides the Administrator with power to influence the protocol indirectly by adjusting the Trust policies in favor of the Adversary. A naive solution may be to allow the parties to together come up with Trust Policies. We call it naive because this solution is vulnerable to sybil attacks − An adversary just has to spawn majority number (or any threshold) of nodes to set it's desired policies.

Our model only considers a *static* adversary, while a stronger model of mobile adversary may exist. It remains a challenge to develop a model where the adversary plays their move after the Trust Structure has been declared by the Administrator.

References

[AAHMA16] Md Momin Al Aziz, Mohammad Z Hasan, Noman Mohammed, and Dima Alhadidi. Secure and efficient multiparty computation on genomic data. In *Proceedings of the 20th International Database Engineering & Applications Symposium*, pages 278–283, 2016.

[ABL+18] David W Archer, Dan Bogdanov, Yehuda Lindell, Liina Kamm, Kurt Nielsen, Jakob Illeborg Pagter, Nigel P Smart, and Rebecca N Wright. From keys to databases—real-world applications of secure multi-party computation. *The Computer Journal*, 61(12):1749–1771, 2018.

[AL17] Gilad Asharov and Yehuda Lindell. A full proof of the bgw protocol for perfectly secure multiparty computation. *Journal of Cryptology*, 30(1):58–151, 2017.

[BCD+09] Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas P. Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, Michael I. Schwartzbach, and Tomas Toft. Secure multiparty computation goes live. In Roger Dingledine and Philippe Golle, editors, *Financial Cryptography and Data Security, 13th International Conference, FC 2009, Accra Beach, Barbados, February 23-26, 2009. Revised Selected Papers*, volume 5628 of *Lecture Notes in Computer Science*, pages 325–343. Springer, 2009.

[BGG+20] Fabrice Benhamouda, Craig Gentry, Sergey Gorbunov, Shai Halevi, Hugo Krawczyk, Chengyu Lin, Tal Rabin, and Leonid Reyzin. Can a public blockchain keep a secret? In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part I*, volume 12550 of *Lecture Notes in Computer Science*, pages 260–290. Springer, 2020.

[BHZ+18] Assi Barak, Martin Hirt, Eth Zurich, Lior Koskas, and Yehuda Lindell. An End-to-End System for Large Scale P2P MPC-as-a-Service and Low-Bandwidth MPC for Weak Participants *. *ACM SIGSAC Conference on Computer & Communications Security*, 18:18, 10 2018.

[BKK+] Dan Bogdanov, Liina Kamm, Baldur Kubo, Reimo Rebane, Ville Sokk, and Riivo Talviste. Students and Taxes: a Privacy-Preserving Study Using Secure Computation. *Proceedings on Privacy Enhancing Technologies*, 2016(3):117–135.

[BLV17] Azer Bestavros, Andrei Lapets, and Mayank Varia. User-centric distributed solutions for privacy-preserving analytics. *Commun. ACM*, 60(2):37–39, 2017.

[BOGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, page 1–10, New York, NY, USA, 1988. Association for Computing Machinery.

[Can20] Ran Canetti. Universally composable security. *J. ACM*, 67(5):28:1–28:94, 2020.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (abstract) (informal contribution). In Carl Pomerance, editor, *Advances in Cryptology – CRYPTO'87*, volume 293 of *Lecture Notes in Computer Science*, page 462, Santa Barbara, CA, USA, August 16–20, 1988. Springer, Heidelberg, Germany.

[CGG+] Arka Rai Choudhuri, Aarushi Goel, Matthew Green, Abhishek Jain, and Gabriel Kaptchuk. Fluid MPC: Secure Multiparty Computation with Dynamic Participants. Technical report.

[CGH+18] Koji Chida, Daniel Genkin, Koki Hamada, Dai Ikarashi, Ryo Kikuchi, Yehuda Lindell, and Ariel Nof. Fast large-scale honest-majority MPC for malicious adversaries. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 10993 LNCS, pages 34–64. Springer Verlag, 2018.

[CWB18] Hyunghoon Cho, David J Wu, and Bonnie Berger. Secure genome-wide association analysis using multiparty computation. *Nature biotechnology*, 36(6):547–551, 2018.

[DMS04] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: The second-generation onion router. In Matt Blaze, editor, *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*, pages 303–320. USENIX, 2004.

[Dou02] John R. Douceur. The sybil attack. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 2429, pages 251–260. Springer Verlag, 2002.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all NP-statements in zero-knowledge, and a methodology of cryptographic protocol design. In Andrew M. Odlyzko, editor, *Advances in Cryptology – CRYPTO'86*, volume 263 of *Lecture Notes in Computer Science*, pages 171–185, Santa Barbara, CA, USA, August 1987. Springer, Heidelberg, Germany.

[GSB+17] Adrià Gascón, Phillipp Schoppmann, Borja Balle, Mariana Raykova, Jack Doerner, Samee Zahur, and David Evans. Privacy-preserving distributed linear regression on high-dimensional data. *Proc. Priv. Enhancing Technol.*, 2017(4):345–364, 2017.

[HLOW16] Brett Hemenway, Steve Lu, Rafail Ostrovsky, and William Welser. High-precision secure computation of satellite collision probabilities. In *Lecture*

Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2016.

[HM00] Martin Hirt and Ueli M. Maurer. Player simulation and general adversary structures in perfect multiparty computation. *J. Cryptol.*, 13(1):31–60, 2000.

[LJLA17] Jian Liu, Mika Juuti, Yao Lu, and N. Asokan. Oblivious neural network predictions via minionn transformations. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 619–631. ACM, 2017.

[LYCL13] Ming Li, Shucheng Yu, Ning Cao, and Wenjing Lou. Privacy-preserving distributed profile matching in proximity-based mobile social networks. *IEEE Transactions on Wireless Communications*, 12(5):2024–2033, 2013.

[MZ17] Payman Mohassel and Yupeng Zhang. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 19–38. IEEE Computer Society, 2017.

[NPS99] Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM conference on Electronic commerce*, pages 129–139, 1999.

[NWI+13] Valeria Nikolaenko, Udi Weinsberg, Stratis Ioannidis, Marc Joye, Dan Boneh, and Nina Taft. Privacy-preserving ridge regression on hundreds of millions of records. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 334–348. IEEE Computer Society, 2013.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[WELF16] Philipp Winter, Roya Ensafi, Karsten Loesing, and Nick Feamster. Identifying and characterizing sybils in the tor network. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 1169–1185, 2016.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 162–167. IEEE Computer Society, 1986.