A Little Honesty Goes a Long Way: The Two-Tier Model for Secure Multiparty Computation

Juan A. Garay*, Ran Gelles**, David S. Johnson***,
Aggelos Kiayias†, and Moti Yung‡

Abstract. Secure multiparty computation (MPC) as a service is becoming a tangible reality. In such a service, a population of clients wish to utilize a set of servers to delegate privately and reliably a given computation on their inputs. MPC protocols have a number of desired properties including tolerating active misbehavior by some of the servers and guaranteed output delivery. A fundamental result is that in order to achieve the above, an honest majority among servers is necessary. There are settings, however, where this condition might be overly restrictive, making it important to investigate models where this impossibility result can be circumvented, allowing secure computation to be performed even when the number of malicious participants outweighs the number of honest participants.

To this end, we introduce the *two-tier model* for MPC, where a set of m parties that are guaranteed to be honest (the *first tier*) remains "hidden" within a set of $n-m$ servers which are of dubious trustworthiness (the *second tier*), and where the objective is to perform MPC withstanding a number of active misbehaviors that is larger than $m/2$. Indeed, assuming αn of the second-tier servers are dishonest (where $\alpha \in (0,1)$), we present an MPC protocol that can withstand up to $(1-\epsilon)(1-\alpha)n/2$ *additional* faults, for any $\epsilon > 0$ and $m = \omega(\log n)$. Somewhat surprisingly, this allows the total number of faulty parties to exceed $n/2$ across both tiers.

We demonstrate that the two-tier model naturally arises in various settings, as in the case, for example, of a resource-constrained service provider wishing to utilize a pre-existing set of servers.

1 Introduction

A technically interesting and practically relevant configuration for performing secure multiparty computation (MPC) [GMW87]) is the commodity-based *client-server* approach, in which the vast part of the computation is delegated from one or more clients to one or more servers [Bea97]. Indeed, these settings have plenty of practical value, as demonstrated for example by the implementation and deployment of an auction system in the Danish sugar-beet market [BCD+09], and, more generally, in the emerging secure cloud computing paradigm.

A fundamental result in MPC with actively malicious ("corrupted") participants is that in order to be able to securely compute any function, it is necessary (and sufficient) that a majority of the parties are honest [Cle86, GMW87, CFGN96]. There are settings, however, where such a requirement might be too limiting. Thus, it is important to investigate models where it is possible to securely carry out any computation even though the number of malicious participants may be *higher* than the number of honest parties.

In this paper we put forth a new model for performing client-server-based MPC which we call the *two-tier model* for MPC. In this model, m servers are guaranteed to be properly functioning at the onset of the computation (those are identified by the set $\mathcal{P}_1$), while the remaining $n-m$ parties (the set $\mathcal{P}_2$) are of of dubious trustworthiness. In addition, it is assumed that $m \ll n$. We call $\mathcal{P}_1$

* Yahoo Labs, garay@yahoo-inc.com.
** University of California, Los Angeles, gelles@cs.ucla.edu.
*** Columbia University, dstiflerj@gmail.com.
† National and Kapodistrian University of Athens, aggelos@kiayias.com. Research supported by ERC project CODAMODA.
‡ Google Inc. and Columbia University, moti@cs.columbia.edu.

the *first-tier* servers and $\mathcal{P}_2$ the *second-tier* servers. The objective is to run MPC withstanding a number of active corruptions greater than $m/2$.

At first sight, it might seem unlikely that the two-tier setting could provide any advantage in circumventing the honest-majority requirement. Indeed, if we were to apply an MPC protocol directly, a subset of the n servers would need to be elected to execute it. Let α denote the ratio (out of all n servers) that are initially corrupted (thus, $m \leq (1-\alpha)n$). We observe the following: (i) if the MPC protocol is executed by the first-tier servers only, then the number of corruptions is bounded from above by $m/2$; (ii) on the other hand, if we apply the MPC protocol to all n servers indiscriminately, then the number of corruptions the protocol withstands is bounded by $\max\{0, (\frac{1}{2}-\alpha)n\}$; and (iii) the above bounds cannot be improved even if an arbitrary strategy is used to elect a subset from $\mathcal{P}_2$ while including all $\mathcal{P}_1$ servers which are known to be initially honest[1]. We thus conclude that applying standard MPC in the two-tiered setting achieves at best tolerance of $\max\{m/2, (\frac{1}{2}-\alpha)n\}$ malicious participants, which equals $m/2$ for the interesting case of an initial dishonest majority ($\alpha \geq 1/2$).

However, had we known the second-tier servers that are honest at the onset of the computation, we could have (at least in principle) beaten the above bound by using those servers as well. The bound on the number of corruptions in this case is $(1-\alpha)n/2$, which surpasses $m/2$. In fact, if such a protocol was at all feasible, it would imply that the *total* number of dishonest parties would be $(1-\alpha)n/2 + \alpha n$, which is larger than $n/2$, for any $\alpha > 0$.

Somewhat surprisingly, we show a protocol that achieves the above level of corruptions under the assumption that the uncorrupted servers from the two tiers can be made indistinguishable in the view of the adversary. Effectively, this enables our protocol to take advantage of *all* the honest second-tier servers, even in settings where an (unknown) overwhelming majority of them are corrupted. Specifically, we show the following:

Theorem 1 (Informal). *Given a set of n servers $\mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2$ such that an unknown α-fraction of them are initially corrupted, yet the servers in $\mathcal{P}_1$ are guaranteed to be honest, then for any $\epsilon > 0$ there is a two-tier MPC protocol which is adaptively secure against any adversary corrupting up to $(1-\epsilon)\cdot\frac{1-\alpha}{2}\cdot n$ additional servers, assuming $|\mathcal{P}_1| = \omega(\log n)$ and that the two tiers are indistinguishable to the adversary.*

The main idea behind our construction is to have all the servers take part in the protocol, albeit in a way that only the tier-1 servers perform the actual computation, while the tier-2 servers' role is to keep the identities of the tier-1 servers hidden. This is done by utilizing a novel message delivery mechanism we describe below, which has the net effect that the adversarial view of our MPC protocol transcript is hidden in a traffic analysis-resistant way amidst a large set of irrelevant (but indistinguishable) messages.

Performing MPC with a hidden, anonymous set of servers raises many interesting cryptographic questions; in particular:

> *How can first-tier servers run an MPC protocol amongst themselves, while any specific server (whether first- or second-tier) remains oblivious to other servers' identities?*

We solve this apparent contradiction by introducing the notion of *Anonymous yet Authentic Communication* (AAC), which allows a party to send a message to any other party in an anonymous *and* oblivious way. Despite being anonymous, the delivery is *authenticated*, that is, only the certified

[1] If t servers from $\mathcal{P}_2$ are elected, for some value of $t \in [n-m]$, then the number of corruptions needs to be bounded by $m/2 + t/2 - \alpha nt/(n-m)$. This function is maximized by either one of the previous bounds depending on the value of α with respect to $(1-m/n)/2$.

party will be able to send a valid message, and only the certified recipient will be able to correctly learn the message.

In more detail, in an AAC message delivery the sender will reveal to the recipient only his "virtual" protocol identity, but not his real identity. At the same time, the sender will remain oblivious to the real identity of the recipient, which will only be specified by its protocol identity. We show how to implement AAC message delivery by utilizing an *anonymous broadcast* protocol [Cha88], which allows parties to broadcast messages without disclosing the real identity of the sender of each message, and composing it with a suitable authentication mechanism. Finally, by substituting point-to-point channels with AAC activations in a suitable (adaptively secure) MPC protocol, we achieve our desired two-tier MPC functionality. The fundamental observation in the security proof is that the usage of the AAC message delivery mechanism effectively transforms any adaptive corruption strategy of the adversary against the MPC protocol to a *randomized* corruption strategy. Given this observation, we apply a probabilistic analysis using the tail bounds of the hypergeometric distribution and establish Theorem 1.

How to obtain two-tiers: the corruption/inspection game. The above result is predicated on being able to establish a subset $\mathcal{P}_1$ of honest parties, and that $\mathcal{P}_1$ and $\mathcal{P}_2$ can be made indistinguishable. Theorem 1 says that a super-logarithmic number of $\mathcal{P}_1$ servers would be sufficient to harness the maximal "resiliency" of the system in terms of number of corrupted servers that can be tolerated. However, it seems challenging to obtain a set $\mathcal{P}_1$ such that it is guaranteed that all the servers in $\mathcal{P}_1$ are honest, and still keep them hidden within the remaining servers. For example, one cannot form $\mathcal{P}_1$ simply by introducing new servers into a preexisting pool of servers, as those would easily be identified by the adversary (whose existence in the pool of servers precedes the event of the introduction of the new servers). To address this, we now illustrate a setting where two tiers naturally arise.

Assume that there is a single pool of machines out of which an α fraction is corrupted. Furthermore, assume we are allowed to *inspect* a fraction β of the servers, and *restore* their operating program into a safe state if found corrupt[2]. We can now define the set $\mathcal{P}_1$ to consist of all the servers that were inspected and found *clean* (i.e., uncorrupted). Note that the restored servers cannot be in $\mathcal{P}_1$, as these would not be indistinguishable from the other honest servers, since the adversary may be aware that he is no longer controlling them. We let $\mathcal{P}_2$ denote all the remaining servers.

For a given rate of corruption α and rate of inspection β at the onset, the question now is what is the maximal possible fraction of active faults γ we can still withstand when running an MPC protocol. We formalize the above as the following "Corruption/Inspection Game" between a service provider S and an adversary $\mathcal{A}$:

1. $\mathcal{A}$ corrupts $\alpha \cdot n$ of the servers for a parameter $\alpha \in (0, 1)$. Distinguishing corrupted from uncorrupted servers is undetectable at this stage (for the service provider S).
2. S inspects $\beta \cdot n$ servers and if they are corrupted it returns them to a clean state. β is the *inspection rate* of the service provider.
3. S opens the service by choosing a subset of the n servers to be tier-1 and the remaining servers tier-2; each server performs a designated protocol specific to its tier. Once the service is activated, then $\mathcal{A}$ may adaptively corrupt an additional $\gamma \cdot n$ servers. γ is called the *adaptive corruption rate*.

Fig. 1. *The Corruption/Inspection Game.*

[2] We assume here that corrupting a server means altering its operating program. Therefore, "inspecting" a server means comparing its loaded program with a clean version of the program, and "restoring" a server can be done by simply restoring the original program ("format and reinstall"). Once restored, the machine should be considered as any other honest machine; in particular, it may be corrupted again just like any other machine.

The problem posed by the above game is that for a fixed α, S wants to maximize γ while minimizing β. In the general case, one wants to maximize γ for any given α and β. Observe that, theoretically speaking, the maximum value of γ that can be attained is $(1-\alpha+\alpha\beta)/2$ (see Figure 2), which corresponds to half the honest servers among the ones originally clean plus half the ones that were reset to a clean state. For the special case of $\beta \to 0$, the theoretical maximum is $\gamma = (1-\alpha)/2$. Indeed, Theorem 1 implies that the service provider can examine a vanishing fraction of the servers and still run a successful MPC protocol amongst those inspected servers that were found clean, given that the adversary's corruption rate γ is below $(1-\alpha)/2$. However, this still does not show how to obtain the maximal γ for *any* choice of α, β, when the service provider is unaware of the identity of the $(1-\alpha)n$ honest servers.

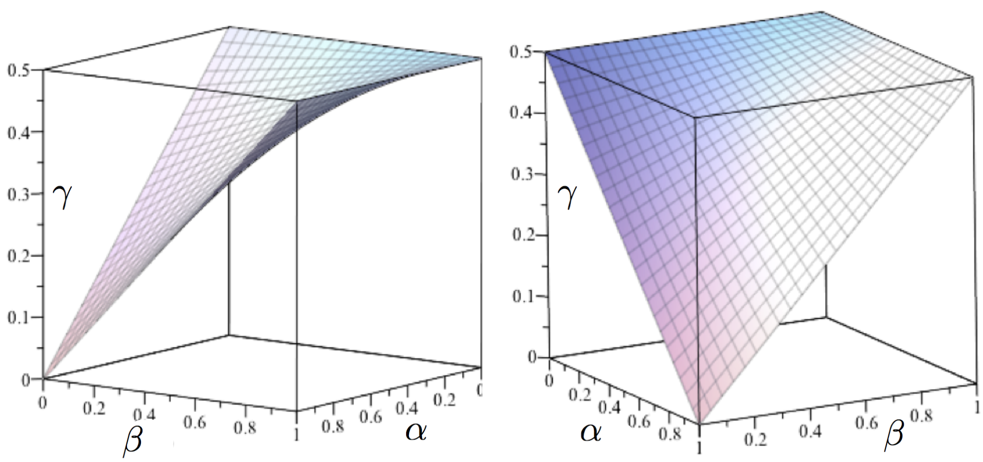

Fig. 2. The maximum adaptive corruption rate γ given α, β in the Corruption/Inspection Game.

Note that the above course of action for the service provider, where the first tier consists of only inspected servers which were uncorrupted, takes no advantage of the servers that were restored to a clean state. As mentioned above, those restored servers cannot be part of the first-tier since such servers would be detected by the adversary and hence the required indistinguishability between tiers would be violated. However, by performing a more sophisticated selection of servers which also exploits a random subset of restored servers, we can improve on the amount of adaptive corruption obtained by Theorem 1, and maximize γ and for any choice of α, β. Specifically:

Theorem 2 (Informal). *In the corruption/inspection game, for any constants $\alpha, \beta \in (0,1)$ and any constant $\epsilon > 0$, there exists a two-tier MPC protocol tolerating adaptive corruption rate $\gamma \le (1-\epsilon)\frac{1-\alpha+\alpha\beta}{2}$.*

That is, for any constants α, β, we achieve the maximal theoretical corruption rate of almost half the honest parties across the two tiers. This means our protocol tolerates a total corruption rate arbitrarily close to $\frac{1-\alpha+\alpha\beta}{2} + \alpha(1-\beta)$ across both tiers. Such a corruption rate is above $1/2$ for any $\alpha > 0$, surpassing the maximal corruption rate of the plain model.

The proof of Theorem 2 is slightly more complex than Theorem 1, as we now have to account for the fact that some information is leaked to the adversary in the network layer. The adversary can distinguish the cleaned servers from the remaining, therefore it can cluster the network layer in those two disjoint subsets. Nevertheless, we are able to apply a similar analysis as in Theorem 1 by observing that any adaptive corruption strategy effectively amounts to a partially randomized corruption strategy which can only control the cluster where the corruption action is directed to.

Related work. To our knowledge the two-tier model for MPC has not been considered in the literature. Our work is inspired by recent work on "resource-based corruptions" [GJKY13], in which corrupting a party (server) is assigned a cost. Different parties may have different corruption costs, and this information is hidden from the resource-bounded adversary. Due to being uninformed of such costs, the adversary is then "forced" to waste his budget on servers whose corruption cost is high. For a fixed adversarial budget, robustness in the hidden-cost model greatly outweighs robustness in the setting in which all parties have the same corruption cost.

Our anonymous message transmission notion, AAC, is related to (but distinct from) the notion of "secret handshakes" [BDS+03, CJT04]. Similarly to this notion, we work in a setting where a certain special action takes place between two parties if and only if they are both members of a hidden subset. If it happens that one party is not a member of the hidden subset, then it cannot infer the membership status of the other party. Our work is, to the best of our knowledge, the first application of such "covert subset" techniques in a setting where anonymity is not the prime objective. In fact, our work shows how anonymity can be effectively used to increase the robustness (specifically, the number of tolerated corruptions) of MPC, continuing to demonstrate the power of such tools. For example, it has been shown that variants of an anonymous channel can be used to implement unconditionally secure point-to-point channels and a broadcast channel [FGMO05, IKOS06], as well as more efficient natural secure computation tasks, such as private information retrieval (PIR) [IKOS06].

Roadmap. The paper is organized as follows. Notation, definitions, and the two-tier (TT) model for MPC are presented in Section 2. The TT MPC protocol, as well as the AAC (Anonymous yet Authentication Communication) notion and construction it relies on, are presented in Section 3. Finally, the analysis yielding the selection of the two tiers allowing to tolerate the maximal corruption rate appears in Section 4. Auxiliary definitions and constructions are presented in the appendix.

2 Model and Definitions

We let κ be the security parameter, and assume that any function, set size or running time implicitly depends on this parameter (especially when we write negl to describe a negligible function in κ—i.e., $\mathsf{negl} < 1/\mathsf{poly}(\kappa)$ for large enough κ). For any ε, we say that two distribution ensembles $\{X_\kappa\}_{\kappa\in\mathbb{N}}$, $\{Y_\kappa\}_{\kappa\in\mathbb{N}}$ are *ε-indistinguishable*, denoted $\{X_\kappa\} \approx_\epsilon \{Y_\kappa\}$, if for any probabilistic polynomial-time (PPT) algorithm C, for large enough κ,

$$|\Pr[C(1^\kappa, X_\kappa) = 1] - \Pr[C(1^\kappa, Y_\kappa) = 1]| < \epsilon + \mathsf{negl}(\kappa).$$

We say that X and Y are *computationally indistinguishable* and denote $\{X_\kappa\} \approx \{Y_\kappa\}$ if they are ε-indistinguishable with $\varepsilon = 0$. We now proceed to describe some of the cryptographic primitives and building blocks that we use throughout the paper.

Security of multiparty protocols. For defining security of a multiparty protocol for computing an n-ary function f, we follow the standard simulation-based approach [GMW87, Can00], in which the protocol execution is "compared" to an ideal protocol where the parties send their inputs to a trusted party who computes f and returns the designated output to each party. Commonly, the trusted-party activity for computing the function f is captured via a so-called ideal functionality $\mathcal{F}_f$.

Let $\mathsf{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(\kappa, \boldsymbol{x})$ denote an execution of the n-party protocol π with an adversary $\mathcal{A}$ and an environment $\mathcal{Z}$, with $\boldsymbol{x} = x_1, \ldots, x_n$ being the vector of inputs of the parties. In the same manner, define $\mathsf{IDEAL}_{\mathcal{F}_f,\mathcal{S},\mathcal{Z}}(\kappa, \boldsymbol{x})$ to be an execution in the ideal-model, where the ideal functionality is described by $\mathcal{F}_f$, $\mathcal{S}$ is the adversary (commonly known as *simulator*), $\mathcal{Z}$ is the environment, and $\boldsymbol{x}$ defined as above. We say that π *securely realizes* the functionality $\mathcal{F}_f$ if for every polynomial-time real-model adversary $\mathcal{A}$ and any PPT environment $\mathcal{Z}$, there is a polynomial time ideal-model simulator $\mathcal{S}$ such that for any input vector $\boldsymbol{x}$,

$$\{\mathsf{EXEC}_{\pi,\mathcal{A},\mathcal{Z}}(\kappa, \boldsymbol{x})\}_{\kappa\in\mathbb{N}} \approx_\varepsilon \{\mathsf{IDEAL}_{\mathcal{F}_f,\mathcal{S},\mathcal{Z}}(\kappa, \boldsymbol{x})\}_{\kappa\in\mathbb{N}}$$

where ε is a negligible function in the security parameter κ. Throughout this paper, we assume n and κ are polynomially related.

We refer the reader to Appendix A for additional standard definitions and other building blocks we use. We now describe the basics of the two-tier (TT) model for MPC.

The two-tier (TT) model for secure multiparty computation. There are n parties (servers) $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$, each of them identified by a name P_i, referred to as its *real* identity, and a "virtual" name from $\mathcal{P}^* = \{P_1^*, \ldots, P_n^*\}$, referred to as its *protocol pseudonym*, which identifies them as participants in the MPC protocol; all are probabilistic polynomial-time (PPT) machines. We assume a bijection $\nu : \mathcal{P} \to \mathcal{P}^*$ which maps a real identity P_i to its protocol pseudonym $\nu(P_i) \in \mathcal{P}^*$. The parties are assumed to know both their real name and pseudonym, but they do not know the specific ν.

The type of application we are interested in is secure function evaluation [GMW87] performed by the servers in $\mathcal{P}$. The inputs to the computation are assumed to be held by a set of clients, who are assumed to be outside the set $\mathcal{P}$. Each such client has an input x_i, and the goal is to compute a joint function f of the clients' inputs. Servers do not have an input of their own and they expect no output from the computation—their sole purpose is to carry out the computation and deliver the output back to clients.

As in the standard MPC setting, parties are connected by pair-wise authentic and reliable channels, which are identified by the *real* names of the two connected parties. Accessing this communication channel does not mandate the disclosure of the protocol pseudonyms of the communicating parties. We assume a synchronous communication model where a party can send a message to multiple parties at the same time [Can00].

The set of servers is divided into two disjoint sets $\mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2$—the first and second tier servers respectively. Our communication model assumes that the two tiers are indistinguishable at the communication (real name) layer. As mentioned in Section 1, the two tiers are subject to different adversarial capabilities with respect to corruption. Among the servers in $\mathcal{P}_2$, an unlimited number t_s of *static* corruptions are allowed. The servers in $\mathcal{P}_1$, on the other hand, are assumed to be uncorrupted at the onset of the computation. During the course of the computation, *all* servers are subject to adaptive corruptions; we denote the number of such corruptions by t_a. We assume a threshold corruption model, in which the adversary is restricted to corrupting at most t $(= t_\mathsf{s} + t_\mathsf{a})$ of the parties overall. At each step, the adversary may choose a party $P_i \in \mathcal{P}$ and corrupt it, as long as the total number of corrupted parties does not exceed his "budget" t. Once P_i gets corrupted, the adversary learns its internal state, including its tier level and protocol pseudonym $\nu(P_i)$.

We assume a standard public-key infrastructure (PKI) setup, in which each party P_i, $i \in \{1, \ldots, n\}$, is given *two* pairs of public/secret keys (pk_i, sk_i), (pk_j^*, sk_j^*) corresponding to its real name and protocol pseudonym, as well as the public keys of all other users (in a certified way) in the form $\{(pk_k, P_k)\}_{k\neq i}$ and $\{(pk_k^*, P_k^*)\}_{k\neq j}$. Note that the correspondence between names and protocol pseudonyms is not revealed. More formally, we express this as the parties having access to two instances of an ideal PKI functionality, denoted by $\mathcal{F}_{\mathsf{PKI}}^{\mathcal{P}}$ and $\mathcal{F}_{\mathsf{PKI}}^{\mathcal{P}^*}$ (see [Can05] for definition of an ideal PKI functionality). If $\nu : \mathcal{P} \to \mathcal{P}^*$ maps between real and protocol identities, we shorthand these two functionalities by $\mathcal{F}_{\mathsf{PKI}}^{\nu} = (\mathcal{F}_{\mathsf{PKI}}^{\mathcal{P}}, \mathcal{F}_{\mathsf{PKI}}^{\nu(\mathcal{P})=\mathcal{P}^*})$.

3 Secure Multiparty Computation in the TT Model

In this section we present our MPC protocol in the two-tier model, and obtain Theorem 1. As mentioned above, exploiting the indistinguishability between the two tiers requires new cryptographic tools that enable anonymous communication among servers. To this end, our construction assumes a communication capability which allows parties to communicate messages in an authenticated way but without compromising their real identity, which we term *Anonymous yet Authentic Communication* (AAC). Specifically, AAC allows entities to communicate with each other in an authenticated fashion at the protocol (application) layer, yet anonymously at the network (real-name) layer; the latter property comes from the fact that the correspondence between real and protocol names is hidden from the adversary and the functionality does not reveal it. We now define the ideal functionality of such a communication channel, and construct a protocol that securely realizes it.

3.1 The $\mathcal{F}_{\mathsf{AAC}}^{\nu}$ ideal functionality

In the ideal world, the sender delivers to the functionality the message μ along with the protocol pseudonym of the intended receiver . The adversary is notified of this event and receives the pseudonyms of the two communicating entities. However, the real names of the two entities remain hidden. The functionality is parameterized by a mapping ν that gives the correspondence between names and pseudonyms. When the adversary instructs the functionality to deliver the message, the functionality recovers the real identity of the receiving entity and writes the message on its network tape along with the protocol pseudonym of the sender. We formally describe the functionality in Figure 3.

Functionality $\mathcal{F}_{\mathsf{AAC}}^{\nu}$

$\mathcal{F}_{\mathsf{AAC}}^{\nu}$ is parameterized by a security parameter κ and a set of n parties with real names $\mathcal{P}$, protocol pseudonyms $\mathcal{P}^*$, and a bijection $\nu : \mathcal{P} \to \mathcal{P}^*$; it assumes a message space $\mathcal{M} = \mathcal{M}(\kappa)$ and proceeds as follows, running with parties $P_1, ..., P_n \in \mathcal{P}$ and an adversary $\mathcal{S}$:

- Upon receiving $(\mathsf{Send}, sid, P_i, \mu, P_j^*)$ from P_i, record this tuple. Once a message is recorded from all honest parties send to the adversary $\mathcal{S}$ the sequence of tuples $(\mathsf{SendLeak}, sid, \nu(P_i), \mu, P_j^*)$ lexicographically ordered.
- Upon receiving $(\mathsf{Deliver}, sid)$ from $\mathcal{S}$, check that a Send message was recorded on behalf of all senders, and if so, for any recorded tuple of the form $(\mathsf{Send}, sid, P_i, \mu, P_j^*)$, deliver $(\mathsf{Sent}, sid, \nu(P_i), \mu)$ to party $\nu^{-1}(P_j^*)$.
- Upon receiving (Abort, sid, A) from $\mathcal{S}$ check that A is a non-empty subset of corrupted parties and forward this message to all honest parties.
- Upon receiving $(\mathsf{Corrupt}, P_i)$ from $\mathcal{S}$ mark P_i as corrupted and return $\nu(P_i)$ to $\mathcal{S}$.

If P_i is corrupted then $\mathcal{S}$ is allowed to submit $(\mathsf{Send}, sid, P_i, \mu, P_j^*)$ on behalf of P_i.

Fig. 3. *Ideal functionality for anonymous yet authentic communication (AAC).*

We now show how this functionality can be securely realized assuming an *anonymous broadcast channel* functionality (cf. [Cha88]) tolerating an arbitrary number of corrupted parties[3]. Recall that such functionality can be thought of as a bulletin board on which any party can post messages without revealing its identity. This is modeled as the ideal functionality $\mathcal{F}_{\mathsf{ABC}}$ in Figure 4, which we later on show how to implement assuming a PKI setup.[4]

Functionality $\mathcal{F}_{\mathsf{ABC}}$

The functionality assumes a message space $\mathcal{M} = \mathcal{M}(\kappa)$ with κ being the security parameter, and works as follows, running with n parties $P_1, \ldots, P_n$ and an adversary $\mathcal{S}$:

- Upon receiving $(\mathsf{AnonBcast}, sid, P_i, \mu)$ from P_i record this tuple. Once a message is recorded for all honest parties send to the adversary $\mathcal{S}$ the message $(\mathsf{AnonBcastLeak}, sid, M)$ where M is the (lexicographically ordered) set of messages μ from all recorded tuples of the form $(\mathsf{AnonBcast}, sid, P_i, \mu)$.
- Upon receiving $(\mathsf{Deliver}, sid)$ from $\mathcal{S}$, ignore further AnonBcast messages, and deliver $(\mathsf{AnonBcastDeliver}, sid, M')$ to all parties $P_1, \ldots, P_n$ where M' is the set of messages μ (lexicographically ordered) from all recorded tuples of the form $(\mathsf{AnonBcast}, sid, P_i, \mu)$.
- Upon receiving (Abort, sid, A) from $\mathcal{S}$ check that A is a non-empty subset of corrupted parties and forward this message to all honest parties.
- Upon receiving $(\mathsf{Corrupt}, P_i)$ from $\mathcal{S}$ mark P_i as corrupted and return the recorded $(\mathsf{AnonBcast}, sid, P_i, \mu)$ to $\mathcal{S}$.

If P_i is corrupted then $\mathcal{S}$ is allowed to submit (or substitute existing) $(\mathsf{AnonBcast}, sid, P_i, \mu)$ messages on behalf of P_i.

Fig. 4. *Ideal anonymous broadcast channel functionality (ABC).*

Using the ideal functionality $\mathcal{F}_{\mathsf{ABC}}$ and the PKI setting described in Section 2, we now describe a secure realization of $\mathcal{F}^{\nu}_{\mathsf{AAC}}$. The protocol makes use of an existentially unforgeable digital signature scheme (cf. Appendix A). The implementation is rather straightforward: the sender uses the (protocol layer) PKI to sign the message and anonymously broadcast it. Any party receiving the message checks whether it is the intended protocol-layer recipient, and if so, it verifies the signature and decrypts the message. This approach prevents impersonation at the protocol layer while still hiding the correspondence between protocol names and real names.

The AAC protocol is described in Figure 5 and operates in the $(\mathcal{F}^{\nu}_{\mathsf{PKI}}, \mathcal{F}_{\mathsf{ABC}})$-hybrid world.

Theorem 3. *Let $n \in \mathbb{N}$, parties $\mathcal{P}$ with protocol pseudonyms $\mathcal{P}^*$ and a bijection $\nu : \mathcal{P} \to \mathcal{P}^*$. The* AAC *protocol from Figure 5 securely realizes $\mathcal{F}^{\nu}_{\mathsf{AAC}}$ against an adaptive adversary corrupting $t < n$ parties in the $(\mathcal{F}^{\nu}_{\mathsf{PKI}}, \mathcal{F}_{\mathsf{ABC}})$-hybrid model.*

Proof (sketch). Consider a PPT adversary $\mathcal{A}$ and a PPT environment $\mathcal{Z}$. We use the notation $\mathcal{M}$ to denote the space of all messages. We construct a simulator $\mathcal{S}$ so that for every vector of inputs $\boldsymbol{x} = x_1, \ldots, x_n$ with $x_i \in \{(\mathsf{Send}, sid, P_i, \mu, P^*_j) \mid j \in \{1, \ldots, n\}, \mu \in \mathcal{M}\}$, for $i = 1, \ldots, n$, the following holds:

$$\mathsf{EXEC}^{\mathcal{F}^{\nu}_{\mathsf{PKI}}, \mathcal{F}_{\mathsf{ABC}}}_{\mathsf{AAC}, \mathcal{A}, \mathcal{Z}}(\kappa, \boldsymbol{x}) \approx \mathsf{IDEAL}_{\mathcal{F}^{\nu}_{\mathsf{AAC}}, \mathcal{S}, \mathcal{Z}}(\kappa, \boldsymbol{x})$$

where $\nu : \mathcal{P} \to \mathcal{P}^*$ is a random bijection. As a setup step, $\mathcal{S}$ generates keys for all the identities in $\mathcal{P}^*$, and gives $\mathcal{A}$ all the public keys. The simulator maintains a list M which is empty at initialization.

[3] We remark that performing an AAC message delivery means that in case the AAC protocol terminates with abort, the protocol is repeated with a subset of parties currently not marked as corrupt.

[4] We note that (most) security proofs in Canetti's synchronous model [Can00] carry over to the *Universal Composability* framework [Can05], given that certain functionalities are available to the protocol [KMTZ13].

Protocol AAC

Setup: Assume a ecurity parameter κ, and let $(\mathsf{GenS}, \mathsf{Sig}, \mathsf{Ver})$ be an existentially unforgeable signature scheme. The PKI delivers real-layer keys (pk_i, sk_i) and protocol-layer keys (pk_i^*, sk_i^*) as described in Section 2. Each pair of keys is generated using $\mathsf{GenKey}(1^\kappa) = (\mathsf{GenE}(1^\kappa), \mathsf{GenS}(1^\kappa))$.
We assume real names $\mathcal{P}$ and protocol pseudonyms $\mathcal{P}^*$ are known to all entities (but not ν).

Send message: On input $(\mathsf{Send}, sid, P_i, \mu, P_j^*)$ the sender party P_i sends $(\mathsf{AnonBcast}, sid, P_i, (P_i^*, P_j^*, \mu, \sigma))$ to $\mathcal{F}_{\mathsf{ABC}}$ where $\sigma \leftarrow \mathsf{Sig}_{sk_i^*}(P_j^*, \mu, sid)$.

Receive message: P_j, $1 \leq j \leq n$, upon receiving $(\mathsf{AnonBcast}, sid, M)$ from $\mathcal{F}_{\mathsf{ABC}}$, if it holds that $P_j^* = B$ for some $(A, B, \mu, \sigma) \in M$ (i.e., P_j is the intended protocol level receiver of that message), P_j checks $\mathsf{Ver}_{pk_i^*}(P_j^*, \mu, sid, \sigma)$ where i is such that $A = P_i^*$ and provided Ver returns 1 it records (P_i^*, μ). The action terminates by returning all recorded tuples.

Abort: If $\mathcal{F}_{\mathsf{ABC}}$ returns (Abort, sid, A) then terminate and return A.

Fig. 5. *A protocol realizing* $\mathcal{F}^{\nu}_{\mathsf{AAC}}$.

The simulation is straightforward: when $\mathcal{S}$ receives $(\mathsf{SendLeak}, sid, P_i^*, \mu, P_j^*)$ from $\mathcal{F}^{\nu}_{\mathsf{AAC}}$ it generates a signature $\sigma = \mathsf{Sig}_{sk_i^*}(P_j^*, \mu, sid)$ and updates the list $M = M \cup (P_i^*, P_j^*, \mu, \sigma)$. Once $\mathcal{S}$ processes the $\mathsf{SendLeak}$ message for all the honest parties, it sends $(\mathsf{AnonBcastLeak}, sid, M)$ over to $\mathcal{A}$. If $\mathcal{A}$ issues an Abort message, $\mathcal{S}$ forwards the abort to $\mathcal{F}^{\nu}_{\mathsf{AAC}}$. Otherwise, $\mathcal{A}$ issues a $\mathsf{Deliver}$ message which is also forwarded to $\mathcal{F}^{\nu}_{\mathsf{AAC}}$.

When the adversary requests to corrupt some party P_i, the simulator forwards the request to $\mathcal{F}^{\nu}_{\mathsf{AAC}}$ and learns P_i's protocol pseudonym $\nu(P_i)$. Next, it forms the inner state of P_i accordingly (that contains the signing key of pseudonym $\nu(P_i)$), and delivers this information to $\mathcal{A}$. It is clear that the honest parties' output is identically distributed between the real and ideal executions, with the exception of the event that the adversary $\mathcal{A}$ (or the environment $\mathcal{Z}$) forges a signature on behalf of an honest party. In this case the simulator will fail, but this will happen with negligible probability based on the security of the underlying digital signature scheme. □

There are several possible ways to realize $\mathcal{F}_{\mathsf{ABC}}$ so that up to $t < n$ corruptions can be tolerated assuming our setup configuration (PKI). We consider some alternatives in Appendix B.

3.2 Pseudonymity and randomized corruptions

With forthsight, the approach we will follow is to replace every communication in an (adaptively secure) MPC protocol for the standard setting with an invocation to $\mathcal{F}^{\nu}_{\mathsf{AAC}}$. We now show that if a protocol π that operates at the pseudonym layer is unaware of the real/protocol name correspondence, then the approach does not reveal any information about the mapping ν. (In addition, it is straightforward to verify that the modified protocol would remain correct, i.e., it produces the same outputs as π.)

Let π be a protocol defined over the "pseudonym" protocol layer, i.e., running with parties $P_1^*, \ldots, P_n^*$. Further, π operates in (synchronous) communication rounds. Normally, in an execution of π with an adversary $\mathcal{A}$, an environment $\mathcal{Z}$ and parties $P_1^*, \ldots, P_n^*$, $\mathcal{A}$ is capable of issuing $(\mathsf{Corrupt}, P_i^*)$ messages when it wants to corrupt party P_i^*.[5] We consider a stronger notion of execution, denoted rcEXEC, in which the adversary is allowed to issue $(\mathsf{Corrupt})$ requests to a corruption oracle, upon which a *randomly* chosen honest party gets corrupted. We call this an execution with *randomized* corruptions. Note that $\mathsf{rcEXEC}_{\pi,\mathcal{A},\mathcal{Z}}$ is the ensemble of views over the

[5] We emphasize that since π exists only in the pseudonym layer, the parties' identifiers are $\mathcal{P}^*$, and the adversary corrupts by specifying a certain P_i^*. However, when running in our TT model setup, the identities of the parties are $\mathcal{P}$, and the adversary corrupts a party by specifying a certain P_i.

adversary's (and environment's) coin tosses, the parties' coin tosses *and* the randomness of the corruption oracle.

Now consider the setting where the communication is handled by a lower "physical" layer where each party has a physical (real) identity $P_1, \dots, P_n$ and there is a mapping $\nu : \mathcal{P} \to \mathcal{P}^*$ that corresponds protocol identities to communication identities (real names). Given any protocol π that operates in rounds, we can easily obtain a protocol $\tilde{\pi}^{\mathcal{F}^{\nu}_{\mathrm{AAC}}}$ that runs with parties $P_1, \dots, P_n$ and whenever π, acting on behalf of P_i^*, wishes to send a message μ to party P_j^* the $\tilde{\pi}$ protocol delivers $(\mathsf{Send}, sid, \nu^{-1}(P_i^*), \mu, P_j^*)$ to $\mathcal{F}^{\nu}_{\mathrm{AAC}}$. Thus, each communication round of π is equivalent to a single instantiation of $\mathcal{F}^{\nu}_{\mathsf{AAC}}$.

Next, we show that $\tilde{\pi}^{\mathcal{F}^{\nu}_{\mathsf{AAC}}}$ with a randomly chosen ν is simulatable in the randomized-corruptions setting. For ease of notation, we identify a bijection $\nu : \mathcal{P} \to \mathcal{P}^*$ with a permutation on n elements.

Lemma 4. *Let π and $\tilde{\pi}^{\mathcal{F}^{\nu}_{\mathsf{AAC}}}$ be as above. For any PPT adversary $\mathcal{A}$ and environment $\mathcal{Z}$, and for any input vector $\boldsymbol{x}$, there exist a PPT simulator $\mathcal{S}$ such that*

$$\left\{\mathsf{EXEC}^{\mathcal{F}^{\nu}_{\mathsf{AAC}}}_{\tilde{\pi},\mathcal{A},\mathcal{Z}}(\kappa, \boldsymbol{x}^{\nu})\right\}_{\nu \in_R \mathrm{Perm}(n)} \approx \mathsf{rcEXEC}_{\pi,\mathcal{S},\mathcal{Z}}(\kappa, \boldsymbol{x})$$

where $\mathrm{Perm}(n)$ *is the set of all the possible permutations on n elements.*

Proof. Consider the following simulator. At first it fixes a randomness tape for $\mathcal{A}$ and follows the computation, replacing each "communication round" with a $\mathcal{F}^{\nu}_{\mathsf{AAC}}$ simulation. Namely, after each round of communication, $\mathcal{S}$ gathers all the messages sent in this round, and provides the adversary with a lexicographical list whose entries are of the form $(\mathsf{SendLeak}, sid, P_i^*, \mu, P_j^*)$ matching the case where P_i^* sent P_j^* the message μ. Note that we assume π runs in rounds, so each party sends exactly one message at each communication round.

When $\mathcal{A}$ issues $(\mathsf{corrupt}, P_i)$, the simulator issues $(\mathsf{corrupt})$ and as a result, P_j^* gets corrupted, for a random j (out of all the parties that are still honest). The simulator sets $\nu(i) = j$ and simulates the inner state of P_j^* so it would correspond to the real identity P_i in a straightforward way.

Note that at the end of the simulation, the simulator has defined a partial mapping ν. The output of this simulation is exactly the same as the output of any instance of the left-hand side experiment, running with the same adversary (set to the same randomness tape), for any mapping ν' that agrees with ν on the identities of all corrupted parties. It easily follows that the two ensembles are identically distributed. ☐

3.3 The TT MPC protocol

Recall our setting in which n parties (servers) with real names $\mathcal{P} = \{P_1, P_2, \dots, P_n\}$, are split into two tiers $\mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2$, and where the computation is effectively carried out only by servers in the first tier. Let $|\mathcal{P}_1| = m$ and $|\mathcal{P}_2| = n - m$. In addition, we assume there are $c \in \mathbb{N}$ clients, each holding a private input x_i; let $\boldsymbol{x} = x_1, \dots, x_c$. The clients wish to compute some function f of their inputs, described as the c-party functionality $\mathcal{F}_f(\boldsymbol{x})$.

We now describe the two-tier MPC protocol performed by the servers, *assuming they have already (verifiably) secret-shared*[6] *the clients' inputs*. This operation is in fact easy to achieve using standard techniques and without the need for AAC communication. For example, one may assume that the i-th client computes an $(m, \lceil m/2 \rceil - 1)$-verifiable secret sharing of x_i using the adaptively secure VSS scheme of Abe and Fehr [AF04]. Then, the client broadcasts a signed copy of the j-th share encrypted with P_j^*'s public key. (Recall that protocol identities, and in particular those

[6] Refer to Appendix A for the definition of VSS.

corresponding to servers in $\mathcal{P}_1$, are public.) As a result of the computation, the servers obtain a *share* of $\mathcal{F}_f(\boldsymbol{x})$'s output—we denote this modified functionality by $\mathcal{F}_f^{\mathsf{vss}}(\boldsymbol{x})$; the shares are then sent to the clients.[7]

We now explain how the servers carry out the actual computation of $\mathcal{F}_f(\boldsymbol{x})$. The two-tiered MPC protocol operates in the $(\mathcal{F}_{\mathsf{PKI}}^{\nu}, \mathcal{F}_{\mathsf{AAC}}^{\nu})$-hybrid world and is presented in Figure 6.

MPC in the Two-Tier Model

Assume n parties with real names $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$, split into two disjoint subsets $\mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2$, where $|\mathcal{P}_1| = m$. Parameters n and m are public. Furthermore, assume a c-ary functionality $\mathcal{F}_f(\boldsymbol{x})$ to be securely computed on inputs $\boldsymbol{x} = x_1 \ldots x_c$, where each x_i is $(m, \lceil m/2 \rceil - 1)$-VSSed in $\mathcal{P}_1$.

Trusted setup. Public and secret keys, as well as protocol identities are handed to each party by $\mathcal{F}_{\mathsf{PKI}}^{\nu}$, as described in Section 2.

Computation phase.

- Let $\mathcal{F}_f^{\mathsf{vss}}$ be the m-party functionality that performs the same task as the c-party functionality $\mathcal{F}_f$, assuming that each of the m parties holds a share of each of the c inputs.
  The output of $\mathcal{F}_f^{\mathsf{vss}}$ is a $(m, \lceil m/2 \rceil - 1)$-VSS share of each of the c outputs of $\mathcal{F}_f$.
- The parties in $\mathcal{P}_1$ adaptively securely compute $\mathcal{F}_f^{\mathsf{vss}}$ amongst themselves (for example, via [CFGN96]). During the execution, messages between any two parties are sent invoking $\mathcal{F}_{\mathsf{AAC}}^{\nu}$ (Fig. 3).

Fig. 6. *Computation phase of the TT MPC protocol.*

It is immediate that the protocol in Figure 6 securely realizes $\mathcal{F}_f^{\mathsf{vss}}$ as long as the adversary does not corrupt a majority of the tier-1 servers. Formally,

Proposition 5. *Let $n, m, c \in \mathbb{N}$. For any given c-ary functionality $\mathcal{F}_f$ and for any bijection $\nu : \mathcal{P} \to \mathcal{P}^*$, the protocol of Figure 6 operating in the $(\mathcal{F}_{\mathsf{PKI}}^{\nu}, \mathcal{F}_{\mathsf{AAC}}^{\nu})$-hybrid world securely realizes $\mathcal{F}_f^{\mathsf{vss}}$ conditioned on the event that the adversary corrupts at most $\lceil m/2 \rceil - 1$ servers.*

Next, we prove a combinatorial lemma, showing that for any $\epsilon > 0$ and t_{s} initial static corruptions among the tier-2 servers, if an adversary adaptively corrupts up to $(1-\epsilon)\frac{n-t_{\mathsf{s}}}{2}$ parties without knowing the two-tier partition, then the probability of corrupting a majority of $\mathcal{P}_1$ servers is negligible in $|\mathcal{P}_1|$.

Lemma 6. *Assume n parties $\mathcal{P}$, m of which are in $\mathcal{P}_1$ and $t_{\mathsf{s}} \leq n - m$ of $\mathcal{P}_2$ are initially corrupted. Assume that the adversary is bounded to adaptively corrupting t_{a} parties with $t_{\mathsf{a}} \leq (1-\epsilon)\frac{n-t_{\mathsf{s}}}{2}$, for some constant $\epsilon > 0$, where $\epsilon \cdot m \geq 2$. Furthermore, assume that by corrupting a party $P_i \in \mathcal{P}$, the adversary learns its tier level (but not the tier level of other parties). Then, the probability that adversary corrupts at least $m/2$ parties from $\mathcal{P}_1$ is at most $2^{-\Omega(m)}$.*

Proof. Let K be the random variable describing the number of $\mathcal{P}_1$ servers that were corrupted, assuming the adversary corrupts additional $t_{\mathsf{a}} = (1-\epsilon)\frac{n-t_{\mathsf{s}}}{2}$ servers (i.e., on top of statically-corrupting t_{s} parties). K is distributed according to the Hypergeometric distribution (see Appendix C) with parameters $(n - t_{\mathsf{s}}, m, t_{\mathsf{a}})$, and we denote $K \sim \mathsf{HypGeo}_{n-t_{\mathsf{s}},m,t_{\mathsf{a}}}$. We get that

$$\mathbb{E}[K] = (1-\epsilon)\frac{n-t_{\mathsf{s}}}{2} \cdot \frac{m}{n-t_{\mathsf{s}}} = (1-\epsilon)\frac{m}{2}.$$

[7] We note that in case the identity of the first-tier servers needs to remain hidden (say, for the continuation of the service in a forthcoming MPC execution) the output delivery should be done anonymously as well. This can be easily achieved, for example, by extending the AAC mechanism to include both servers and clients at the protocol layer.

Assuming that m is odd (the case of an even m is similar) we can use the tail bound of Lemma 10 to bound the probability that more than $m/2$ servers get corrupted.

$$\begin{aligned}\Pr[K > m/2] &= \Pr[K - \mathbb{E}[K] > \epsilon m/2] \\ &< e^{-2\frac{n-t_{\mathsf{s}}+2}{4(m+1)(n-t_{\mathsf{s}}-m+1)}(\epsilon^2 m^2 - 1)} \\ &\leq 2^{-\Omega(\frac{n-t_{\mathsf{s}}}{n-t_{\mathsf{s}}-m+1} m)} \\ &= 2^{-\Omega(m)},\end{aligned}$$

since in our case $\alpha_{n,m,t}$ of Lemma 10 satisfies $\alpha_{n,m,t} \geq \frac{n+2}{(m+1)(n-m+1)}$, and assuming $\epsilon m \geq 4$. ◻

Theorem 1. *Assume $m = \omega(\log n)$ and $\epsilon > 0$. For any given c-ary functionality $\mathcal{F}_f$, there exists a two-tier MPC protocol in the $(\mathcal{F}^{\nu}_{\mathsf{PKI}}, \mathcal{F}^{\nu}_{\mathsf{AAC}})$-hybrid world that securely realizes $\mathcal{F}_f$ against any PPT adversary with $t_{\mathsf{a}} \leq (1-\epsilon)\frac{n-t_{\mathsf{s}}}{2}$ and $t_s \leq n - m$.*

Proof. Observe that (i) the MPC protocol is secure as long as a majority of $\mathcal{P}_1$ are honest (Proposition 5); (ii) given that the adversary learns the protocol pseudonym and tier-level of a party only when this party is corrupt, when restricted to $(1-\epsilon)\frac{n-t_{\mathsf{s}}}{2}$ corruptions, it has only an exponentially-small probability (in m) to corrupt a majority of $\mathcal{P}_1$ (Lemma 6); (iii) By Lemma 4 an adaptive adversary learns only negligible information about ν (for uncorrupt parties), that is, it does not have an advantage in learning the protocol identity (i.e., tier-level) of uncorrupted parties from the transcript. Therefore, an adaptive adversary has exponentially-small probability (in m) to break the the protocol of Figure 6. Setting $m = \omega(\log n)$ makes the adversary's success probability negligible in n. ◻

4 Optimal Strategy for the Corruption/Inspection Game

To conclude, we present the analysis for the Corruption/Inspection Game (Figure 1). We obtain, for any parameters (α, β), a strategy that maximizes γ up to the theoretical limit. In the previous sections we demonstrated that, *given a two-tier model*, MPC can be realized to resist as much corruptions as less than half the amount of the still-honest parties. However, it is left to be shown *how* to split the n servers into two tiers so that (i) the two tiers are indistinguishable and (ii) the tier-1 servers are honest (at the onset of the computation).

As mentioned in the Introduction, one possible strategy for the service provider S is to set as tier-1 all the servers that were inspected and found clean. However, S cannot use the servers which were found corrupt, as these are no longer indistinguishable from the honest servers. This strategy leads to a non-optimal adaptive corruption rate of $\gamma = \frac{1-\alpha}{2}$. Thus, better strategies should be sought in order to utilize the "restored" machines. Next, we show a strategy for the service provider which maximizes his utility in the Corruption/Inspection Game. Specifically, we prove the following:

Theorem 2. *For any constants $\alpha, \beta \in (0,1)$, and for any $\varepsilon > 0$, there exists a two-tier MPC protocol in the $(\mathcal{F}^{\nu}_{\mathsf{PKI}}, \mathcal{F}^{\nu}_{\mathsf{AAC}})$-hybrid world, and a winning strategy for a service provider in the Corruption/Inspection Game, such that the protocol is adaptively secure against any PPT adversary with corruption rate $\gamma \leq (1-\varepsilon)(1-\alpha+\alpha\beta)/2$.*

We begin by showing a strategy for the service provider that beats any adversary who learns the tier-level of honest parties only by corrupting them. The idea of the strategy is to use two sets of servers as tier-1. One set comprises all the servers that were inspected and found clean, while the second one is a subset of the servers that were restored to a clean state. Note that, from the point of view of the adversary, the first set is hidden within all the uncorrupt servers, while the second set is hidden within all the servers that were restored to a clean state. We set the size of the second group so that in both these sets, the ratio of tier-1 servers to the size of the set it is hidden within, is the same.

Lemma 7. *Assume S and $\mathcal{A}$ play the Corruption/Inspection Game with some constants $\alpha, \beta \in (0,1)$ and a small constant $\varepsilon > 0$. Furthermore, assume that when a server becomes corrupt (and only then), the adversary learns its tier level. Then, there exists a strategy for S for choosing tier-1 servers, such that given a corruption rate $\gamma \leq (1-\varepsilon)(1-\alpha+\alpha\beta)/2$ the adversary has negligible probability to corrupt half (or more) of tier-1 servers.*

Proof. S will choose the tier-1 servers as a subset of the βn inspected servers. We distinguish between two groups of inspected servers according to their state before the inspection: servers that were uncorrupt before the inspection (denoted G_1), and servers that were corrupt but recovered to a safe state by the inspection (G_2). From the point of view of $\mathcal{A}$, The first group is 'hidden' within the set $\widehat{G_1}$ of size $(1-\alpha)n$ of the uncorrupt servers at the onset. The second group is fully known to the adversary ($\widehat{G_2} = G_2$ with $\alpha\beta n$ servers[8]). S will to pick a small subset of servers in G_2 as tier-1; these will be hidden within the entire $\widehat{G_2}$. Note that the adversary knows which servers belong in $\widehat{G_1}$ and which are in $\widehat{G_2}$, but doesn't know the tier level of each party within each set. That way, the indistinguishability requirement between tier-1 and tier-2 servers still holds, yet separately in $\widehat{G_1}$ and $\widehat{G_2}$.

Specifically, S chooses tier-1 servers in the following way: all the $(1-\alpha)\beta n$ servers in G_1 are chosen as tier-1 in addition to a random subset of servers in G_2. We equalize the fraction of tier-1 servers in both groups (with respect to the group it is 'hidden' within). Thus, out of the $\alpha\beta n$ servers in G_2, S randomly picks $y = \alpha\beta^2 n$ servers to be tier-1, so that

$$\frac{y}{\alpha\beta n} = \frac{(1-\alpha)\beta n}{(1-\alpha)n}.$$

We allow the adversary to corrupt at most $t = (1-\varepsilon)(1-\alpha+\alpha\beta)n/2$ servers out of the uncorrupt servers $\widehat{G_1} \cup \widehat{G_2}$. Assume the adversary splits his budget so that it corrupts t_1 servers from $\widehat{G_1}$, and t_2 servers from $\widehat{G_2}$, where $t_1 + t_2 = t$.[9]

Let $r = t_1/t$ (thus, $1-r = t_2/t$); observe that the adversary cannot spend more budget than the population of each set so $t_1 \leq (1-\alpha)n$ and $t_2 \leq \alpha\beta n$, hence $1 - \frac{\alpha\beta}{t} \leq r \leq \frac{1-\alpha}{t}$. Let K_1, K_2 be the random variables that describe the number of servers $\mathcal{A}$ adaptively corrupts out of $\widehat{G_1}$ and $\widehat{G_2}$ respectively, with budget t_1, t_2 respectively. It is clear that

$$K_1 \sim \mathsf{HypGeo}_{(1-\alpha)n,(1-\alpha)\beta n,t_1}, \qquad K_2 \sim \mathsf{HypGeo}_{\alpha\beta n,\alpha\beta^2 n,t_2}.$$

In order to win the game, $\mathcal{A}$ needs to corrupt at least of half of the tier-1 servers, where some can be in $\widehat{G_1}$ and the rest in $\widehat{G_2}$. However no matter how $\mathcal{A}$ splits its budget, $\mathcal{A}$ corrupts more than half of the overall tier-1 servers with only a negligible probability. To that end we use a tail bound

[8] The mentioned sizes of the groups are only their *expected value*. However for large enough n (and especially, for our asymptotical analysis where $n \to \infty$), with high probability the real size will be very close to the expected value and we treat those sets as having sizes exactly $(1-\alpha)n$ and $\alpha\beta n$, etc.

[9] While we assume fixed values t_1 and t_2, in general the attack might be of any arbitrary distribution among the two sets. However, for any such attack we can repeat the analysis with t_1 being the expected number of servers corrupted out of $\widehat{G_1}$, and the two analyses differ with negligible probability when $n \to \infty$.

on the Hypergeometric distribution (see Appendix C, Lemma 10). Specifically, the probability that the adversary corrupts, out of $\widehat{G_1}$, at least an r-fraction of *half* of all the tier-1 servers, is negligible:

$$\begin{aligned}
\Pr\left[K_1 > r\tfrac{1}{2}((1-\alpha)\beta+\alpha\beta^2)n\right] &= \Pr\left[K_1 > t_1\beta\frac{(1-\alpha+\alpha\beta)n}{2t}\right] \\
&= \Pr\left[K_1 > \frac{1}{1-\varepsilon}\mathbb{E}[K_1]\right] \\
&= \Pr\left[K_1 > (1+\varepsilon')\mathbb{E}[K_1]\right] \\
&< e^{-\Omega(\beta t)} = e^{-\Omega(n)}
\end{aligned}$$

Where the second transition follows from $\mathbb{E}[K_1] = t_1\frac{(1-\alpha)\beta n}{(1-\alpha)n} = t_1\beta$.

In a similar way for $\widehat{G_2}$, the probability that $\mathcal{A}$ corrupts more than $1-r$ fraction of half of tier-1 servers is negligible:

$$\begin{aligned}
\Pr\left[K_2 > (1-r)\tfrac{1}{2}((1-\alpha)\beta+\alpha\beta^2)n\right] &= \Pr\left[K_2 > t_2\beta(1-\alpha+\alpha\beta)n/2t\right] \\
&= \Pr\left[K_2 > \frac{1}{1-\varepsilon}\cdot\mathbb{E}[K_2]\right] \\
&= \Pr\left[K_2 > (1+\varepsilon')\mathbb{E}[K_2]\right] \\
&< e^{-\Omega(n)}
\end{aligned}$$

It follows that there is a negligible probability for the adversary to corrupt at least

$$(r+(1-r))\cdot\tfrac{1}{2}((1-\alpha)\beta+\alpha\beta^2)n$$

tier-1 servers, and since the total number of tier-1 servers is $((1-\alpha)\beta+\alpha\beta^2)n$, S wins the game. □

Since the tier-1 servers are now split into two separate sets, we need to extend Lemma 4 to the case where ν is not uniform over $\mathrm{Perm}(n)$. Specifically, we assume now that $\{P_1,\ldots,P_n\}$ are partitioned into r disjoint sets, $\mathcal{P}_1,\ldots,\mathcal{P}_r$, with respective sizes $s_1,\ldots,s_r$, such that $\sum_{i=1}^r s_i = n$. Additionally, assume the protocol pseudonyms $\mathcal{P}^*$ are also partitioned into r disjoint sets $\mathcal{P}^*_1,\ldots,\mathcal{P}^*_r$ where for every $1\le i\le r$, $|\mathcal{P}_i| = |\mathcal{P}^*_i|$. We assume that the mapping ν is composed of r independent uniform permutations on the specific partitions. That is $\nu = (\nu_1,\ldots,\nu_r)$ where $\nu_i : \mathcal{P}_i \to \mathcal{P}^*_i$. For notational convenience, we also treat ν_i as a permutation on $\{1,\ldots,s_i\}$.

We show that even in this setting, where the adversary has some partial knowledge on ν, his best corruption strategy is equivalent to corrupting a random party. To that end, we re-define rcEXEC to be such that the simulator is allowed to choose the set from which the next party will be corrupted. That is, $\mathcal{S}$ may issue $(\mathsf{corrupt}, i)$ in which a random honest party in $\mathcal{P}_i$ will get corrupted. We denote an execution of this model as $\mathsf{rc}_r\mathsf{EXEC}$.

Lemma 8. *Let π and $\tilde{\pi}^{\mathcal{F}^{\nu}_{\mathsf{AAC}}}$ be as above. Assume the parties are divided into r sets $\mathcal{P}_1,\ldots,\mathcal{P}_r$ of sizes $s_1,\ldots,s_r$. For any PPT adversary $\mathcal{A}$ and environment $\mathcal{Z}$, and for any input vector $\boldsymbol{x}$, there exist a PPT simulator $\mathcal{S}$ such that*

$$\left\{\mathsf{EXEC}^{\mathcal{F}^{\nu}_{\mathsf{AAC}}}_{\tilde{\pi},\mathcal{A},\mathcal{Z}}(\kappa,\boldsymbol{x}^{\nu})\right\}_{\substack{\nu=\ (\nu_1,\ldots,\nu_r)\\ \in_R(\mathrm{Perm}(s_1),\ldots,\mathrm{Perm}(s_r))}} \approx \mathsf{rc}_r\mathsf{EXEC}_{\pi,\mathcal{S},\mathcal{Z}}(\kappa,\boldsymbol{x}),$$

where Perm(k) *is the set of all the possible permutations on k elements.*

Proof. The simulation performs similarly to the one of Lemma 4, with the following exception. When the adversary issues $(\mathsf{corrupt}, P_i)$, $\mathcal{S}$ will issue $(\mathsf{corrupt}, \mathsf{k})$ for the set k such that $P_i \in \mathcal{P}_k$. Assume that as a result P_j^* becomes corrupt, then $\mathcal{S}$ sets $\nu_k(i) = j$ and continues as before. Once again, the output of the simulation in this case is identical to any instance of $\mathsf{EXEC}_{\tilde{\pi},\mathcal{A},\mathcal{Z}}^{\mathcal{F}_{\mathsf{AAC}}^{\nu'}}$ running with the same adversary and a mapping ν' that agrees with the partial mapping ν defined by the simulator. □

Given the above lemmas, the proof of Theorem 2 now follows.

Proof. The service provider will pick tier-1 servers according to the strategy described in Lemma 7. That is, the service provider will choose as tier-1 all the inspected servers that were found clean and a random β-fraction of the inspected servers that were found corrupt and then restored to a clean state. Then, the service provider runs the MPC scheme described in Figure 6. Similarly to the proof of Theorem 1 we observe the following:

1. The MPC protocol is secure as long as a majority of tier-1 servers are honest (Proposition 5);
2. given that the adversary learns the protocol pseudonym and tier-level of a party only when this party is corrupt, when restricted to $\gamma \leq (1-\varepsilon)(1-\alpha+\alpha\beta)/2$ corruptions, it has only a exponentially-small probability in $m = O(n)$ to corrupt a majority of tier-1 servers (Lemma 7); and
3. Lemma 8 shows that an adaptive adversary learns only negligible information about ν (for uncorrupt parties).

Therefore, the computation is secure against the above adaptive adversary, except with negligible probability in n.

For Lemma 8, observe that the parties are divided into three sets: the set of clean servers after step (1) of the Corruption/Inspection Game (denoted by $\widehat{G_1}$ in Lemma 7); the set of servers that were restored to a clean state ($\widehat{G_2}$); and the rest of the servers. Setting $r = 3$, it is easy to see that Lemma 8 applies to our case by denoting those sets as $\mathcal{P}_1, \mathcal{P}_2$ and $\mathcal{P}_3$, respectively, and setting the protocol pseudonyms $\mathcal{P}_1^*, \mathcal{P}_2^*$ and $\mathcal{P}_3^*$ such that the number of tier-1 servers in each set matches the strategy of the service provider (e.g, β-fraction of the servers in each of the first two sets are tier-1, and no tier-1 servers in the third set). □

References

[AF04] M. Abe and S. Fehr. Adaptively secure Feldman VSS and applications to universally-composable threshold cryptography. M. Franklin, ed., *Advances in Cryptology — CRYPTO 2004*, *Lecture Notes in Computer Science*, vol. 3152, pp. 317–334. Springer, Heidelberg, 2004.

[AKS83] M. Ajtai, J. Komlós, and E. Szemerédi. An $O(n \log n)$ sorting network. *Proceedings of the fifteenth annual ACM symposium on Theory of computing*, STOC '83, pp. 1–9. ACM, New York, NY, USA, 1983.

[BCD+09] P. Bogetoft, D. Christensen, I. Damgård, M. Geisler, T. Jakobsen, M. Krøigaard, J. Nielsen, J. Nielsen, K. Nielsen, J. Pagter, M. Schwartzbach, and T. Toft. Secure multiparty computation goes live. R. Dingledine and P. Golle, eds., *Financial Cryptography and Data Security*, *Lecture Notes in Computer Science*, vol. 5628, pp. 325–343. Springer, Heidelberg, 2009.

[BdB90] J. Bos and B. den Boer. Detection of disrupters in the DC protocol. J.-J. Quisquater and J. Vandewalle, eds., *Advances in Cryptology — EUROCRYPT '89*, *Lecture Notes in Computer Science*, vol. 434, pp. 320–327. Springer, Heidelberg, 1990.

[BDS+03] D. Balfanz, G. Durfee, N. Shankar, D. K. Smetters, J. Staddon, and H.-C. Wong. Secret handshakes from pairing-based key agreements. *IEEE Symposium on Security and Privacy*, pp. 180–196. IEEE Computer Society, 2003.

[Bea97] D. Beaver. Commodity-based cryptography (extended abstract). *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, STOC '97, pp. 446–455. ACM, New York, NY, USA, 1997.

[Can00] R. Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptology*, 13(1):143–202, 2000.

[Can05] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. *IACR Cryptology ePrint Archive*, p. 67, 2005.

[CFGN96] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multi-party computation. Tech. rep., Massachusetts Institute of Technology, Cambridge, MA, USA, 1996.

[Cha88] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75, 1988.

[CJT04] C. Castelluccia, S. Jarecki, and G. Tsudik. Secret handshakes from ca-oblivious encryption. P. J. Lee, ed., *ASIACRYPT*, *Lecture Notes in Computer Science*, vol. 3329, pp. 293–307. Springer, 2004.

[Cle86] R. Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). J. Hartmanis, ed., *STOC*, pp. 364–369. ACM, 1986.

[FGMO05] M. Fitzi, J. A. Garay, U. M. Maurer, and R. Ostrovsky. Minimal complete primitives for secure multi-party computation. *J. Cryptology*, 18(1):37–61, 2005.

[GJ04] P. Golle and A. Juels. Dining cryptographers revisited. C. Cachin and J. Camenisch, eds., *Advances in Cryptology — EUROCRYPT 2004*, *Lecture Notes in Computer Science*, vol. 3027, pp. 456–473. Springer Berlin, Heidelberg, 2004.

[GJKY13] J. Garay, D. Johnson, A. Kiayias, and M. Yung. Resource-based corruptions and the combinatorics of hidden diversity. *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS '13, pp. 415–428. ACM, New York, NY, USA, 2013.

[GKKZ11] J. A. Garay, J. Katz, R. Kumaresan, and H.-S. Zhou. Adaptively secure broadcast, revisited. C. Gavoille and P. Fraigniaud, eds., *PODC*, pp. 179–186. ACM, 2011.

[GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, STOC '87, pp. 218–229. ACM, New York, NY, USA, 1987.

[HS05] D. Hush and C. Scovel. Concentration of the hypergeometric distribution. *Statistics & Probability Letters*, 75(2):127 – 132, 2005.

[IKOS06] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Cryptography from anonymity. *Foundations of Computer Science, 2006. FOCS '06. 47th Annual IEEE Symposium on*, pp. 239–248. 2006.

[KMTZ13] J. Katz, U. Maurer, B. Tackmann, and V. Zikas. Universally composable synchronous computation. *TCC*, pp. 477–498. 2013.

[PW92] B. Pfitzmann and M. Waidner. Unconditionally untraceable and fault-tolerant broadcast and secret ballot election. Hildesheimer Informatik Berichte. issn. 0941–3014, 1992.

[PW96] B. Pfitzmann and M. Waidner. Information-theoretic pseudosignatures and byzantine agreement for $t \geq n/3$. IBM Research Report RZ 2882 (#90830), 1996.

A Additional Definitions and Building Blocks

Signature schemes. A public-key signature scheme consists of three PPT algorithms $(\mathsf{GenS}, \mathsf{Sig}, \mathsf{Ver})$ such that $(sk, pk) \leftarrow \mathsf{GenS}(1^\kappa)$ generates a key; $sig \leftarrow \mathsf{Sig}_{sk}(m)$ generates a signature for $m \in \mathcal{M}$ and $b \in \{0,1\} \leftarrow \mathsf{Ver}_{pk}(m, sig)$ verifies a signature. For (sk, pk) generated by GenS, it holds that $\mathsf{Ver}_{pk}(m, \mathsf{Sig}_{sk}(m)) = 1$.

We say that a signature scheme is *existentially unforgeable* if any PPT adversary has only negligible advantage (in κ) in winning the following game running with a challenger:

Setup: The challenger runs $(pk, sk) \leftarrow KeyS(1^\kappa)$. It gives the adversary the resulting public key pk and keeps the private key sk to itself.

Queries: The adversary issues signature queries $m_1, \ldots, m_q$. To each query m_i, the challenger computes $sig_i \leftarrow \mathsf{Sig}_{sk}(m_i)$ and sends sig_i back to the adversary. Note that m_i may depend on previous signatures (adaptive queries).

Challenge: The adversary outputs a pair (m, sig), where $m \neq m_i$ for any m_i queried during the previous step. The adversary wins if $\mathsf{Ver}_{pk}(m, sig) = 1$.

Verifiable secret sharing (VSS). A (n, t)-VSS scheme is a protocol between a dealer and n parties $P_1, \ldots, P_n$, which extends a standard secret sharing. It consists of a SHARING PHASE where the dealer initially holds a value σ and finally, each party holds a private share v_i; and a RECONSTRUCTION PHASE in which the parties reveal their shares (a dishonest party may reveal $v_i' \neq v_i$) and a value σ' is reconstructed out of the shares $\sigma' = \mathsf{REC}(v_1', \ldots, v_n')$. Assuming an adversary that corrupts up to t parties, the following holds.

PRIVACY: If the dealer is honest, then the adversary's view during the sharing phase reveals no information about σ. More formally, the adversary's view is identically distributed under all different values of σ.

CORRECTNESS: If the dealer is honest, then the reconstructed value equals to σ.

COMMITMENT: After the sharing phase, a unique value σ^* is determined which will be reconstructed in the reconstruction phase; i.e., $\sigma^* = \mathsf{REC}(v_1', \ldots, v_n')$ regardless of the views provided by the dishonest players.

B Realizing Anonymous Broadcast

First, one may realize ABC via standard adaptively secure multyparty computation techniques [CFGN96]. This construction shows how multiple parties can securely compute any given circuit, which in the case of $\mathcal{F}_{\mathsf{ABC}}$ is a lexicographical sorting of the inputs. An asymptotically optimal sorting circuit is given in [AKS83] using $O(n \log n)$ comparators with depth $O(\log n)$.

Assuming the size of each field element is $O(\kappa)$ bits, a field-element comparator can be constructed out of binary gates in a tree fashion in size $O(\kappa)$ and depth $O(\log \kappa)$, or in a pipeline fashion with depth and size $O(\kappa)$. These constructions yield sorting circuits of size $O(\kappa n \log n)$ and depths $O(\log \kappa \log n)$ and $O(\kappa + \log n)$, respectively.

Note that the AAC protocol incurs only one call of ABC (i.e., there are no concurrent instances). Thus, invoking Canetti's modular composition theorem [Can00, Can05], such a construction gives adaptive security (with identifiable abort) against any number $t < n$ of corruptions. Observe that in the case of an abort, the only information that the adversary learns is the output, which is broadcast to all parties, and the security of the construction is not affected.

We refer to this protocol as $\mathsf{ABC}_{\mathsf{CFGN}}$; the next corollary immediately follows from [CFGN96].

Corollary 9. *Protocol* $\mathsf{ABC}_{\mathsf{CFGN}}$ *securely realizes* $\mathcal{F}_{\mathsf{ABC}}$ *requiring* $O(\min\{\log \kappa \log n, \kappa + \log n\})$ *rounds and total communication* $O(\kappa^2 n \log n)$*, assuming non-committing encryption is used to implement point-to-point secure communication between parties.*

Although the above realization of $\mathcal{F}_{\mathsf{ABC}}$ is sufficient for our purposes, we now discuss other alternatives, hoping for higher efficiency. First, we note that Golle and Juels [GJ04] present a scheme for honest-majority anonymous broadcast which uses bilinear maps, assuming the hardness of the Decisional Bilinear Diffie-Hellman problem (DBDH). Besides the honest-majority requirement, the construction does not consider "collisions," a common problem which arises in DC-nets in the selection of message positions; while the first shortcoming could be addressed by a player-elimination technique, addressing the second seems problematic, short of an MPC-type approach.

In [PW92, PW96], Pfitzmann and Waidner give an information theoretically secure sender-anonymous broadcast, based on Chaum's *DC-nets* [Cha88]. Their scheme assumes a *pre-computation* step during which a reliable broadcast is guaranteed. In our setting, we can replace the pre-computation reliable broadcast demand with an adaptively secure broadcast scheme, assuming a PKI setup [GKKZ11].

At a high level, the Pfitzmann-Waidner protocol consists of performing a many-to-many, corruption-detectable variant of a DC-net [BdB90], in which each user begins with a private input x_i and, if all parties behave as expected, ends with the multiset of inputs $\{x_i\}_i$ without being able to relate an input to its source. If some party deviates from the protocol, the other users notice this event (with high probability) and begin an 'investigation' in which each party should publicly reveal its messages and secret state, along with its private input. The parties can now check for consistency and (locally) identify the cheaters.

The resulting scheme, in the $\mathcal{F}_{\mathsf{PKI}}$-hybrid world, however is less efficient than the generic construction requiring $O(n^4)$ rounds with $O(\kappa n^2)$ communication per round.

C The Hypergeometric Distribution

We recall the Hypergeometric distribution and some of its properties. The Hypergeometric distribution with parameters n, m, t describes the probability to draw k 'good' items out of an urn that contains n items out of which m are good, when one is allowed to draw t items overall. The probability is given by

$$\mathsf{HypGeo}_{n,m,t}(k) = \binom{m}{k}\binom{n-m}{t-k}\Big/\binom{n}{t}.$$

The expectation of a random variable $K \sim \mathsf{HypGeo}_{n,m,t}$ is given by $\mathbb{E}[K] = t\frac{m}{n}$.

In our setting and terminology, $\mathsf{HypGeo}_{n,m,t}(k)$ describes the probability of corrupting k tier-1 servers, if there are n servers out of which m are tier-1, and the adversary is allowed to corrupt up to t servers altogether (assuming that the adversary learns the tier level of a specific server only when it gets corrupt).

A useful tool is a tail bound on the Hypergeometric distribution, derived by Hush and Scovel [HS05]:

Lemma 10. *Let $K \sim \mathsf{HypGeo}_{n,m,t}$ be a random variable distributed according to the Hypergeometric distribution with parameters n, m, t. Then,*

$$\Pr[K - \mathbb{E}[K] > \delta] < e^{-2\alpha_{n,m,t}(\delta^2-1)}$$

where

$$\alpha_{n,m,t} = \max\left(\left(\tfrac{1}{t+1} + \tfrac{1}{n-t+1}\right), \left(\tfrac{1}{m+1} + \tfrac{1}{n-m+1}\right)\right)$$

and assuming $\delta > 2$.