Fully-Secure MPC with Minimal Trust

Yuval Ishai
Technion

Arpita Patra
Indian Institute of Science

Sikhar Patranabis*
IBM Research India

Divya Ravi †
Aarhus University

Akshayaram Srinivasan
Tata Institute of Fundamental Research

Abstract

The task of achieving *full security* (with guaranteed output delivery) in secure multiparty computation (MPC) is a long-studied problem. Known impossibility results (Cleve, STOC 86) rule out general solutions in the dishonest majority setting. In this work, we consider solutions that use an *external trusted party* (TP) to bypass the impossibility results, and study the *minimal* requirements needed from this trusted party. In particular, we restrict ourselves to the extreme setting where the size of the TP is *independent* of the size of the functionality to be computed (called "small" TP) and this TP is invoked *only once* during the protocol execution. We present several positive and negative results for fully-secure MPC in this setting.

- For a natural class of protocols, specifically, those with a *universal output decoder*, we show that the size of the TP must necessarily be exponential in the number of parties. This result holds irrespective of the computational assumptions used in the protocol. The class of protocols to which our lower bound applies is broad enough to capture prior results in the area, implying that the prior techniques necessitate the use of an exponential-sized TP. We additionally rule out the possibility of achieving information-theoretic full security (without the restriction of using a universal output decoder) using a "small" TP in the plain model (i.e., without any setup).
- In order to get around the above negative result, we consider protocols without a universal output decoder. The main positive result in our work is a construction of such a fully-secure MPC protocol assuming the existence of a succinct Functional Encryption scheme. We also give evidence that such an assumption is likely to be necessary for fully-secure MPC in certain restricted settings.
- Finally, we explore the possibility of achieving full-security with a semi-honest TP that could collude with other malicious parties (which form a dishonest majority). In this setting, we show that even fairness is impossible to achieve regardless of the "small TP" requirement.

*Most of the work was done while the author was affiliated with ETH Zürich and Visa Research.

†Funded by the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 803096 (SPEC).

1 Introduction

Secure Multiparty Computation (MPC) allows a set of mutually distrusting parties to compute a joint function of their private inputs such that only the output of the function is revealed. Security of MPC protocols is required to hold even if the participating parties are controlled by a centralized *malicious* adversary, who may instruct them to deviate from the protocol specification.

Two desired properties for MPC protocols are *fairness* and *full security* (a.k.a guaranteed output delivery). Fairness requires that if the adversary learns the output of the functionality, then all the honest parties also learn this output. Full security strengthens fairness by requiring that the adversary cannot prevent the honest parties from learning the output of the functionality. Unfortunately, a classical impossibility result of Cleve [Cle86] shows that many functions cannot be fairly computed in the presence of an adversary corrupting a majority of the parties. Two ways to bypass this impossibility result are to restrict the adversary to corrupt only a minority of the parties, or to make use of some external help. In this work, we focus on the second approach, referring to the external help as a *trusted party* (TP).[1] A trusted party can be realized via different standard mechanisms, such as trusted execution environments, hardware tokens, blockchain based approaches, or cloud service providers.

Size of the TP. TPs are useful in circumventing the above impossibility result as they can be used as an ideal functionality that takes inputs from the parties and provides them outputs. A simple way to obtain protocols that satisfy full security in the TP model is for the TP to perform the entire computation on the private inputs of the parties and provide them outputs. However, this approach is less desirable as the size of the TP grows with the size of the function to be computed. Fitzi et al. [FGMO01] showed how to make the TP in the above solution *universal*, in the sense that it is independent of the function being computed. They also showed that to achieve full security, it is necessary to use TPs that take inputs from all the parties. However, this negative result does not rule out a TP which is independent of circuit size of the functionality. Thus, an interesting line of inquiry is to construct protocols where the size of the TP is independent of the circuit size of the functionality to be computed.

Apart from being a theoretically interesting question, it is also motivated by the practical goal of minimizing the use of trustworthy resources. For instance, if a trusted party service is implemented by a cloud service provider who charges fees for the use of its computational resources, it is obviously desirable (for the clients) to minimize the fees. The same holds if the TP is emulated via the use of a large-scale honest-majority MPC protocol. We refer to a setting of a trusted party whose size is independent of the circuit size of the function as the *small-TP* model. This problem is not new to our work and has already been considered in the works of Gordon et al. [GIM$^+$10] and Ishai et al. [IOS12] for the case of fairness and full security respectively. The state of the art result from [IOS12] gave a protocol that achieves guaranteed output delivery with statistical security (in the OT-hybrid model) with a small

[1]This notion differs from the line of work on token-based cryptography initiated by Katz [Kat07], where the tamper-proof tokens are generated locally, and the main challenge is to guarantee security even when tokens can be maliciously generated.

TP, where the parties make n sequential calls to this TP. In the same work, the authors gave a protocol where the parties make a single call to the TP but where the size of the TP grows exponentially in the number of parties (and is otherwise independent of the size of the function to be computed).

Number of Calls to TP. In this work, in addition to considering a small-TP model, we are interested in designing *fully-secure* protocols that make a single call to the TP. Theoretically, one call is the minimal requirement to circumvent the impossibility of [Cle86] for fair and fully-secure MPC. It further opens up the possibility of protocols in a minimal model, reminiscent of private simultaneous message (PSM) [FKN94] model, where given a common randomness, the parties communicate one-shot message to the TP and compute the output on receiving the reply from the TP. One call as opposed to many calls is also likely to generate more practical solution in the real world settings where, for instance, the TP is replaced with a cloud service provider, or a blockchain based approach.

The question which is the main focus to our work is:

Can we construct efficient protocols that make a single call to a "small" TP and achieve full security?

1.1 Our Results

We obtain both positive and negative results on the existence of fully-secure MPC protocols using a small TP. We start by discussing the negative results below.

Impossibility with a Universal Output Decoder. We give evidence that the prior approaches to this problem necessarily require a TP whose size is exponential in the number of parties. To show this, we abstract out the key features of prior protocols and show that any protocol having these features requires an exponential-sized TP (irrespective of the computational assumptions used in the protocol). More concretely, we consider the class of protocols where the parties could interact with each other (in an arbitrary number of rounds), then they make a single call to the trusted party, get a reply from TP, and then apply a *universal decoder* on this reply and their state to compute the output. By universal decoder, we mean that the size of the decoder is independent of the size of the functionality to be computed (considering single bit output functionalities). This model is interesting because it is quite natural and, more importantly, it captures prior approaches of realizing TP-aided MPC protocols [IOS12]. We show that for such protocols, the size of the TP necessarily grows exponentially with the number of parties. Our result holds irrespective of the computational assumptions used by the protocol. Additionally, our result holds even if the size of the TP is allowed to grow with the size of the function output.

Theorem 1 (Informal) *For any fully-secure MPC protocol with a universal output decoder, the size of the TP must necessarily be exponential in the number of parties.*

Necessity of Setup or Computational Assumptions. The above result naturally leads to the question of whether we can have small TP-aided fully secure MPC protocols once the restriction of using a universal decoder is relaxed. In this regard, we prove that any statistically secure protocol (without any trusted setup or correlated randomness) that makes a single call to a small TP cannot be even *semi-honest* secure. This impossibility holds even against protocols that may not have a universal output decoder. This shows that to achieve full security it is necessary to resort to computational assumptions, or assume some sort of setup (such a correlated randomness).

Theorem 2 (Informal) *There exists no MPC protocol that achieves information-theoretic security against semi-honest adversaries in the plain model with a TP whose size is a fixed polynomial in the input size of the functionality to be computed.*

Positive Results. We now focus on the problem of achieving fully-secure MPC protocols using a small TP based on computational assumptions. Our main positive result is captured by the following theorem:

Theorem 3 (Informal) *Assuming a single-key succinct Functional Encryption (FE) scheme, there exists a fully secure efficient MPC protocol that makes a single call to the small TP.*

A single-key succinct Functional Encryption is an FE scheme [SW05, O'N10, BSW11] where the size of the encryption algorithm does not grow with size of the function for which a secret key is released. Using known instantiations of these primitives from various assumptions, we get the following corollary (building on [GKP+13, GGSW13, Wat15]).

Corollary 4 (Informal) *There exists a fully secure efficient MPC protocol that makes a single call to a TP, assuming:*

1. *Learning with Errors (with sub-exponential modulus-to-noise ratio) [GKP+13] if the size of the TP is allowed to only grow with the depth and the output length of the functionality.*
2. *Witness Encryption scheme [GGSW13] and FHE if the size of the TP is allowed to only grow with the output length of the functionality.*
3. *Indistinguishability Obfuscation (iO) [BGI+01, GGH+13, JLS21] and one-way functions, where the size of the TP is independent of the depth and the output length.*

We also give evidence that this assumption might be necessary in certain restricted settings. Specifically, consider a restricted model of computation where the parties do not interact with each other, but make a single-call to the TP and could compute the output of the functionality based on the reply from the TP. This model is reminiscent of the Private Simultaneous Messages setting [FKN94]. It is not too hard to see that this restricted model is equivalent to an MPC protocol with a succinct online phase. Specifically, the computation done by the parties before the TP call can be thought of as the pre-processing phase and this could grow with the circuit-size of the functionality. The messages sent to the TP and the computation performed by the TP correspond to the online phase of the protocol. Since we restrict the size of the TP to be small, it follows that the computation and the communication cost of the online phase is independent of the size of the functionality (i.e., the protocol has succinct online phase). The post-processing phase could grow with the size of the functionality to be computed (this is in fact necessary considering our impossibility with a universal output decoder).

Currently, the only known constructions of an MPC protocol with a succinct online phase are based on Laconic Functional Evaluation [QWW18] (LFE). This is known to imply a succinct Functional Encryption. This suggests that such assumptions are *likely* to be necessary in the restricted setting outlined above. In fact, an MPC protocol with a succinct online phase implies a weaker flavor of LFE with the following property: unlike standard LFE where the size of the encryption algorithm only grows with the input size, the encryption algorithm in this weaker notion of LFE has two components: (i) a pre-processing algorithm which takes the input and the size of the functionality and produces a hint that only grows with the input size, and (ii) a second algorithm that takes the input and the hint and outputs a ciphertext (the size of the second algorithm only grows with the input size). Finally, in this restricted model, we give a positive result by constructing a fully-secure MPC protocol with a single call to a small TP based on LFE.

(Im)Possibility of Reducing the Trust in TP. Finally, we explore the possibility of weakening the security requirements from the TP. Interestingly, our above solutions maintain privacy against the TP, which is an additional desirable feature. More specifically, our constructions are secure if the adversary corrupts the TP in a semi-honest manner (but does not corrupt any of the parties). This led us to explore what happens if we allow the semi-honest TP to collude with the other malicious parties. We showed that irrespective of the size of the TP, such a model would not be enough to circumvent Cleve's impossibility of fairness. This impossibility holds even if we restrict the malicious parties to be fail-stop.[2]

The summary of our results appears in Table 1.

1.2 Open Directions

Our work opens up several interesting research directions. We highlight some of them below.

- **Showing Necessity of Succinct FE.** In this work, we argued that any protocol in the restricted model (where the parties do not communicate with each other before and after the TP invocation) is equivalent to an MPC protocol with a succinct online phase. However, we are unable to extend this to the setting where the parties could potentially communicate with each other before making the TP call. Can we show that such a weaker model also implies some weakening of an MPC protocol with a succinct online phase? This would justify the necessity of a succinct FE assumption.

[2]The notion of fail-stop corruption lies between semi-honest and malicious corruption, where eavesdropping like semi-honest corruption is allowed and the only possible malicious corruption is stopping the execution of the protocol.

Security	**No. of calls**	**Setup**	**Pre-TP call interaction**	**Universal Output Decoder**	**Possible?**	**Reference**
Statistical	1	Plain	Yes	No	No	Theorem 21
Computational	1	C.R.	Yes	Yes	No	Theorem 19
Computational	1	CRS	No	No	Yes (based on LFE)	Theorem 16
Computational	1	Plain	Yes	No	Yes (based on succint FE)	Theorem 17
Statistical	n	C.R	Yes	Yes	Yes	[IOS12]
Computational	n	Plain	Yes	Yes	Yes (based on OT)	[IOS12]
Statistical	1	C.R	Yes	No	**Open**	

Table 1: Results on fully-secure MPC in dishonest majority using *small* TP under different kinds of setup (plain model i.e. no setup / C.R. i.e. correlated randomness setup / CRS i.e. common random string), security guarantees (statistical / computational) and different TP computation models (with / without the restrictions on pre-TP call interaction and universal output decoder).

- **Making more than a Single Call to TP.** As our goal was to minimize the requirements from the TP as much as possible, we considered the extreme setting where a single call is made to the TP. A fascinating direction is to explore the possibility of constructing fully-secure MPC protocols from weaker assumptions which could make more than one call but less than n calls. The key challenge here is to design protocols using a stateless TP. If we allow the TP to be stateful, we can realize a construction based on FHE that makes two calls to a stateful TP (we sketch this construction in Appendix E).
- **Characterization of Fair Computation in the Colluding TP model.** As mentioned previously, in this work we show that it is impossible to achieve fairness in the colluding TP model (where the adversary can corrupt the TP in a semi-honest manner, in addition to corrupting majority of the parties maliciously) for general functions. However, it is still possible to achieve fairness for restricted classes of (non-trivial) functions such as coin-tossing (by using the TP to directly compute the desired function). It is an interesting open question to give a complete characterization of which function classes can be fairly computed in the colluding TP model.

1.3 Technical Highlights and Discussion

In this section, we present a high-level technical overview of our results.

1.3.1 Positive Results

We present two protocols based on LFE [QWW18] and single-key succinct FE [SW05, BSW11] respectively utilizing a *single* call to a stateless "small" TP. We start off with their trade-offs below.

LFE-based Construction. LFE's 2-round minimal communication pattern leads to an MPC in a minimal communication setting that is reminiscent of PSM-style [FKN94] communication. Here, the parties start off with a common randomness. Based on the respective inputs and this randomness, the parties communicate a single message to the TP, which performs certain computation and returns a message to each party. In the end, each party recovers the output receiving the message from the TP. Further, the encryption algorithm of LFE enjoys computation that is only dependent on the depth and the output length (and not size) of the function to be computed. This allows our TP to be "small". Here with the best known realizations of LFE, we can achieve a TP of size $\texttt{poly}(n, \kappa, d, m)$, where d denotes depth of the circuit and m denotes input and output size of the circuit, n denotes the number of parties and κ denotes the security parameter. Removing m from the complexity of the TP seems hard, intuitively because the parties never communicate with each other and they communicate only once via the TP. Achieving depth and input-size independence in this minimal communication setting is left as an interesting open question which can possibly contribute back to the LFE regime. In particular, a solution in our setting where TP is of size $\texttt{poly}(n, \kappa, m)$ will lead to a LFE where the encryption scheme and size of the ciphertext are completely independent of the depth of the function under consideration.

FE-based Construction. Unlike the LFE-based construction, our FE-based construction requires communication amongst the parties before making the TP call. While it loses on this front, there are two positive features that it brings to the table: (a) possibly weaker assumption (b) the TP's computation can be independent of d, m. Elaborating further, LFE is seemingly a stronger assumption than FE, since it is known to imply FE, while the other way is not known [QWW18]. Based on the realization of FE under various assumptions, we achieve multiple variants of the protocol where the TP's computation ranges from being completely independent of input, output and function to linearly dependent on output size (yet independent of the function) to linearly dependent on the output size and the depth of the function. To be specific, under iO and OWFs, our FE based construction leads to a TP of size $\texttt{poly}(n, \kappa)$, completely independent of the function to be computed.

Construction Overview. Our constructions follow a three-phase structure as follows: (a) phase 1: here the parties, on holding a common randomness and respective inputs, prepare a (message, state) pair, where the message is sent to the TP and the state is saved; (b) phase 2: the TP, on receiving messages from the parties, performs some computation and returns a message to every party; and (c) phase 3: the parties, on receiving the message from the TP, uses its state to recover the output. Phase 1 involves communication amongst the parties in the FE-based construction.

We keep our TP small for both constructions by carefully assigning the tasks under LFE and FE to the parties and TP. In both the constructions, the TP performs the "encryption" part of the computation, which makes it circuit-size independent. Whereas the parties take care of the set-up part (digest generation for LFE and master public/secret key and function secret key for FE) and the decryption part, both of which are circuit-size dependent. We provide an informal overview behind the idea for each construction below.

Overview of LFE-based Solution. We present here a simplified version of our LFE-based construction of fully-secure MPC for ease of exposition. The actual construction, detailed in Section 3.3, is significantly more nuanced and uses several techniques to achieve full security against malicious corruptions of parties. In the simplified treatment presented here, we focus on the case of semi-honest corruption, with the aim of highlighting how we manage to keep the TP size small (i.e., independent of the function size). Note that throughout this paper, we assume that each party communicates with the TP via a separate secure channel, and hence an adversary (corrupting a subset of the parties) cannot eavesdrop on the communication between the TP and any honest party.

Given this model, a simplified version of our LFE-based protocol works as follows. Each party first uses a common randomness to (locally) derive a CRS for the LFE scheme and a digest corresponding to the function f. Each party then sends the LFE CRS and the function digest to the TP, along with its own input. The TP uses the CRS and the digest to compute an LFE ciphertext encapsulating the inputs of all of the parties, and sends this ciphertext back to the parties. Finally, each party uses the LFE CRS and its local randomness of digest generation to recover the function output. Observe that the size of the messages to the TP and the computation done by the TP are independent of the size of the function f; this follows immediately from the succinctness properties of the underlying LFE scheme. Finally, we can invoke the privacy guarantees of LFE to argue that the parties learn no more information than the output of the MPC protocol, as desired.

As mentioned earlier, our actual LFE-based protocol uses additional techniques to guarantee full security in the presence of malicious corruptions. This includes techniques that enable the TP to "partition" the parties into various sets depending on their messages to the TP, and to substitute default input values for (malicious) parties not in the partition when preparing partition-specific LFE ciphertexts. Further, we augment the construction to achieve privacy against the TP. We refer to Section 3.3 for the detailed description and analysis of our construction.

Overview of FE-based Solution. We now present a simplified version of our FE-based construction of fully-secure MPC. Once again, our actual protocol, detailed in Section 3.3 uses additional techniques to achieve full security against malicious corruptions of parties; we avoid detailing all of these in the simplified treatment for ease of exposition and focus on the setting of semi-honest corruptions. As in the LFE-base solution, we again assume that each party communicates with the TP via a separate secure channel, and hence an adversary (corrupting a subset of the parties) cannot eavesdrop on the communication between the TP and any honest party.

Given this model, the simplified version of our FE-based protocol works as follows. The parties initially engage in an MPC protocol (with identifiable abort security) to decide on a common set of public parameters and a common master public key for the FE scheme. The MPC protocol additionally outputs to each party a functional secret key for the function f to be evaluated. Each party then simply sends the master public key and its own input to the TP. The TP uses the master public key to compute an FE ciphertext encapsulating the inputs of all of the parties, and sends this ciphertext back to the parties. Finally, each party uses the functional secret key to recover the function output. Observe that the size of the messages to the TP and the computation done by the TP are independent of the size of the function f as long as the FE scheme is succinct. Finally, we can invoke the privacy guarantees of FE to argue that the parties learn no more information than the output of the MPC protocol, as desired.

Note that in the above simplified exposition, the TP incurs an overhead that grows with the size of the inputs and output of the function f to be evaluated. In our actual protocol, we use additional techniques to get rid of this dependence. In particular, we use a carefully designed indirection mechanism that allows the TP to simply partition the set of parties (depending on their messages to the TP) and encapsulate this partition information into the FE ciphertext, while delegating all computation dependent on the input/function size entirely to the parties. These techniques serve two purposes: (a) making the TP size independent of the function input/output size (and thereby asymptotically smaller than the TP size for our LFE-based solution) and (b) achieving full security against malicious corruptions of parties. Interestingly, this solution also achieves privacy against the TP. We refer to Section 3.4 for the detailed description and analysis of our construction.

1.3.2 Negative Results

We present two impossibility results for fully-secure MPC that utilizes a small TP. Our two results are as follows: **(1)** First, we show that it is impossible to achieve a fully secure TP-aided MPC utilizing a single call to a small TP, for a class of protocols that have an universal output decoder. This result holds irrespective of computational assumptions used in the protocol. The universal output decoder is independent of the function to be computed and only performs $\texttt{poly}(n, \kappa)$ computation. **(2)** Second, we show an impossibility in the plain model, for any statistically-secure MPC even in the semi-honest setting. This result does not assume that the protocol uses an universal output decoder. We present the high-level intuition of both the impossibility arguments.

Impossibility of Fully-Secure MPC protocols with universal output decoder in the Correlated Randomness Model. We now present a simplified argument of our impossibility result and refer to Section 4.1 for the details. Consider an execution of an MPC protocol with full security, where the adversary behaves honestly until the TP call. During the TP call, he can choose to make any subset of corrupt parties, say S, abort; where the number of such subsets is exponential in the number of parties. Since the protocol achieves full security, it must be the case that the TP is able to enable output computation by the parties, no matter which subset S the adversary chooses. Further, the output must be such that it is computed on the default input of the corrupt parties in S and the honest inputs of others (i.e. the input used until and including the TP call). Intuitively, this means that the information given to the TP is such that it can be used to recover 2^n output values (one for each possible subset). Since the TP is small, this information must be 'short' and can therefore be perceived as a 'compression' of the 2^n output values. Building on the above intuition, we show that a fully secure protocol with universal output decoder would imply an (encoding, decoding) scheme which can produce an encoding that is smaller than the size of the message domain of the encoding scheme. This breaches the known incompressibility argument. Precisely, we use a result of De et al. [DTT10], which formalizes the notion that it is impossible to compress every element in a set X to a string less than $\log |X|$ bits long.

Impossibility of Statistical MPC in the Plain Model. At a high-level, we show this impossibility by demonstrating that such a protocol would imply a semi-honest information-theoretic oblivious transfer (OT) extension, which is known to be impossible [Bea96]. Here, OT extension refers to a protocol that allows a sender and a receiver to extend a relatively small number of base OTs (say k) to a larger number of OTs (say $k+1$) using only symmetric-key primitives.

The main idea of the proof is to construct an OT extension protocol using the semi-honest statistically-secure protocol, say Π, as follows. We choose the functionality computed by Π as computing $(k+1)$ oblivious transfer instances. Since the TP is small, its size must be strictly less than the circuit computing $(k+1)$ oblivious transfer instances. Roughly speaking, Π can thus be viewed as a protocol that enables the parties to generate $(k+1)$ OTs, by having access to the TP whose functionality can be realized by strictly less than $(k+1)$ OTs (say k OTs). We build on this idea to construct an information-theoretic semi-honest OT extension protocol where the parties begin with k base OTs and use Π to generate $(k+1)$ OTs.

1.3.3 Impossibility of Fair MPC with Colluding TP

Our results show that small TP is sufficient for positive results in the computational security regime. But what happens when the TP is no longer a stand-alone entity, but behaves as another party that can not only eavesdrop but also collude with the corrupt parties (while remaining semi-honest by itself)? This is a model where the adversary controls a majority of the parties maliciously (or even fail-stop fashion) and *simultaneously* corrupts the TP semi-honestly. For this model, we ask the questions: *Can such a TP circumvent Cleve's [Cle86] impossibility result?*

We show a negative result for the above question even for fail-stop adversaries (i.e., the malicious parties still follow the protocol specification but may choose to stop arbitrarily). At a high level, we take the following route. Note that the colluding adversarial model can be viewed more generally, in terms of the general mixed adversarial model that has been studied in works such as [HMZ08, FHM99, BFH+08]. We then use the characterization proposed in [HMZ08] for fair and fully-secure MPC tolerating mixed adversaries to rule out a fair protocol in the colluding model even when malicious corruption is replaced with fail-stop corruption. In particular, we define an adversarial structure complying with the colluding security model and show that this structure is ruled out by the characterization provided in [HMZ08].

In light of this generic negative result, we also explore whether a TP can be used in the colluding model to realize fair MPC protocols for certain *specific* classes of non-trivial functions such as randomized functions without inputs (e.g. coin-tossing). A naïve solution uses the TP to directly compute the desired function; however, such a TP can no longer be small. We give evidence that a better solution using a small TP is unlikely to exist.

1.4 Related Work

There are several fascinating works in the MPC literature that attempt to bypass fundamental feasibility results using external aid. Impossibility of fair MPC in dishonest majority [Cle86] is one such classic impossibility result that has received noteworthy attention. We focus on three broad categories of related works. First is the most closely related line of work to ours which studies the 'minimal help' required to compute all functions fairly, where the helper is characterized as a 'complete' primitive. Second, we outline the line of works that circumvent the impossibility of [Cle86] by considering non-standard notions of fairness. Lastly, we outline the works that circumvent yet another classical impossibility, namely, impossibility of secure computation of general functionalities within the universal composability (UC) framework in presence of dishonest majority in the plain model [CF01] by using hardware tokens and physically unclonable functions (PUFs).

The work of [FGMO01] initiated the study of minimal complete primitives for secure computation, focusing on the minimal cardinality of complete primitives for various thresholds. In particular, they showed that cardinality n is necessary for any complete primitive in dishonest majority and proposed Universal Black Box (UBB) as one such primitive. Subsequently, the work of [GIM$^+$10] proposed a simpler complete primitive for fairness in dishonest majority, namely 'fair reconstruction'. While [GIM$^+$10] focused on the computational setting, [IOS12] presented the first unconditional construction of a complete primitive for full security, whose complexity does not grow with the complexity of the function being evaluated (in contrast to the UBB solution of [FGMO01]). However, this unconditional construction of [IOS12] utilizes number of calls that scales with the circuit size. To improve the number of calls, [IOS12] also proposes another construction where the number of calls depends only on the number of parties (n) and the output size of the circuit but settles for computational security in the plain model. Finally, they also have a variant where the number of calls is reduced to 1 at the price of increasing the complexity of the computation done by the complete primitive exponentially in n.

As mentioned earlier, an interesting feature that our constructions satisfy is to maintain privacy against the TP. We note that the unconditional variant of [IOS12] (that utilizes number of calls scaling with circuit size) leaks the inputs of the parties to the TP. With respect to the computational variants in [IOS12] that only leak the output of the computation to the TP, we note that it can be tweaked to maintain privacy of the output by adopting the technique of [GIM$^+$10].

Other works related to breaking barriers imposed by the impossibility of [Cle86] include the works of [GK09, GHKL11, ABMO15] that achieve fairness in dishonest majority for restricted functionalities. Some other works explore non-standard notions of fairness such as [GK12, BOO15, BLOO20] that considers partial fairness, [BK14, KB14, ADMM14] that enforce fairness by imposing penalties, [CGJ$^+$17] that use bulletin boards and [EGL85, GMPY11, PST17] that explore resource-fairness.

The sequence of works of [Kat07, CKS$^+$14, DMRV13, CGS08, CCOV19, HPV16] study UC-security with tamper-proof hardware token, both in the stateful and stateless variants. Another interesting utility of hardware tokens is reflected in designing Non-Interactive Secure Computation (NISC) protocols using minimal assumptions. The work of [BJOV18] proposes a UC-secure NISC protocol based on the minimal assumption of one-way functions using hardware token. Lastly, the works of [BFSK11,OSVW13,BKOV17] explore UC-secure computation assuming access to PUFs.

Paper Outline. We formally define TP-aided MPC protocols in Section 2. Our positive results appear in Section 3. The corresponding security proofs are deferred to Appendices A and B. Our negative results for TP-aided MPC appear in Section 4, with some extensions presented in Appendix C. Our negative results in the colluding TP model are given in Section 5 and Appendix D. Finally, Appendix E outlines some results on TP-aided MPC in the stateful TP model.

2 Security Model

In this section, we present our definitions in the UC-framework [Can01]. We denote by $[p]$ the set $\{1, \dots, p\}$, for a positive integer p.

The Real World. An n-party protocol Π with n parties $\mathcal{P} = (P_1, \dots, P_n)$ is an n-tuple of probabilistic polynomial-time (PPT) interactive Turing machines (ITMs), where each party P_i is initialized with input $x_i \in \{0,1\}^*$ and random coins $r_i \in \{0,1\}^*$. These parties interact in synchronous rounds. In every round parties can communicate either over a broadcast channel or a fully connected point-to-point (P2P) network, where we additionally assume all communication to be private and ideally authenticated. Further, we assume that there exists a special party P^* called a "trusted party" (abbreviated henceforth as TP) such that each party P_i can interact with P^* via private and authenticated point-to-point channels. The TP P^* does not typically hold any inputs, and also does not obtain any output at the end of the protocol. Further, the TP is *stateless* in the sense that it does not keep any state between calls.

We let $\mathcal{A}$ denote a special ITM that represents the adversary. $\mathcal{A}$ is coordinated by another special non-uniform ITM environment $\mathcal{Z} = \mathcal{Z}_\kappa$. At setup, $\mathcal{Z}$ gives input $(1^\kappa, x_i)$ to each party P_i. At the same time, $\mathcal{Z}$ provides to $\mathcal{A}$ the tuple $(\mathcal{C}, \{x_i\}_{i \in \mathcal{C}}, \mathsf{aux})$, where $\mathcal{C} \subset [n] \cup \{P^*\}$ denotes the set of all corrupt parties, and aux denotes some auxiliary input.

During the execution of the protocol, the maliciously corrupt parties (sometimes referred to as 'active') receive arbitrary instructions from the adversary $\mathcal{A}$, while the honest parties and the semi-honestly corrupt (sometimes referred to as 'passive') parties faithfully follow the instructions of the protocol. We consider the adversary $\mathcal{A}$ to be rushing, i.e., during every round the adversary can see the messages the honest parties sent before producing messages from corrupt parties.

At the conclusion of the protocol, $\mathcal{A}$ gives to the environment $\mathcal{Z}$ an output which is an arbitrary function of $\mathcal{A}$'s view throughout the protocol. $\mathcal{Z}$ is additionally given the outputs of the honest parties. Finally, $\mathcal{Z}$ outputs a bit. We let $\mathsf{real}_{\pi,\mathcal{A},\mathcal{Z}}(\kappa)$ be a random variable denoting the value of this bit.

Definition 5 (Real-world execution) *Let Π be an n-party protocol amongst $(P_1, \ldots, P_n)$ computing an n-party function $f : (\{0,1\}^*)^n \to (\{0,1\}^*)^n$ and let $\mathcal{C} \subseteq [n] \cup \{P^*\}$ denote the set of indices of the corrupted parties. The execution of Π under $(\mathcal{Z}, \mathcal{S}, \mathcal{C})$ in the real world, on input vector $\vec{x} = (x_1, \ldots, x_n)$, auxiliary input* aux *and security parameter κ, denoted* $\mathsf{real}_{\Pi,\mathcal{C},\mathcal{A}(\mathsf{aux})}(\vec{x}, \kappa)$*, is defined as the output of $\mathcal{Z}$ resulting from the protocol interaction.*

The Ideal World. We describe ideal world executions with unanimous abort ($\mathsf{un\text{-}abort}$), identifiable abort ($\mathsf{id\text{-}abort}$), fairness ($\mathsf{fairness}$) and full security aka. guaranteed output delivery (full).

Definition 6 (Ideal Computation) *Consider* $\mathsf{type} \in \{\mathsf{un\text{-}abort}, \mathsf{id\text{-}abort}, \mathsf{fairness}, \mathsf{full}\}$*. Let $f : (\{0,1\}^*)^n \to (\{0,1\}^*)^n$ be an n-party function. Once again, we have a non-uniform environment $\mathcal{Z} = \mathcal{Z}_\kappa$ that gives (at setup) input $(1^\kappa, x_i)$ to each party P_i, while also providing to the simulator $\mathcal{S}$ the tuple $(\mathcal{C}, \{x_i\}_{i \in \mathcal{C}}, \mathsf{aux})$, where $\mathcal{C} \subset [n] \cup \{P^*\}$ denotes the set of all corrupt parties, and* aux *denote some auxiliary input. Then, the ideal execution of f under $(\mathcal{Z}, \mathcal{S}, \mathcal{C})$ on input vector $\vec{x} = (x_1, \ldots, x_n)$, auxiliary input* aux *to $\mathcal{S}$ and security parameter κ, denoted* $\mathsf{ideal}^{\mathsf{type}}_{f,\mathcal{C},\mathcal{S},(\mathsf{aux})}(\vec{x}, \kappa)$*, is defined as the output bit of $\mathcal{Z}$ resulting from the following ideal process.*

1. Parties send inputs to trusted party*: An honest party P_i sends its input x_i to the trusted party. The simulator $\mathcal{S}$ may send to the trusted party arbitrary inputs for the corrupt parties. Let x'_i be the value actually sent as the input of party x_i.*
2. Trusted party speaks to simulator*: The trusted party computes $(y_1, \ldots, y_n) = f(x'_1, \ldots, x'_n)$. If there are no corrupt parties or* $\mathsf{type} = \mathsf{full}$*, proceed to step 4.*
   (a) If $\mathsf{type} \in \{\mathsf{un\text{-}abort}, \mathsf{id\text{-}abort}\}$: *The trusted party sends $\{y_i\}_{i \in \mathcal{C}}$ to $\mathcal{S}$.*
   (b) If $\mathsf{type} = \mathsf{fairness}$: *The trusted party sends* `ready` *to $\mathcal{S}$.*
3. Simulator $\mathcal{S}$ responds to trusted party*:*
   (a) If $\mathsf{type} \in \{\mathsf{un\text{-}abort}, \mathsf{fairness}\}$: *The simulator can send* `abort` *to the trusted party.*
   (b) If $\mathsf{type} = \mathsf{id\text{-}abort}$: *If it chooses to abort, the simulator $\mathcal{S}$ can select a corrupt party $i^* \in \mathcal{C}$ who will be blamed, and send* $(\texttt{abort}, i^*)$ *to the trusted party.*
4. Trusted party answers parties*:*
   (a) *If the trusted party got* `abort` *from the simulator $\mathcal{S}$,*
      i. *It sets the abort message* $\mathsf{abortmsg}$*, as follows:*
         - *if* $\mathsf{type} \in \{\mathsf{un\text{-}abort}, \mathsf{fairness}\}$*, we let* $\mathsf{abortmsg} = \perp$.
         - *if* $\mathsf{type} = \mathsf{id\text{-}abort}$*, we let* $\mathsf{abortmsg} = (\perp, i^*)$.
      ii. *The trusted party then sends* $\mathsf{abortmsg}$ *to every party P_j, $j \in [n] \setminus \mathcal{C}$.*

      Note that, if $\mathsf{type} = \mathsf{full}$*, we will never be in this setting, since $\mathcal{S}$ was not allowed to ask for an abort.*
   (b) *Otherwise, it sends y_j to every P_j, $j \in [n]$.*

5. Outputs*: Honest parties always output the message received from the trusted party while the corrupt parties output nothing. At the conclusion of the above execution, $\mathcal{S}$ provides $\mathcal{Z}$ with an output which is an arbitrary function of $\mathcal{S}$'s view throughout the protocol. $\mathcal{Z}$ is additionally given the outputs of the honest parties. Finally, $\mathcal{Z}$ outputs a bit. We let $\mathsf{ideal}^{\mathsf{type}}_{f,\mathcal{S},\mathcal{Z}}(\kappa)$ be a random variable denoting the value of this bit.*

Security Definitions. We now define the security notions used in this paper.

Definition 7 (Colluding and Non-colluding Security) *Consider* $\mathsf{type} \in \{\mathsf{un\text{-}abort}, \mathsf{id\text{-}abort}, \mathsf{fairness}, \mathsf{full}\}$*. Let $f : (\{0,1\}^*)^n \to (\{0,1\}^*)^n$ be an n-party function. A protocol Π securely computes the function f in the* colluding model *with* type *security if for any adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}$ such that for any security parameter κ and any circuit family $\mathcal{Z} = \{\mathcal{Z}_\kappa\}$ corrupting any $\mathcal{C} \subset [n]$ maliciously and the TP P^* semi-honestly simultaneously, we have*

$$\left\{\mathsf{real}_{\Pi,\mathcal{C},\mathcal{A}(\mathsf{aux})}(\vec{x},\kappa)\right\}_{\vec{x}\in(\{0,1\}^*)^n,\kappa\in\mathbb{N}} \equiv \left\{\mathsf{ideal}^{\mathsf{type}}_{f,\mathcal{C},\mathcal{S}(\mathsf{aux})}(\vec{x},\kappa)\right\}_{\vec{x}\in(\{0,1\}^*)^n,\kappa\in\mathbb{N}}.$$

When the corruption is non-simultaneous i.e. either *any subset of $[n]$ are maliciously corrupt* or *the TP P^* is semi-honestly corrupt, we denote the security by* non-colluding*. Therefore we need the above indistinguishability to hold in two corruption cases: (a) $\mathcal{C} \subset [n]$ malicious corruption (b) $\mathcal{C} = P^*$ semi-honest corruption.*

A protocol achieves computational security, if the above distributions are computationally close in the presence of the parties, $\mathcal{A}$, $\mathcal{S}$, $\mathcal{Z}$ that are PPT. A protocol achieves statistical (resp. perfect) security if the distributions are statistically close (resp. identical).

3 Fully-secure MPC with Single Call to Small TP

Here, we present TP-aided MPC protocols that make a single call to a small TP and achieve full security in the non-colluding setting against malicious corruption of majority of parties and semi-honest corruption of the TP. We present two flavors of protocols– one based on laconic function evaluation (LFE) [QWW18] and the other based on succinct single-key functional encryption (FE) [GKP+13]. We begin by recalling the definitions for these primitives.

3.1 Laconic Function Evaluation (LFE)

We recall the definition of LFE – a primitive introduced in [QWW18].

Definition 8 (Laconic Function Evaluation) *An* LFE *scheme for a class of circuits $\mathcal{H} = \{\mathcal{H}_m\}_{m\in\mathbb{N}}$ (represented as Boolean circuits with m-bit inputs) is a tuple* ($\mathsf{LFE.Setup}$, $\mathsf{LFE.Compress}$, $\mathsf{LFE.Enc}$, $\mathsf{LFE.Dec}$) *defined below.*

- $\mathsf{LFE.Setup}(1^\kappa) \to \mathsf{LFE.crs}$*: On input the security parameter* 1^κ*, the generation algorithm returns a common random string* $\mathsf{LFE.crs}$*.*
- $\mathsf{LFE.Compress}(\mathsf{LFE.crs}, h) \to (\mathsf{digest}, r)$*: On input* $\mathsf{LFE.crs}$ *and a circuit* h*, the compression algorithm returns a digest* digest *and a decoding information* r*.*
- $\mathsf{LFE.Enc}(\mathsf{LFE.crs}, \mathsf{digest}, x) \to \mathsf{ct}$*: On input* $\mathsf{LFE.crs}$*, a digest* digest*, and a message* x*, the encryption algorithm returns a ciphertext* ct*.*
- $\mathsf{LFE.Dec}(\mathsf{LFE.crs}, \mathsf{ct}, r) \to y$*: On input* $\mathsf{LFE.crs}$*, a ciphertext* ct*, and a decoding string* r*, the decoding algorithms returns a message* y*.*

In this work, we use LFE schemes that satisfy correctness, simulation-security and function-hiding security, as defined formally below.

Definition 9 (Correctness) *Let* $\mathsf{LFE} = (\mathsf{LFE.Setup}, \mathsf{LFE.Compress}, \mathsf{LFE.Enc}, \mathsf{LFE.Dec})$ *be an LFE scheme for a class of functions* $\mathcal{H} = \{\mathcal{H}_m\}_{m\in\mathbb{N}}$*. We say that* LFE *is a correct LFE scheme if for any* $m = \mathtt{poly}(\kappa)$*, for all* $h \in \mathcal{H}_m$*, and for all* $x \in \{0,1\}^m$*, letting* $\mathsf{LFE.crs} \leftarrow \mathsf{LFE.Setup}(1^\kappa)$*, and letting*

$$(\mathsf{digest}, r) \leftarrow \mathsf{LFE.Compress}(\mathsf{LFE.crs}, h), \quad \mathsf{ct} \leftarrow \mathsf{LFE.Enc}(\mathsf{LFE.crs}, \mathsf{digest}, x),$$

the following holds:

$$\Pr[\mathsf{LFE.Dec}(\mathsf{LFE.crs}, \mathsf{ct}, r) = h(x)] = 1 - \mathtt{negl}(\kappa),$$

where the probability is taken over the random coins of $\mathsf{LFE.Setup}$*,* $\mathsf{LFE.Compress}$*, and* $\mathsf{LFE.Enc}$*.*

Definition 10 (Simulation-Security) *Let* $\mathsf{LFE} = (\mathsf{LFE.Setup}, \mathsf{LFE.Compress}, \mathsf{LFE.Enc}, \mathsf{LFE.Dec})$ *be an LFE scheme for a class of functions* $\mathcal{H} = \{\mathcal{H}_m\}_{m\in\mathbb{N}}$*. For every non-uniform PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *and every PPT simulator* $\mathcal{S}$*, consider the following two experiments (*κ *being the security parameter):*

Experiment $\mathsf{Expt}^{\mathsf{real}}_{\mathsf{LFE},\mathcal{A}}(1^\kappa)$*:*

```
$\mathsf{LFE.crs} \leftarrow \mathsf{LFE.Setup}(1^\kappa)$
$(x, h, s, \mathtt{st}_\mathcal{A}) \leftarrow \mathcal{A}_1(1^\kappa, \mathsf{LFE.crs})$
$(\mathsf{digest}, r) \leftarrow \mathsf{LFE.Compress}(\mathsf{LFE.crs}, h; s)$
$\mathsf{ct} \leftarrow \mathsf{LFE.Enc}(\mathsf{LFE.crs}, \mathsf{digest}, x)$
Output $b \leftarrow \mathcal{A}_2(\mathtt{st}_\mathcal{A}, \mathsf{ct})$
```

Experiment $\mathsf{Expt}^{\mathsf{ideal}}_{\mathsf{LFE},\mathcal{A},\mathcal{S}}(1^\kappa)$*:*

```
$\mathsf{LFE.crs} \leftarrow \mathsf{LFE.Setup}(1^\kappa)$
$(x, h, s, \mathtt{st}_\mathcal{A}) \leftarrow \mathcal{A}_1(1^\kappa, \mathsf{LFE.crs})$
$(\mathsf{digest}, r) \leftarrow \mathsf{LFE.Compress}(\mathsf{LFE.crs}, h; s)$
$\widetilde{\mathsf{ct}} \leftarrow \mathcal{S}(\mathsf{LFE.crs}, \mathsf{digest}, h, h(x))$
Output $b \leftarrow \mathcal{A}_2(\mathtt{st}_\mathcal{A}, \widetilde{\mathsf{ct}})$
```

The LFE scheme LFE is said to satisfy (semi-malicious)-simulation-security if for any security parameter $\kappa \in \mathbb{N}$, there exists a PPT simulator $\mathcal{S}$ such that for every non-uniform PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the outcomes of the real and ideal experiments are computationally indistinguishable, i.e., we have

$$\left|\Pr[\mathsf{Expt}^{\mathsf{real}}_{\mathsf{LFE},\mathcal{A}}(1^\kappa) = 1] - \Pr[\mathsf{Expt}^{\mathsf{ideal}}_{\mathsf{LFE},\mathcal{A},\mathcal{S}}(1^\kappa) = 1]\right| \leq \mathtt{negl}(\kappa),$$

where $\mathcal{A}$ is admissible if $h \in \mathcal{H}_m$ for some $m = \mathtt{poly}(\kappa)$, and the probability is taken over the random coins of $\mathsf{LFE.Setup}$, $\mathsf{LFE.Compress}$, $\mathsf{LFE.Enc}$, $\mathcal{A}_1$, and $\mathcal{S}$.

Definition 11 (Function-Hiding Security) *Let* $\mathsf{LFE} = (\mathsf{LFE.Setup}, \mathsf{LFE.Compress}, \mathsf{LFE.Enc}, \mathsf{LFE.Dec})$ *be an LFE scheme for a class of functions* $\mathcal{H} = \{\mathcal{H}_m\}_{m\in\mathbb{N}}$*. For every non-uniform PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *and every PPT simulator* $\mathcal{S}$*, consider the following two experiments (*κ *being the security parameter):*

```
Experiment $\mathsf{Expt}^{\mathsf{real,FH}}_{\mathsf{LFE},\mathcal{A}}(1^\kappa)$:

    $\mathsf{LFE.crs} \leftarrow \mathsf{LFE.Setup}(1^\kappa)$
    $(h, \mathtt{st}_\mathcal{A}) \leftarrow \mathcal{A}_1(1^\kappa, \mathsf{mpk})$
    $(\mathsf{digest}, r) \leftarrow \mathsf{LFE.Compress}(\mathsf{LFE.crs}, h)$
    Output $b \leftarrow \mathcal{A}_2(\mathtt{st}_\mathcal{A}, \mathsf{digest})$
```

```
Experiment $\mathsf{Expt}^{\mathsf{ideal,FH}}_{\mathsf{LFE},\mathcal{A},\mathcal{S}}(1^\kappa)$:

    $\mathsf{LFE.crs} \leftarrow \mathsf{LFE.Setup}(1^\kappa)$
    $(h\mathtt{st}_\mathcal{A}) \leftarrow \mathcal{A}_1(1^\kappa, \mathsf{LFE.crs})$
    $\widetilde{\mathsf{digest}} \leftarrow \mathcal{S}(\mathsf{LFE.crs}, \mathcal{F})$
    Output $b \leftarrow \mathcal{A}_2(\mathtt{st}_\mathcal{A}, \widetilde{\mathsf{digest}})$
```

The LFE scheme LFE is said to satisfy function-hiding simulation-security if for any security parameter $\kappa \in \mathbb{N}$, there exists a PPT simulator $\mathcal{S}$ such that for every non-uniform PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the outcomes of the real and ideal experiments are computationally indistinguishable, i.e., we have

$$\left|\Pr[\mathsf{Expt}^{\mathsf{real,FH}}_{\mathsf{LFE},\mathcal{A}}(1^\kappa) = 1] - \Pr[\mathsf{Expt}^{\mathsf{ideal,FH}}_{\mathsf{LFE},\mathcal{A},\mathcal{S}}(1^\kappa) = 1]\right| \leq \mathtt{negl}(\kappa),$$

where $\mathcal{A}$ is admissible if $h \in \mathcal{H}_m$ for some $m = \mathtt{poly}(\kappa)$, and the probability is taken over the random coins of $\mathsf{LFE.Setup}$, $\mathsf{LFE.Compress}$, $\mathcal{A}_1$, and $\mathcal{S}$.

3.2 Succinct Single-key Functional Encryption

We now recall the definition of succinct single-key functional encryption (FE).

Definition 12 (Functional Encryption) *A functional encryption scheme* FE *for a class of functions* $\mathcal{H} = \{\mathcal{H}_m\}_{m\in\mathbb{N}}$ *(represented as Boolean circuits with* m*-bit inputs), is a tuple of four PPT algorithms* $(\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ *such that:*

- $\mathsf{FE.Setup}(1^\kappa) \to (\mathsf{mpk}, \mathsf{msk})$*: On input the security parameter* κ*, the setup algorithm outputs a master public key* mpk *and a master secret key* msk*.*
- $\mathsf{FE.KeyGen}(\mathsf{msk}, h) \to \mathsf{sk}_h$*: On input the master secret key* msk *and a function* $h \in \mathcal{H}$*, the key generation algorithm outputs a key* sk_h*.*

- $\mathsf{FE.Enc}(\mathsf{mpk}, x) \to \mathsf{ct}$*: On input the master public key* mpk *and an input* $x \in \{0,1\}^m$ *for some* $m = \mathtt{poly}(\kappa)$*, the encryption algorithm outputs a ciphertext* ct.
- $\mathsf{FE.Dec}(\mathsf{sk}_h, \mathsf{ct}) \to y$*: On input a key* sk_h *and a ciphertext* ct*, the decryption algorithm outputs a value* y.

In this work, we use single-key FE schemes that satisfy correctness, single-key full-simulation-security and succinctness, as defined formally below.

Definition 13 (Correctness) *Let* $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ *be a single-key FE scheme for a class of functions* $\mathcal{H} = \{\mathcal{H}_m\}_{m\in\mathbb{N}}$*. We say that* FE *is a correct single-key FE scheme if for any* $m = \mathtt{poly}(\kappa)$*, for all* $h \in \mathcal{H}_m$*, and for all* $x \in \{0,1\}^m$*, letting*

$$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{FE.Setup}(1^\kappa), \quad \mathsf{sk}_h \leftarrow \mathsf{FE.KeyGen}(\mathsf{msk}, h), \quad \mathsf{ct} \leftarrow \mathsf{FE.Enc}(\mathsf{mpk}, x),$$

the following holds:

$$\Pr[\mathsf{FE.Dec}(\mathsf{sk}_h, \mathsf{ct}) = h(x)] = 1 - \mathtt{negl}(\kappa),$$

where the probability is taken over the random coins of $\mathsf{FE.Setup}$*,* $\mathsf{FE.KeyGen}$*, and* $\mathsf{FE.Enc}$.

Definition 14 (Full-Simulation Security) *Let* $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ *be a single-key FE scheme for a class of functions* $\mathcal{H} = \{\mathcal{H}_m\}_{m\in\mathbb{N}}$*. For every non-uniform PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *and every PPT simulator* $\mathcal{S}$*, consider the following two experiments (*κ *being the security parameter):*

```
Experiment $\mathsf{Expt}^{\mathsf{real}}_{\mathsf{FE},\mathcal{A}}(1^\kappa)$:

    $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{FE.Setup}(1^\kappa)$
    $(h, \mathtt{st}_\mathcal{A}) \leftarrow \mathcal{A}_1(1^\kappa, \mathsf{mpk})$
    $\mathsf{sk}_h \leftarrow \mathsf{FE.KeyGen}(\mathsf{msk}, h)$
    $(x, \mathtt{st}'_\mathcal{A}) \leftarrow \mathcal{A}_2(\mathtt{st}_\mathcal{A}, \mathsf{sk}_h)$
    $\mathsf{ct} \leftarrow \mathsf{FE.Enc}(\mathsf{mpk}, x)$
    Output $(\mathtt{st}'_\mathcal{A}, \mathsf{ct})$
```

```
Experiment $\mathsf{Expt}^{\mathsf{ideal}}_{\mathsf{FE},\mathcal{A},\mathcal{S}}(1^\kappa)$:

    $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{FE.Setup}(1^\kappa)$
    $(h, \mathtt{st}_\mathcal{A}) \leftarrow \mathcal{A}_1(1^\kappa, \mathsf{mpk})$
    $\mathsf{sk}_h \leftarrow \mathsf{FE.KeyGen}(\mathsf{msk}, h)$.
    $(x, \mathtt{st}'_\mathcal{A}) \leftarrow \mathcal{A}_2(\mathtt{st}_\mathcal{A}, \mathsf{sk}_h)$
    $\widetilde{\mathsf{ct}} \leftarrow \mathcal{S}(\mathsf{mpk}, \mathsf{sk}_h, h(x), 1^{|x|})$
    Output $(\mathtt{st}'_\mathcal{A}, \widetilde{\mathsf{ct}})$
```

The FE scheme FE is said to satisfy (single-key) full-simulation-security if for any security parameter $\kappa \in \mathbb{N}$, there exists a PPT simulator $\mathcal{S}$ such that for every non-uniform PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the outcomes of the real and ideal experiments are computationally indistinguishable, i.e., we have

$$\mathsf{Expt}^{\mathsf{real}}_{\mathsf{FE},\mathcal{A}}(1^\kappa) \approx_c \mathsf{Expt}^{\mathsf{ideal}}_{\mathsf{FE},\mathcal{A},\mathcal{S}}(1^\kappa).$$

Definition 15 (Succinctness) *Let* $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$ *be a single-key FE scheme for a class of functions* $\mathcal{H} = \{\mathcal{H}_m\}_{m\in\mathbb{N}}$*. We say that* FE *is succinct if for any* $m = \mathtt{poly}(\kappa)$*, for all* $h \in \mathcal{H}_m$*, and for all* $x \in \{0,1\}^m$*, letting*

$$(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{FE.Setup}(1^\kappa), \quad \mathsf{ct} \leftarrow \mathsf{FE.Enc}(\mathsf{mpk}, x),$$

the size of the ciphertext ct *(i.e.,* $|\mathsf{ct}|$*) does not grow with the size of the circuit for* h*, but only with its depth.*

3.3 Fully-secure MPC from Laconic Cryptography

In this subsection, we present our construction of TP-aided MPC from LFE.

Construction Overview. The high-level description of the construct, following the three-phase structure (as discussed in Section 1.3), is presented in two steps. In the first step, we assume an honest TP and allow the parties to hand out the inputs to the TP in the clear. In the second step, input privacy against the TP is put in place via *function-hiding* LFE. Throughout, we assume an LFE with a common *random* string (CRS), as is the case for the construction of LFE in [QWW18].

In the first phase, every party uses the common randomness to derive a CRS for the LFE and subsequently computes a digest of f (the function to be computed) using the CRS. It sends the CRS, the digest and its input to the TP. The TP needs to compute an encryption of the collective inputs under the correct digest and CRS. However, a malicious party may send a incorrect digest, say for a function that leaks an honest party's input. The TP can verify the correctness of the digest, since the compress function of the LFE scheme is deterministic. But this amounts to a computation that is dependent on the circuit size, breaking the promise of small TP. To tackle this issue without recomputing the function digest, the TP partitions the set of parties based on the CRS and digest. For every set that sends the same copy of both, gets an encryption under the digest, of the message that consists of the real inputs received from that set and default inputs for those outside that set. This trick ensures that a corrupt party does not get encryption of the inputs of the honest parties under its ill-formed digest. Lastly, on receiving the encryption from the TP, a party simply uses the CRS to learn the function output.

To additionally ensure input privacy against the TP, the function f for LFE is replaced with a related function g that hard-codes n random masks and takes as input n masked inputs of the parties. It first unmasks the masked inputs and then performs the f-computation. The masks are derived from common randomness and thus are known to all. We can use one-time pad for masking. This implies every party has the knowledge of g and can generate a digest that is supposed to be the same. Now, every party uses its respective mask to mask its input before sending to the TP. The TP performs the same computation as before, but now on the received masked inputs, digest for g and CRS. To hide the random masks that are hard-coded inside g from the TP who will learn the digest, we switch to *function-hiding* LFE. This makes sure the TP learns neither about the inputs, not about the output. The LFE security ensures the parties learn nothing but the output of g. The detailed construction is as described below.

Protocol Π_{LFE}

Inputs: Each party P_i has input x_i. All parties share a common randomness of the form $r\|r'$.

Output: $f(x_1, \ldots x_n)$

Primitive: The following building blocks are used

- An LFE scheme $\mathsf{LFE} = (\mathsf{LFE.Setup}, \mathsf{LFE.Compress}, \mathsf{LFE.Enc}, \mathsf{LFE.Dec})$.

Phase 1 (Pre-TP Call): Each party P_i does the following:

- Set $\mathsf{LFE.crs} := r$, where r is obtained from the common randomness $r\|r'$.
- Derive n random pads $\{r_j\}_{j\in[n]}$, where $|r_j| = |x_j|$, using r' obtained from the common randomness $r\|r'$.
- Compute $(\mathsf{digest}^g, r^g) \leftarrow \mathsf{LFE.Compress}\Big(g, \mathsf{LFE.crs}\Big)$, where function g is as follows and send $(\mathsf{LFE.crs}, \mathsf{digest}^g, z_i = x_i \oplus r_i)$ to the TP.
  - g hard-codes the n pads $\{r_j\}_{j\in[n]}$
  - it takes n inputs $z_1, \ldots, z_n$
  - it computes f on input $\{z_j \oplus r_j\}_{j\in[n]}$.

  We note that $(\mathsf{LFE.crs}, \mathsf{digest}^g, r^g)$ is supposed to be the same for all parties, since they use the common randomness r and f.

Phase 2(TP Call): The TP carries out the following computation:

- Initialize the set $\mathcal{Z} = \varnothing$. Add P_j to $\mathcal{Z}$ if nothing (or syntactically incorrect message) is received from P_j.
- Partition the set $\mathcal{P} \setminus \mathcal{Z}$ into subsets $S_1, S_2 \ldots S_\ell$ according to the values of $(\mathsf{LFE.crs}, \mathsf{digest}^g)$ received from the parties i.e. all parties in a subset have sent the same $(\mathsf{LFE.crs}, \mathsf{digest}^g)$.
- For each S_α for $\alpha \in \{1, \ldots, \ell\}$
  - Let $\mathsf{LFE.crs}_\alpha, \mathsf{digest}^g_\alpha$ denote the common values submitted by parties in S_α.
  - For each $j \in \{1, \ldots, n\}$, set $\bar{z}_j = z_j$ if $j \in S_\alpha$, and $\bar{z}_j = z'_j$ otherwise, where z_j is received from P_j and $\{z'_j\}_{j\in\{1,\ldots,n\}}$ are the default (masked) inputs sampled randomly by the TP.
  - Send $\mathsf{ct}_\alpha, S_\alpha$ to every party in S_α, where $\mathsf{ct}_\alpha \leftarrow \mathsf{LFE.Enc}\Big(\mathsf{digest}^g_\alpha, \big(\bar{z}_1, \ldots, \bar{z}_n\big)\Big)$.

Phase 3 (Post-TP Call): A party P_i, on receiving ct, computes output y as

$$y \leftarrow \mathsf{LFE.Dec}\Big(\mathsf{LFE.crs}, \mathsf{ct}, r^g\Big),$$

using $\mathsf{LFE.crs}, r^g$ from Phase 1.

Figure 1: Fully-secure MPC with single TP call based on LFE

Our result can be summarized via the following theorem.

Theorem 16 (TP-Aided MPC from LFE) *Assuming the existence of a laconic function evaluation (LFE) scheme that satisfies correctness, simulation-security and function-hiding security, there exists a TP-aided MPC protocol Π_{LFE} for any functionality f that:*

- *utilizes a single call to a stateless TP of size $\texttt{poly}(n, \kappa, m, \alpha, \beta)$ (where n is the number of parties, κ is the security parameter, m is the size of each party's input to f, and α and β denote the sizes of a single digest and a single ciphertext, respectively, in the LFE scheme), and*
- *achieves full security against malicious corruption of up to $(n-1)$ parties and semi-honest corruption of the TP in the non-colluding model (see Definition 7).*

We defer the formal proof of this theorem to Appendix A. We present here an asymptotic analysis of the size of the TP in our protocol.

TP Size. To begin with, note that the input to the TP from each party P_i is of size $\texttt{poly}(\kappa, m, \alpha)$, where κ is the security parameter (we assume here that the LFE scheme generates $\mathsf{LFE.crs}$ and randomness r of size $\texttt{poly}(\kappa)$), α is the size of the digest generated by the LFE scheme, and m is the size of each party's input to f. Hence, the overall size of inputs received by the TP across all parties is $\texttt{poly}(n, \kappa, m, \alpha)$. Also, the TP generates an LFE ciphertext ct of size, say β, for each partition of parties. Hence, the overall size of the TP is at most $\texttt{poly}(n, \kappa, m, \alpha, \beta)$, as summarized in Theorem 16.

3.4 Fully-secure MPC from Single-Key Succinct FE

In this subsection, we show how to construct TP-aided MPC from single-key succinct FE.

Construction Overview. The high-level description of the construct, following the three-phase structure (as discussed in Section 1.3), is presented in two steps. In the first step, we assume an honest TP and allow the parties to hand out the inputs to the TP. In the second step, input privacy is put in place via a SKE.

For our construction, in the first phase, the parties execute an MPC protocol with identifiable abort[3] amongst the n parties that establishes the setup of the FE and gives the parties sk_f (corresponding to the function f desired to be computed) to aid in output computation. Since this execution may result in abort (where only corrupt parties may get the output), we cannot allow the MPC to output the FE ciphertext corresponding to the parties' inputs directly. Instead, the ciphertext is computed by the TP to whom the parties submit their inputs when Phase 1 is successful (which may need repeated run of the MPC with identifiable abort). To enable the TP to do so, the parties additionally submit mpk (obtained in Phase 1) to the TP. In order to ensure that privacy of honest parties' inputs is maintained against a corrupt party who sends mpk distinct from the one obtained in Phase 1, the TP does the following: partition the set of parties based on the value of mpk they submitted. For each partition, the TP returns ciphertext based on actual inputs of parties within the partition and default otherwise. This ensures that a corrupt party who submits an incorrect mpk (say mpk' which is distinct from the one obtained from Phase 1) never get access to a ciphertext computed using mpk' that involves an honest party's input. Lastly, the parties use the ciphertext obtained from the TP and sk_f to obtain the output.

Note that the above protocol is not secure in the non-colluding model as it does not achieve input privacy against a semi-honest TP. Further, the computation done by the TP grows with the size of the parties' inputs. In order to achieve security against a semi-honest TP and make the computation of the TP independent of the size of the parties' inputs, we make the following modifications. First, the input of each party is hidden in a ciphertext of a SKE. The MPC with identifiable abort now takes as input the inputs of the parties,

[3]Some of the protocols in the literature realizing this functionality for general functions are [GS18].

computes distinct ciphertexts for the inputs, each under a distinct secret key, and delivers only the ith secret key to P_i. Instead of the inputs, these keys are sent to the TP, who performs similar computation as before, but with respect to these keys. To make the both ends meet, the function to be computed by FE is changed to a related function g (instead of the function to be computed f) that hard-codes the ciphertexts of the inputs and takes the n keys as inputs. The function g first decrypts the ciphertexts and then compute f on the decrypted messages. The MPC with identifiable abort now prepares and gives out the secret key of FE corresponding to g. To prevent the parties from tampering the secret keys for SKE while sending to the TP, we use a signature scheme. The MPC samples a (public, secret) key pair for a digital signature scheme and delivers signed messages meant for TP (SKE key and mpk in this case) and the public key for verification to a party. The parties forward this to the TP, who now discards the parties whose verification fails, partitions the parties based on the verification key and proceeds as before. The detailed construction is as described below.

Protocol Π_{FE}

Inputs: Each P_i participates with input x_i.

Output: $f(x_1, \ldots x_n)$

Primitive: The following building blocks are used

- An MPC protocol Π_{idua} that achieves security with identifiable abort.
- A succinct single-key simulation-secure FE scheme $\mathsf{FE} = (\mathsf{FE.Setup}, \mathsf{FE.KeyGen}, \mathsf{FE.Enc}, \mathsf{FE.Dec})$
- An IND-CPA secure symmetric-key encryption scheme $\mathsf{SKE} = (\mathsf{SKE.Gen}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$.
- A digital signature scheme $(\mathsf{Sign}, \mathsf{Vrfy})$.

Phase 1 (Pre-TP Call): Each P_i invokes an instance of Π_{idua} with input x_i to compute a function that does the following:

- Generate a default input x'_i for every P_i.
- Generate a secret key $\mathsf{k}_i \leftarrow \mathsf{SKE.Gen}(1^\kappa)$ for every party P_i.
- Generate $(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{FE.Setup}(1^\kappa)$.
- Generate $e_i \leftarrow \mathsf{SKE.Enc}(\mathsf{k}_i, x_i)$ for every P_i.
- Generate $\mathsf{sk}_g = \mathsf{FE.KeyGen}(\mathsf{msk}, g)$, where g is a function defined as follows:
  - g embeds the ciphertexts $\{e_j\}_{j\in[n]}$ and default inputs $\{x'_j\}_{j\in[n]}$.
  - g takes as input a set of keys $\{\mathsf{k}_j\}_{j\in[n]}$ and an n-length bit vector $\{b_j\}_{j\in[n]}$.
  - g outputs $f(\bar{x}_1, \ldots, \bar{x}_n)$ where for each $j \in [n]$, $\bar{x}_j = \mathsf{SKE.Dec}(\mathsf{k}_i, e_j)$ if $b_i = 1$ and $\bar{x}_j = x'_j$ otherwise.
- Generate $(\mathsf{sk}, \mathsf{vk})$ for the digital signature scheme.
- For each $i \in [n]$, output $(\mathsf{vk}, \mathsf{mpk}, \mathsf{k}_i, \sigma_i, \mathsf{sk}_g)$ to P_i where $\sigma_i = \mathsf{Sign}(\mathsf{sk}, (i, \mathsf{mpk}, \mathsf{k}_i))$.

If Π_{idua} outputs $(\bot, \mathcal{C})$, re-run **Phase 1** among the set of parties $\mathcal{P} \setminus \mathcal{C}$ (the inputs of parties in $\mathcal{C}$ are substituted using default inputs). Else, continue to the next phase. Each P_i invokes the TP with $\mathsf{in}_i = (\mathsf{vk}, \mathsf{mpk}, \mathsf{k}_i, \sigma_i)$.

Phase 2 (TP Call): The TP carries out the following computation:

- Initialize $\mathcal{Z} = \varnothing$. Add P_j to $\mathcal{Z}$ if nothing is received or $\mathsf{Vrfy}(\mathsf{vk}, (j, \mathsf{mpk}, \mathsf{k}_j, \sigma_j) = 0$, for a tuple $(\mathsf{vk}, \mathsf{mpk}, \mathsf{k}_j, \sigma_j)$ received from P_j.
- Partition the set $\mathcal{P} \setminus \mathcal{Z}$ into subsets $S_1, S_2 \ldots S_\ell$ according to the values of vk received from the parties i.e. all parties in a subset have sent the same vk.
- For each S_α for $\alpha \in \{1, \ldots, \ell\}$
  - Let mpk_α denote the common mpk submitted by parties in S_α.
  - For each $j \in [n]$, set $\mathsf{k}_{\alpha,j} = \mathsf{k}_j$ and $b_{\alpha,j} = 1$ if $j \in S_\alpha$, and $\mathsf{k}_{\alpha,j} = \perp$ and $b_{\alpha,j} = 0$ otherwise.
  - Compute and return ct_α to every party in S_α, where

$$\mathsf{ct}_\alpha \leftarrow \mathsf{FE.Enc}\Big(\mathsf{mpk}_\alpha, \big(\{\mathsf{k}_{\alpha,j}\}_{j\in[n]}, \{b_{\alpha,j}\}_{j\in[n]}\big)\Big).$$

Phase 3 (Post-TP Call): A party computes output $y = \mathsf{FE.Dec}\Big(\mathsf{sk}_g, \mathsf{ct}_\alpha\Big)$ using sk_g obtained from Phase 1 and ct_α obtained from Phase 2.

Figure 2: Fully-secure MPC with single TP call based on Succinct Single-Key FE

Our result can be summarized via the following theorem:

Theorem 17 (TP-Aided MPC from Single-Key Succinct FE) *Assuming the existence of an FE scheme that satisfies correctness, (single-key) simulation-security and succinctness, there exists a TP-aided MPC protocol Π_{FE} for any functionality f that:*

- *utilizes a single call to a stateless TP of size $\texttt{poly}(n, \kappa, \beta)$ (where n is the number of parties, κ is the security parameter, and β denotes the size of a single ciphertext in the FE scheme), and*
- *achieves full security against malicious corruption of up to $(n-1)$ parties and semi-honest corruption of the TP in the non-colluding model (see Definition 7).*

We defer the formal proof of this theorem to Appendix B. We present here an asymptotic analysis of the size of the TP in our protocol.

TP Size. To begin with, note that the input to the TP from each party P_i is of size $\texttt{poly}(\kappa)$, where κ is the security parameter. This follows from the following assumptions:

- The FE scheme generates a master public key mpk of size $\texttt{poly}(\kappa)$.
- The digital signature scheme generates a verification key vk and a signature σ_i of size $\texttt{poly}(\kappa)$ each.
- Each symmetric key K_i is of size $\texttt{poly}(\kappa)$.

Hence, the overall size of inputs received by the TP across all parties is $\texttt{poly}(n, \kappa)$. Also, the TP generates an FE ciphertext ct of size, say β, for each partition of parties. Hence, the overall size of the TP is at most $\texttt{poly}(n, \kappa, \beta)$, as summarized in Theorem 17.

4 Impossibilities in the Non-colluding Model

In this section, we present our negative results for small-TP aided MPC.

4.1 Impossibility in the Correlated Randomness Model for protocols with universal output decoder

Here, we make following assumptions– (a) small TP: the TP performs $\texttt{poly}(n, \kappa)$ computation, (b) small output decoder: the parties, on receiving the message from the TP, perform $\texttt{poly}(n, \kappa)$ computation to compute the output. We show that in this model, it is impossible to design a fully secure MPC, even if parties have access to correlated randomness and irrespective of computational assumptions used in the protocol. This holds even if the parties are corrupted in fail-stop fashion in the non-colluding model. Before we begin, we formalize the class of protocols for which the impossibility holds.

Notation. A fully-secure n-party protocol Π in the correlated randomness model that utilizes a single call to a small stateless TP comprises of the following phases.

- **Correlated Randomness Setup.** The setup computes correlated randomness $(\mathsf{cr}_1, \mathsf{cr}_2, \ldots, \mathsf{cr}_n)$ and outputs cr_i to P_i $(i \in [n)$.
- **Pre-TP Computation.** In this phase, the parties may interact with each other (before the TP call), where each P_i participates with input x_i and randomness r_i. Let st_i denotes the state of P_i at the end of this phase, where st_i comprises of its input x_i, randomness r_i, correlated randomness cr_i (received as part of the setup) and in addition, the messages sent / received during this phase, if this phase was interactive. Lastly, each P_i computes algorithm $(\mathsf{in}_i, \mathsf{st}'_i) \leftarrow \mathsf{preTP}_i(\mathsf{st}_i)$ and invokes TP with in_i.
- **TP Computation.** For each $i \in [n]$, the TP computes its response as $\mathsf{out}_i \leftarrow \mathsf{TP}_i(\mathsf{in}_1, \ldots, \mathsf{in}_n; r_{\mathsf{TP}})$, where r_{TP} denotes the internal randomness of the TP and TP_i denotes the algorithm used by the TP to compute its response to P_i.
- **Post-TP Computation.** Each P_i $(i \in [n])$ computes its output as $y \leftarrow \mathsf{postTP}_i(\mathsf{st}'_i, \mathsf{out}_i)$, where postTP_i denotes the algorithm used by P_i to compute its output. We refer to this algorithm as output decoder occasionally.[4]

In our model, (a) each TP_i for $i \in [n]$ is $\texttt{poly}(n, \kappa)$-time (b) each postTP_i for $i \in [n]$ is $\texttt{poly}(n, \kappa)$-time.

To show the impossibility, we show that a fully secure protocol would imply a statistically-correct (encoding, decoding) scheme which can produce an encoding that is smaller than the

[4]We believe that a non-interactive post-TP computation phase is essentially without loss of generality. In other words, any fully secure MPC protocol (having access to one TP call) with interaction amongst the parties can be transformed to one where the parties do not communicate at all amongst themselves after receiving TP's response. We give a proof in Appendix C.

size of the message domain of the encoding scheme. This breaches the known incompressibility argument. Precisely, we use the following proposition of De et al. [DTT10], which formalizes the notion that it is impossible to compress every element in a set X to a string less than $\log|X|$ bits long.

Proposition 18 *[Incompressibility Argument [DTT10]] Let $E : X \times \{0,1\}^\rho \to \{0,1\}^m$ and $D : \{0,1\}^m \times \{0,1\}^\rho \to X$ be randomized encoding and decoding procedures such that, for every $x \in X$, $Pr_{r\in\{0,1\}^\rho}[D(E(x,r),r) = x] \geq \delta$. Then $m \geq \log(|X|) - \log(1/\delta)$.*

Theorem 19 *A general fully secure MPC protocol is impossible in the non-colluding model (see Definition 7), where the parties have access to arbitrary correlated randomness, a single call to a TP of size $\texttt{poly}(n,\kappa)$, and are allowed to use an output decoder of size $\texttt{poly}(n,\kappa)$, even when malicious corruption of parties in $\mathcal{P}$ is restricted to fail-stop corruption.*

Proof: Towards a contradiction, assume such a protocol Π computing an arbitrary function f exists (f is defined later) that achieves full security in the correlated randomness model, satisfying correctness with overwhelming probability. Without loss of generality, Π comprises of the phases (Correlated randomness setup, pre-TP computation, TP computation, post-TP computation) described previously.

Below, we show that Π leads to a statistically-correct randomized (encoding, decoding) scheme (E, D) (as defined in Proposition 18).

Algorithm (E, D)

$E : \{0,1\}^{2^{n-1}} \times \{0,1\}^\rho \to \{0,1\}^m$**:** This algorithm takes as input 2^{n-1} bits, say $(b_1, b_2, \ldots, b_{2^{n-1}})$, an randomness $r \in \{0,1\}^\rho$ and computes its encoding as follows:

1. For each $i \in [n]$, choose a pair of inputs (x_i, x_i^*) using r.
2. Consider a set S containing tuples of the form $(x_1, x_2', \ldots, x_n')$ where $x_i' \in \{x_i, x_i^*\}$ for $i \in \{2, \ldots, n\}$. Note that x_1 is fixed in all the tuples and $|S| = 2^{n-1}$.
3. Consider a lexicographic ordering of the elements in S generated as follows. For each $i \in [n]$, map x_i to bit 0 and x_i^* to bit 1. Now each tuple in S can be viewed as an n bit string and the elements in S can be lexicographically ordered. Let us denote the jth element as S_j. Let M be a mapping between S and $(b_1, b_2, \ldots, b_{2^{n-1}})$, where S_j is mapped to b_j for $j \in [2^{n-1}]$.
4. Construct an n-input function $f(X_1, \ldots, X_n)$ that outputs $M\big(X_1, \ldots, X_n\big)$, when $(X_1, \ldots, X_n) \in S$ and $\perp$ otherwise.
5. Suppose Π computes f on input X_i from P_i. Consider an execution of Π where parties $\{P_1, \ldots, P_n\}$ participate using inputs $\{x_i\}_{i\in[n]}$, randomness $\{r_i\}_{i\in[n]}$ and correlated randomness $\{\mathsf{cr}_i\}_{i\in[n]}$ (the latter two picked using r). Further, Π uses x_i^* as the default input of P_i ($i \in [n]$). Emulate the steps of this execution until the pre-TP computation to obtain $\{\mathsf{st}_i', \mathsf{in}_i\}_{i\in[n]}$. Let $\bar{\mathsf{st}}_1'$ denote the subset of st_1' used in postTP_1; with size restricted to $\texttt{poly}(n,\kappa)$, as dictated by Π (recall that postTP function is allowed to do only $\texttt{poly}(n,\kappa)$ computation).
6. The encoding of input $(b_1, b_2, \ldots, b_{2^{n-1}})$ is defined as $\{\bar{\mathsf{st}}_1', \mathsf{in}_1, \ldots, \mathsf{in}_n\}$, TP_1 (the algorithm used by the TP to compute its response to P_1) and postTP_1 (the output computation algorithm of P_1).

$D : \{0,1\}^m \times \{0,1\}^\rho \to \{0,1\}^{2^{n-1}}$**:** It takes as input the encoding $\{\bar{\mathsf{st}}_1', \mathsf{in}_1, \ldots, \mathsf{in}_n\}$ and the $r \in \{0,1\}^\rho$ used by E. For each subset $S' \subseteq \{2, \ldots, n\}$ in lexicographic order (starting from $S' = \varnothing$ to $S' = \{2, \ldots, n\}$), do the following (below we abuse the notation and use S' to denote the decimal value corresponding to the binary representation):

1. Compute $\mathsf{out}_1^{(S')} \leftarrow \mathsf{TP}_1(\mathsf{in}_1', \mathsf{in}_2', \ldots, \mathsf{in}_n'; r_{\mathsf{TP}})$, where $\mathsf{in}_i' = \mathsf{in}_i$ for $i \notin S'$ [a] and $\mathsf{in}_i' = \perp$ for $i \in S'$. Here, r_{TP} is derived from r as per the distribution corresponding to the internal randomness of the TP in Π.
2. Compute $b_{(S')} \leftarrow \mathsf{postTP}_1(\bar{\mathsf{st}}_1', \mathsf{out}_1^{(S')})$.

Output $(b_1, b_2, \ldots, b_{2^{n-1}})$.

[a]Note $\mathsf{in}_1' = \mathsf{in}_1$ is always satisfied as S' is defined as subsets of $\{2, \ldots, n\}$.

Figure 3: A Randomized Encoding and Decoding Scheme

Lemma 20 *(E, D) is a statistically-correct encoding and decoding scheme.*

Proof: We now claim that the above pair (E, D) is statistically correct. That is the following holds good: for every $(b_1, \ldots, b_{2^{n-1}}) \in \{0,1\}^{2^{n-1}}$, $Pr_{r \in \{0,1\}^\rho}[D(E((b_1, \ldots, b_{2^{n-1}}), r), r) = (b_1, \ldots, b_{2^{n-1}})] \geq \delta$. This is because Π computes f that, for every input in S, as defined in E, maps to one distinct bit in the sequence $(b_1, \ldots, b_{2^{n-1}})$ (recall that the jth element of S, S_j is mapped to b_j). Further, Π computes f and achieves full security (guaranteed output delivery) and satisfies correctness with overwhelming probability. Specifically, if a subset of parties P_i such that $i \in S'$ do not invoke the TP during Π, then the TP receives $\{\mathsf{in}_i\}$ only from the other parties P_i where $i \notin S'$ and sets $\mathsf{in}_i = \perp$ for parties in S'. The output computed by the TP is on the default input x_i^* for each party P_i with $i \in S'$ and x_i for each party P_i with $i \notin S'$.

Since S' is defined as subsets of $\{2, \ldots, n\}$ and never includes the index 1, the above captures executions of Π where P_1 is honest, participated honestly with input x_1 and invoked the TP with $\mathsf{in}_1' = \mathsf{in}_1$. This allows us to rely on the correctness of the output computed by postTP_1. We can thus infer that the 2^{n-1} bits computed during decoding indeed correspond to the set of outputs of f for each subset S', namely $(b_1, b_2, \ldots, b_{2^{n-1}})$.

Notice that the above argument holds good even if Π satisfies full security tolerating fail-stop corruption where the parties do not send their message to the TP. Furthermore, Π satisfying fairness is not enough to claim that (E, D) is (statistically) correct, because D may fail to recover $(b_1, \ldots, b_{2^{n-1}})$ always.

□

By the incompressibility argument of [DTT10] (which is formally stated above), it must hold that $|\bar{\mathsf{st}}_1'| + |\mathsf{in}_1| + \ldots |\mathsf{in}_n| + |\mathsf{out}_1| + |\mathsf{postTP}_1| \geq 2^{n-1}$. We can thus infer that at least one of the terms $\geq \frac{2^{n-1}}{n+3}$. Recall that by our assumption on small output decoder, the terms $|\bar{\mathsf{st}}_1'|$ and $|\mathsf{postTP}_1|$ are bounded by size $\mathtt{poly}(n, \kappa)$. Therefore, it must be the case that one of the terms $\mathsf{in}_1, \ldots, \mathsf{in}_n$, out_1 must be of size $\geq \frac{2^{n-1}}{n+3}$. However, this contradicts our assumption that the TP has size $\mathtt{poly}(n, \kappa)$ as $\mathsf{in}_1, \ldots, \mathsf{in}_n$ comprises of the input to the

TP and $\texttt{out}_1$ is the algorithm run by the TP to compute its response to P_1. We have thus arrived at a contradiction; completing the proof.

□

4.2 Impossibility in the Plain Model

In this section, we show that in the plain model (without correlated randomness), it is impossible to design statistically secure MPC with the non-colluding security, even when the parties are only semi-honestly corrupt. That is, we prove that a protocol is impossible when the adversary in the non-colluding TP model can either (a) corrupt majority of the parties $\{P_1, \ldots, P_n\}$ semi-honestly or (b) control the TP semi-honestly). We state the formal theorem below.

Theorem 21 *A general statistically-secure MPC protocol is impossible in the plain and the non-colluding TP model (see Definition 7), where the parties have access to a single call to a small TP of size* $\texttt{poly}(n, \kappa)$*, even when malicious corruption of parties in* $\mathcal{P}$ *is restricted to semi-honest corruption.*

Proof. Towards a contradiction, assume that there exists a statistically-secure 2-party protocol Π securely computing f against a semi-honest adversary in the non-colluding TP model. Let f be defined as the functionality computing $(k+1)$ oblivious transfer (OT) instances i.e.

$$f\big(x_1 = (m_i^0, m_i^1)_{i \in [k+1]}, x_2 = (b_1, \ldots, b_k, b_{k+1})\big) = (m_1^{b_1}, m_2^{b_2} \ldots, m_{k+1}^{b_{k+1}})$$

Here, the input of P_1 (who acts as the sender) consists of $(k+1)$ pairs of bits and the input of P_2 (who acts as the receiver) consists of $(k+1)$ bits.

Suppose C_{TP} denotes the circuit describing the function $\{\mathsf{TP}\}_{i \in [n]}$ computed by the TP during Π. Based on our assumption that the TP is 'small', it must hold that $|C_{\mathsf{TP}}| \leq \texttt{poly}(n, \kappa)$ which is independent of the function f being computed. Specifically, this means that the computation done by the TP must be strictly less than computing $(k+1)$ OTs.

We claim that Π computing f can be used to build a semi-honest OT extension protocol Π'. Assume a semi-honest setting where the parties are given k OT correlations generated as the base OTs of the OT extension protocol Π'. Π' proceeds as follows:

1. The parties execute the steps of Π in the pre-TP computation phase.
2. Next, the parties emulate the TP computation phase of Π by executing the perfectly-secure semi-honest GMW protocol [GMW87] to compute the function described by C_{TP}. For this, the parties use the k OT correlations (given as base OTs). Note that these OT correlations must suffice as computing C_{TP} must involve computing fewer than $(k+1)$ OTs (based on our assumption).

3. Finally, the parties use the output of the execution of the GMW protocol (which computes the TP response of Π) to carry out the steps of output computation as per Π. This will result in the parties obtaining the output of f.

We note that Π' does not involve any calls to the stateless TP. Since Π' computes $(k+1)$ OTs starting with k base OTs and involves execution of steps in Π and the GMW protocol, which are both information-theoretically secure; we can conclude that Π' is indeed a semi-honest information-theoretic OT extension protocol. However, this is a contradiction as information-theoretically secure OT extension does not exist in the plain model [Bea96]. This completes the proof.

5 Impossibilities in the Colluding Model

In this section, we present some negative results in the colluding security model (see Definition 7). Recall that, in this model, we assume that the adversary controls a majority of the parties among $\{P_1, \ldots, P_n\}$ maliciously and simultaneously corrupt the TP semi-honestly. Our impossibility holds good even when malicious corruption is weakened to fail-stop corruption and the requirement of full security is relaxed to fairness.

5.1 Impossibility of Fair MPC

At a high level, we follow the following route. We note that the colluding adversarial model can be viewed more generally, in terms of the general mixed adversarial model that has been studied in works such as [HMZ08, FHM99, BFH+08]. We begin with the details of mixed adversaries below.

A general mixed adversary is characterized by an adversary structure

$$\mathbb{Z} = \{(A_1, E_1, F_1), \ldots, (A_m, E_m, F_m)\},$$

(for some m) which is a monotone set of triples of party sets. At the beginning of the protocol, the adversary chooses one of these triples $\mathbb{Z}^* = (A^*, E^*, F^*) \in \mathbb{Z}$ and actively corrupts parties in A^*, semi-honestly corrupts the parties in E^* and fail-corrupts the parties in F^* (i.e. the adversary can make the fail-corrupt parties crash at any time during the protocol). For any triple (A, E, F), it is assumed that $A \subseteq F$ and $A \subseteq E$, since any actively corrupted party can behave as passive or fail-corrupted.

Adversary structure for colluding model. If we view the TP as an additional party P_{n+1} (who can be semi-honestly corrupted) and the party set $\mathcal{P} = \{P_1, \ldots, P_n, P_{n+1}\}$, the adversarial structure for the colluding TP model can be expressed as: $\mathbb{Z} = \big\{\mathbb{Z}_1, \ldots, \mathbb{Z}_n\big\}$, where for each $i \in [n]$

$$\mathbb{Z}_i = \Big(A_i = \mathcal{P} \setminus \{P_i, P_{n+1}\}, E_i = \mathcal{P} \setminus \{P_i\}, F_i = \mathcal{P} \setminus \{P_i, P_{n+1}\}\Big).$$

Specifically, the above denotes the maximal class of the adversarial structure of the colluding TP model, since these subsume all other possible corruption scenarios indicated by subsets of the triples in each $\mathbb{Z}_i$. I.e. the adversary can choose to corrupt (A^*, E^*, F^*), such that there exists $(\bar{A}, \bar{E}, \bar{F}) \in \mathbb{Z} : A^* \subseteq \bar{A}, E^* \subseteq \bar{E}, F^* \subseteq \bar{F}$.

Now restricting the malicious adversaries to behave in a fail-stop manner, we refine the maximal adversarial structure as $\mathbb{Z}' = \{\mathbb{Z}'_1, \ldots, \mathbb{Z}'_n\}$, where for each $i \in [n]$,

$$\mathbb{Z}'_i = \Big(A_i = \varnothing, E_i = \mathcal{P} \setminus \{P_i\}, F_i = \mathcal{P} \setminus \{P_i, P_{n+1}\}\Big).$$

We observe that in the regime of mixed adversarial model, we consider the TP to be just another party that can communicate freely with the other parties while maintaining states across the communication. This is a more liberal setting than the stateless TPs that we consider in this work. This implies that our impossibility holds even for state-full TPs.

We are now ready to prove our impossibility.

Theorem 22 *There exists a function f such that it is impossible to design a fair MPC protocol securely computing f in the computational colluding model (see Definition 7) even when malicious corruption of parties in $\mathcal{P}$ is restricted to fail-stop corruption.*

Proof. [HMZ08] showed that fair (non-reactive) MPC is possible in the computational setting against an adversary characterized by $\mathbb{Z}$ only if there exists an ordering $\big((A_1, E_1, F_1), \ldots, (A_m, E_m, F_m)\big)$ of the maximal adversarial class such that $\forall i, j \in [m]$, $i \leq j$: $E_j \cup F_i \neq \mathcal{P}$.

Consider the maximal adversarial structure $\mathbb{Z}' = \{\mathbb{Z}'_1, \ldots, \mathbb{Z}'_n\}$, where for each $i \in [n]$, $\mathbb{Z}'_i = \big(A_i = \varnothing, E_i = \mathcal{P} \setminus \{P_i\}, F_i = \mathcal{P} \setminus \{P_i, P_{n+1}\}\big)$. For $\mathbb{Z}'$, we note that such an ordering does not exist. For contradiction, consider that such an ordering exists with the triple $\mathbb{Z}'_k = \big(A_k = \varnothing, E_k = \mathcal{P} \setminus \{P_k\}, F_k = \mathcal{P} \setminus \{P_k, P_{n+1}\}\big)$ as the last triple in the ordered set of triples. Then for any $i < k$, $E_k \cup F_i = \mathcal{P}$ would hold because F_i for any $i \neq k$ comprises of the party P_k. We can thus conclude that the condition of [HMZ08] does not hold, implying that fair MPC is impossible against the adversary characterized by $\mathbb{Z}'$ i.e in the colluding TP model.

5.2 Randomized Function without Inputs

While the above negative result (Theorem 22) rules out fair MPC for arbitrary functions in the colluding model, there may be special classes of functions that can be computed fairly in the colluding model. Specifically, we observe that certain useful functions such as randomized functions without inputs (such as "coin-tossing") can be fairly computed using the TP. Since such functions do not involve private inputs from the parties that need to be protected from the adversary, the TP can be simply used as a trusted third-party (which directly computes the desired randomized function and returns the output to all).

While the above trivial solution does in fact gives full security (stronger than fairness), it involves computation by the TP which is as much as computing the desired function itself. This opens up the question regarding whether this is inherent. We present here a high level intuition for why we believe that this is likely to be inherent for the functionality of coin tossing. We begin by noting that there exists a fairness and computation preserving transformation that compiles any generic TP-aided fair protocol in the colluding model to a fair protocol with a single TP call at the beginning of the protocol. One can then design a series of specialized adversarial strategies where the adversary aborts during sequential rounds, depending on the response of the first (and only) TP call (at a high level, this is similar to the sequence of adversarial strategies used in the well-known impossibility result for fair MPC due to Cleve [Cle86]). This seems to indicate that once the TP invocation is completed, the distribution of the outputs does not change. Intuitively, this seems to capture that interaction post TP invocation does not help in output computation and therefore it seems likely that the computation done by the TP is as much as computing the function itself. We elaborate on the details in Appendix D.

Acknowledgments

We thank the anonymous reviewers of IACR TCC 2022 for their helpful comments and suggestions. Y. Ishai was supported in part by ERC Project NTSC (742754), BSF grant 2018393, and ISF grant 2774/20. A. Patra would like to acknowledge financial support from DST National Mission on Interdisciplinary Cyber-Physical Systems (NM-ICPS) 2020-2025. A. Srinivasan was supported in part by a SERB startup grant. D. Ravi was funded by the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 803096 (SPEC).

References

[ABMO15] Gilad Asharov, Amos Beimel, Nikolaos Makriyannis, and Eran Omri. Complete characterization of fairness in secure two-party computation of boolean functions. In *TCC 2015*, volume 9014 of *Lecture Notes in Computer Science*, pages 199–228. Springer, 2015.

[ADMM14] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Lukasz Mazurek. Secure multiparty computations on bitcoin. In *IEEE SP 2014*, pages 443–458. IEEE Computer Society, 2014.

[Bea96] Donald Beaver. Correlated pseudorandomness and the complexity of private computations. In *ACM STOC 1996*, pages 479–488. ACM, 1996.

[BFH$^+$08] Zuzana Beerliová-Trubíniová, Matthias Fitzi, Martin Hirt, Ueli M. Maurer, and Vassilis Zikas. MPC vs. SFE: perfect security in a unified corruption model. In *TCC 2008*, volume 4948 of *Lecture Notes in Computer Science*, pages 231–250. Springer, 2008.

[BFSK11] Christina Brzuska, Marc Fischlin, Heike Schröder, and Stefan Katzenbeisser. Physically uncloneable functions in the universal composition framework. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 51–70. Springer, 2011.

[BGI$^+$01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2001.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Innovations in Theoretical Computer Science 2012*, 2012.

[BJOV18] Saikrishna Badrinarayanan, Abhishek Jain, Rafail Ostrovsky, and Ivan Visconti. Non-interactive secure computation from one-way functions. In *ASIACRYPT 2018*, pages 118–138, 2018.

[BK14] Iddo Bentov and Ranjit Kumaresan. How to use bitcoin to design fair protocols. In *CRYPTO 2014*, volume 8617 of *Lecture Notes in Computer Science*, pages 421–439. Springer, 2014.

[BKOV17] Saikrishna Badrinarayanan, Dakshita Khurana, Rafail Ostrovsky, and Ivan Visconti. Unconditional uc-secure computation with (stronger-malicious) pufs. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 382–411, 2017.

[BLOO20] Amos Beimel, Yehuda Lindell, Eran Omri, and Ilan Orlov. 1/p-secure multiparty computation without an honest majority and the best of both worlds. *J. Cryptol.*, 33(4):1659–1731, 2020.

[BOO15] Amos Beimel, Eran Omri, and Ilan Orlov. Protocols for multiparty coin toss with a dishonest majority. *J. Cryptol.*, 28(3):551–600, 2015.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC 2011*, volume 6597 of *Lecture Notes in Computer Science*, pages 253–273. Springer, 2011.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS*, 2001.

[CCOV19] Nishanth Chandran, Wutichai Chongchitmate, Rafail Ostrovsky, and Ivan Visconti. Universally composable secure computation with corrupted tokens. In *CRYPTO 2019*, pages 432–461, 2019.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In *CRYPTO 2001*, pages 19–40, 2001.

[CGJ$^+$17] Arka Rai Choudhuri, Matthew Green, Abhishek Jain, Gabriel Kaptchuk, and Ian Miers. Fairness in an unfair world: Fair multiparty computation from public bulletin boards. In *ACM CCS 2017*, pages 719–728. ACM, 2017.

[CGS08] Nishanth Chandran, Vipul Goyal, and Amit Sahai. New constructions for UC secure computation using tamper-proof hardware. In *EUROCRYPT 2008*, pages 545–562, 2008.

[CKS$^+$14] Seung Geol Choi, Jonathan Katz, Dominique Schröder, Arkady Yerukhimovich, and Hong-Sheng Zhou. (efficient) universally composable oblivious transfer using a minimal number of stateless tokens. In *TCC 2014*, pages 638–662, 2014.

[Cle86] Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *ACM STOC*, 1986.

[DMRV13] Dana Dachman-Soled, Tal Malkin, Mariana Raykova, and Muthuramakrishnan Venkitasubramaniam. Adaptive and concurrent secure computation from new adaptive, non-malleable commitments. In *ASIACRYPT 2013*, pages 316–336, 2013.

[DTT10] Anindya De, Luca Trevisan, and Madhur Tulsiani. Time space tradeoffs for attacks against one-way functions and prgs. In *CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 649–665. Springer, 2010.

[EGL85] Shimon Even, Oded Goldreich, and Abraham Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, 1985.

[FGMO01] Matthias Fitzi, Juan A. Garay, Ueli M. Maurer, and Rafail Ostrovsky. Minimal complete primitives for secure multi-party computation. In *CRYPTO 2001*, pages 80–100, 2001.

[FHM99] Matthias Fitzi, Martin Hirt, and Ueli M. Maurer. General adversaries in unconditional multi-party computation. In *ASIACRYPT 1999*, volume 1716 of *Lecture Notes in Computer Science*, pages 232–246. Springer, 1999.

[FKN94] Uriel Feige, Joe Kilian, and Moni Naor. A minimal model for secure computation (extended abstract). In *ACM STOC 1994*, pages 554–563, 1994.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC 2009*, pages 169–178. ACM, 2009.

[GGH$^+$13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *IEEE FOCS 2013*, pages 40–49. IEEE Computer Society, 2013.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *ACM STOC 2013*, pages 467–476. ACM, 2013.

[GHKL11] S. Dov Gordon, Carmit Hazay, Jonathan Katz, and Yehuda Lindell. Complete fairness in secure two-party computation. *J. ACM*, 58(6):24:1–24:37, 2011.

[GIM$^+$10] S. Dov Gordon, Yuval Ishai, Tal Moran, Rafail Ostrovsky, and Amit Sahai. On complete primitives for fairness. In *TCC 2010*, pages 91–108, 2010.

[GK09] S. Dov Gordon and Jonathan Katz. Complete fairness in multi-party computation without an honest majority. In *TCC 2009*, volume 5444 of *Lecture Notes in Computer Science*, pages 19–35. Springer, 2009.

[GK12] S. Dov Gordon and Jonathan Katz. Partial fairness in secure two-party computation. *J. Cryptol.*, 25(1):14–40, 2012.

[GKP$^+$13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *STOC 2013*, pages 555–564, 2013.

[GMPY11] Juan A. Garay, Philip D. MacKenzie, Manoj Prabhakaran, and Ke Yang. Resource fairness and composability of cryptographic protocols. *J. Cryptol.*, 24(4):615–658, 2011.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *ACM STOC*, 1987.

[GS18] Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In *EUROCRYPT 2018*, pages 468–499, 2018.

[HMZ08] Martin Hirt, Ueli M. Maurer, and Vassilis Zikas. MPC vs. SFE : Unconditional and computational security. In *ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2008.

[HPV16] Carmit Hazay, Antigoni Polychroniadou, and Muthuramakrishnan Venkitasubramaniam. Composable security in the tamper-proof hardware model under minimal complexity. In *TCC 2016*, pages 367–399, 2016.

[IOS12] Yuval Ishai, Rafail Ostrovsky, and Hakan Seyalioglu. Identifying cheaters without an honest majority. In *TCC 2012*, pages 21–38, 2012.

[JLS21] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *STOC '21*, pages 60–73, 2021.

[Kat07] Jonathan Katz. Universally composable multi-party computation using tamper-proof hardware. In *EUROCRYPT 2007*, pages 115–128, 2007.

[KB14] Ranjit Kumaresan and Iddo Bentov. How to use bitcoin to incentivize correct computations. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 30–41. ACM, 2014.

[O'N10] Adam O'Neill. Definitional issues in functional encryption. *IACR Cryptol. ePrint Arch.*, page 556, 2010.

[OSVW13] Rafail Ostrovsky, Alessandra Scafuro, Ivan Visconti, and Akshay Wadia. Universally composable secure computation with (malicious) physically uncloneable functions. In *EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 702–718. Springer, 2013.

[PST17] Rafael Pass, Elaine Shi, and Florian Tramèr. Formal abstractions for attested execution secure processors. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017*, volume 10210 of *Lecture Notes in Computer Science*, pages 260–289, 2017.

[QWW18] Willy Quach, Hoeteck Wee, and Daniel Wichs. Laconic function evaluation and applications. In Mikkel Thorup, editor, *IEEE FOCS 2018*, pages 859–870. IEEE Computer Society, 2018.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer, 2005.

[Wat15] Brent Waters. A punctured programming approach to adaptively secure functional encryption. In *CRYPTO 2015*, volume 9216, pages 678–697. Springer, 2015.

A Proof of Theorem 16 (Full Security of LFE-based TP-Aided MPC)

In this section, we prove Theorem 16 on the full security of our LFE-based TP-Aided MPC protocol. As per the formal definition of full security for TP-aided MPC, our proof is divided into two cases for two distinct corruption scenarios:

- Corruption scenario-1: a semi-honest adversary $\mathcal{A}$ corrupts the TP, while the parties $P_1, \ldots, P_n$ are honest.
- Corruption scenario-2: a malicious adversary $\mathcal{A}$ corrupts a subset $\mathcal{C} \subset [n]$ of the parties $P_1, \ldots, P_n$ such that $|\mathcal{C}| \leq (n-1)$, while the TP is honest.

We tackle each of these cases separately below.

A.1 Case-1: Semi-Honest Corruption of TP

We first consider the case where the TP is corrupted by a semi-honest adversary $\mathcal{A}$, while the parties $P_1, \ldots, P_n$ are honest. In this setting, we construct a simulator $\mathcal{S}_\Pi$ that simulates the view of the adversary $\mathcal{A}$ without the knowledge of the inputs $x_1, \ldots, x_n$ for the honest parties.

Recall that in our protocol, the TP does not have any inputs nor does it receive any outputs at the end of the protocol; consequently, it suffices for $\mathcal{S}_\Pi$ to simulate the messages from the (honest) parties $P_1, \ldots, P_n$ to the TP. To this end, $\mathcal{S}_\Pi$ acts on behalf of each (honest) party P_i while interacting with an ideal-world simulator – a simulator $\mathcal{S}_{\mathsf{LFE,FH}}$ for the LFE scheme in the function-hiding security experiment – as follows:

- $\mathcal{S}_\Pi$ sets $\mathsf{LFE.crs} := r$, where $r\|r'$ is the common randomness.
- $\mathcal{S}_\Pi$ samples a uniformly random string $\widetilde{z}_i$, where $|\widetilde{z}_i| = |x_i|$, on behalf of each (honest) party P_i.
- $\mathcal{S}_\Pi$ then invokes the simulator $\mathcal{S}_{\mathsf{LFE,FH}}$ for the LFE scheme in the function-hiding security experiment to generate

$$\widetilde{\mathsf{digest}} \leftarrow \mathcal{S}_{\mathsf{LFE,FH}}(\mathsf{LFE.crs}, \mathcal{F}).$$

Finally, on behalf of each (honest) party P_i, $\mathcal{S}_\Pi$ sends across to the semi-honest adversary $\mathcal{A}$ corrupting the TP the message

$$\mathsf{in}_i = (\mathsf{LFE.crs}, \widetilde{\mathsf{digest}}, \widetilde{z}_i).$$

Eventually, the adversary $\mathcal{A}$ outputs some string $s_\mathcal{A}$, which $\mathcal{S}_\Pi$ forwards to the environment $\mathcal{Z}$ to complete the simulation.

We argue that the view of the semi-honest adversary $\mathcal{A}$ corrupting the TP is computationally indistinguishable in the real and ideal worlds assuming that the LFE scheme satisfies function-hiding simulation-security. By extension, we claim that for any circuit family $\mathcal{Z} = \{z_\kappa\}$ corrupting the TP P^* semi-honestly, we have

$$\left\{\mathsf{real}_{\Pi,\mathcal{C},\mathcal{A}(\mathsf{aux})}(\vec{x},\kappa)\right\}_{\vec{x}\in(\{0,1\}^*)^n,\kappa\in\mathbb{N}} \approx_c \left\{\mathsf{ideal}^{\mathsf{type}}_{f,\mathcal{C},\mathcal{S}(\mathsf{aux})}(\vec{x},\kappa)\right\}_{\vec{x}\in(\{0,1\}^*)^n,\kappa\in\mathbb{N}},$$

whenever the LFE scheme satisfies function-hiding simulation-security. The argument uses the following hybrids:

- **Hybrid-0:** This hybrid is identical to the real security experiment.
- **Hybrid-1:** In this hybrid, we switch the digest to be sent across to the semi-honest adversary $\mathcal{A}$ from digest as in the real scheme, to $\widetilde{\mathsf{digest}}$ as in the simulation strategy described above.
- **Hybrid-2:** In this hybrid, we switch the padded input string to be sent across to the semi-honest adversary $\mathcal{A}$ as follows: for each (honest) party P_i, we switch it from $z_i = x_i \oplus r_i$ as in the real scheme, to a uniformly random string $\widetilde{z}_i$ such that $|\widetilde{z}_i| = |x_i|$, as in the simulation strategy described above.

Hybrid-0 $\approx_c$ Hybrid-1: We argue the following: assuming that the LFE scheme satisfies function-hiding security, the views of the adversary $\mathcal{A}$ (and hence, the environment $\mathcal{Z}$) in Hybrid-0 and Hybrid-1 are computationally indistinguishable. To see this, suppose that there exists some $(\mathcal{Z}, \mathcal{A})$ pair such that their views in Hybrid-0 and Hybrid-1 are distinguishable with non-negligible advantage ϵ. We construct an algorithm $\mathcal{B}$ that can break function hiding security of the LFE scheme with non-negligible advantage $\epsilon' = \epsilon$.

The algorithm $\mathcal{B}$ samples a function f and receives the inputs $x_1, \ldots, x_n$ for the honest parties $P_1, \ldots, P_n$ from the environment $\mathcal{Z}$, and proceeds as follows:

- $\mathcal{B}$ receives from the challenger in the LFE function-hiding security game a common random string $\mathsf{LFE.crs} = r$ and uses the same random string r in the simulation.

- $\mathcal{B}$ samples additional common randomness r' and uses r' this to derive the random pads $\{r_j\}_{j\in[n]}$ as in the "real" scheme and sets $z_j = x_j \oplus r_j$ for each $j \in [n]$.

- $\mathcal{B}$ outputs to the challenger in the LFE function-hiding security game the function g (where g is as described in the "real" scheme).

- The challenger responds to $\mathcal{B}$ with a digest $\widehat{\mathsf{digest}}$.

- On behalf of each (honest) party P_i, $\mathcal{B}$ sends across to the semi-honest adversary $\mathcal{A}$ corrupting the TP the message

$$\mathsf{in}_i = (\mathsf{LFE.crs}, \widetilde{\mathsf{digest}}, \widetilde{z}_i).$$

- Eventually, the adversary $\mathcal{A}$ outputs some string $s_{\mathcal{A}}$, which $\mathcal{B}$ forwards to the environment $\mathcal{Z}$ to complete the simulation.

- Finally, the environment $\mathcal{Z}$ outputs a bit b. $\mathcal{B}$ outputs the same bit b.

It is easy to see that:

- When $\widehat{\mathsf{digest}}$ is sampled by the challenger from the "real world" distribution, i.e., $\widehat{\mathsf{digest}} = \mathsf{digest}$, the view of the adversary $\mathcal{A}$ (and hence the environment $\mathcal{Z}$) is exactly as in Hybrid-0.

- When $\widehat{\mathsf{digest}}$ is sampled by the challenger from the "ideal world" distribution, i.e., $\widehat{\mathsf{digest}} = \widetilde{\mathsf{digest}}$, the view of the adversary $\mathcal{A}$ (and hence the environment $\mathcal{Z}$) is exactly as in Hybrid-1.

Hence, if $\mathcal{A}$ (and hence $\mathcal{Z}$) has non-negligible advantage ϵ in distinguishing Hybrid-0 and Hybrid-1, then $\mathcal{B}$ has non-negligible advantage $\epsilon' = \epsilon$ in breaking the function privacy of the LFE scheme. This yields a contradiction, as desired.

Hybrid-1 $\approx_s$ Hybrid-2: We now argue that the views of the adversary $\mathcal{A}$ (and hence, the environment $\mathcal{Z}$) in Hybrid-1 and Hybrid-2 are statistically indistinguishable, as desired. To see this, observe that in Hybrid-1, the simulated digest $\widetilde{\mathsf{digest}}$ has no information about the common randomness $r\|r'$ and the corresponding derived randomness pads r_i. As a result, from the point of view of the adversary $\mathcal{A}$, each padded string z_i perfectly hides the input x_i of the (honest) party P_i. In other words, each z_i is statistically indistinguishable from a uniformly random string $\widetilde{z}_i$ such that $|\widetilde{z}_i| = |x_i|$. It immediately follows that the views of the adversary $\mathcal{A}$ in Hybrid-1 and Hybrid-2 are statistically indistinguishable, as desired.

Thus, it follows that the view of the semi-honest adversary $\mathcal{A}$ corrupting the TP is computationally indistinguishable in the real and ideal worlds assuming that the LFE scheme satisfies function-hiding simulation-security, as desired.

A.2 Case-2: Malicious Corruption of Majority of Parties

We now consider the case where a malicious adversary $\mathcal{A}$ corrupts a subset $\mathcal{C} \subset [n]$ of the parties $P_1, \ldots, P_n$ such that $|\mathcal{C}| \leq (n-1)$, while the TP is honest. The simulation strategy in this case is substantially more involved.

We again construct an ideal-world simulator $\mathcal{S}_\Pi$, where $\mathcal{S}_\Pi$ now interacts with a different ideal-world simulator – a simulator $\mathcal{S}_{\mathsf{LFE}}$ for the LFE scheme in the standard simulation-security experiment. We divide the simulation strategy for $\mathcal{S}_\Pi$ into two phases - the pre-TP call phase and the TP-call phase. Note that since the TP is honest, $\mathcal{S}_\Pi$ also acts on behalf of the TP in addition to the honest parties $\{P_i\}_{i\in[n]\setminus\mathcal{C}}$. We assume that $\mathcal{S}_\Pi$ learns from the environment $\mathcal{Z}$ the list $\mathcal{C} \subset [n]$ of all corrupt parties from among $P_1, \ldots, P_n$, along with their inputs $\{x_i\}_{i\in\mathcal{C}}$.

Pre-TP Call Phase: In this phase, $\mathcal{S}_\Pi$ does the following:

- $\mathcal{S}_\Pi$ generates a common randomness string $r\|r'$ and provides $r\|r'$ to $\mathcal{A}$.
- $\mathcal{S}_\Pi$ generates a random default input x'_i for each (honest/corrupt) party P_i (this is done on behalf of the honest TP).
- $\mathcal{S}_\Pi$ sets $\mathsf{LFE.crs} := r$.
- $\mathcal{S}_\Pi$ derive n random pads $\{r_j\}_{j\in[n]}$, where $|r_j| = |x_j|$ from the common randomness sub-string r'.
- $\mathcal{S}_\Pi$ computes $(\mathsf{digest}^g, r^g) \leftarrow \mathsf{LFE.Compress}\Big(\mathsf{LFE.crs}, g\Big)$, where the function g is as in the "real scheme" (hard-coded with the random pad r_i for each party P_i).

TP Call Phase: In this phase, $\mathcal{S}_\Pi$ acts on behalf of the (honest) TP, and interacts with the (real-world) adversary $\mathcal{A}$ (which it internally simulates), and the simulator $\mathcal{S}_{\mathsf{LFE}}$ for the LFE scheme as follows:

- $\mathcal{S}_\Pi$ receives from the adversary $\mathcal{A}$ the message $\mathsf{in}_i = (\mathsf{LFE.crs}_i, \mathsf{digest}_i, z_i)$ corresponding to each corrupt party P_i.
- $\mathcal{S}_\Pi$ then partitions the corrupt parties in $\mathcal{C}$, such that all corrupt parties P_i in a given partition share the same common reference string and the same digest. It then chooses the partition $\mathcal{C}^*$ for which the common reference string $\mathsf{LFE.crs}^*$ and the digest digest^* match the common reference string $\mathsf{LFE.crs}$ and the digest digest that it sampled during phase-1.
- $\mathcal{S}_\Pi$ invokes the trusted party/ideal functionality f_Π with the corrupt set $\mathcal{C}$ and the corresponding inputs as:

$$\{\widehat{x}_i\}_{i\in\mathcal{C}} = \{x_i\}_{i\in\mathcal{C}^*} \cup \{x'_i\}_{i\in\mathcal{C}\setminus\mathcal{C}^*},$$

i.e., $\mathcal{S}_\Pi$ substitutes the default masked input for each corrupt party P_i that does not use the same $\mathsf{LFE.crs}$ and digest as sampled by $\mathcal{S}_\Pi$ in the pre-TP call phase.

- $\mathcal{S}_\Pi$ receives the output $\widetilde{y}$ from the ideal functionality. It then invokes the simulator $\mathcal{S}_{\mathsf{LFE}}$ for the LFE scheme using the tuple

$$(\mathsf{LFE.crs}, \mathsf{digest}, g, \widetilde{y}),$$

  and receives the simulated LFE ciphertext $\widetilde{\mathsf{ct}}$, which it subsequently forwards to the adversary $\mathcal{A}$.

- Eventually, the adversary $\mathcal{A}$ outputs some string $s_{\mathcal{A}}$, which $\mathcal{S}_\Pi$ forwards to the environment $\mathcal{Z}$ to complete the simulation.

We argue that the view of the adversary $\mathcal{A}$ corrupting the TP is computationally indistinguishable in the real and ideal worlds assuming that the LFE scheme satisfies simulation-security. To see this, observe that from the point of view of the adversary $\mathcal{A}$, the only difference between the real and ideal worlds is in the manner in which the LFE ciphertext ct is generated. In particular, if ct is the ciphertext that the adversary $\mathcal{A}$ receives in a uniformly random instance of the real-world experiment, and $\widetilde{\mathsf{ct}}$ is the ciphertext that the adversary $\mathcal{A}$ receives in a uniformly random instance of the ideal-world experiment, then we claim that the distributions of ct and $\widetilde{\mathsf{ct}}$ are computationally indistinguishable. Hence, we claim by extension that for any circuit family $\mathcal{Z} = \{z_\kappa\}$ corrupting any set $\mathcal{C} \subset [n]$ of the parties $P_1, \ldots, P_n$ maliciously, we have

$$\left\{\mathsf{real}_{\Pi,\mathcal{C},\mathcal{A}(\mathsf{aux})}(\vec{x},\kappa)\right\}_{\vec{x}\in(\{0,1\}^*)^n,\kappa\in\mathbb{N}} \approx_c \left\{\mathsf{ideal}^{\mathsf{type}}_{f,\mathcal{C},\mathcal{S}(\mathsf{aux})}(\vec{x},\kappa)\right\}_{\vec{x}\in(\{0,1\}^*)^n,\kappa\in\mathbb{N}},$$

whenever the LFE scheme satisfies simulation-security. To see this, suppose this is not the case, i.e., suppose there exists some $(\mathcal{Z},\mathcal{A})$ pair such that their views in the real and ideal world experiments are distinguishable with non-negligible advantage ϵ. We construct an algorithm $\mathcal{B}$ that can break simulation-security of the LFE scheme with non-negligible advantage $\epsilon' = \epsilon$.

The algorithm $\mathcal{B}$ samples a function f, receives from the environment $\mathcal{Z}$ the inputs for the honest parties among $P_1, \ldots, P_n$, and proceeds as follows:

- $\mathcal{B}$ receives from the challenger in the LFE simulation-security security game a common reference string $\mathsf{LFE.crs} = r$ and uses the same random string r in the simulation.

- $\mathcal{B}$ samples a random string r', provides the common randomness $r\|r'$ (where $r = \mathsf{LFE.crs}$) to the adversary $\mathcal{A}$, uses this randomness r' to derive the random pads $\{r_j\}_{j\in[n]}$ as in the "real" scheme, and sets $z_j = x_j \oplus r_j$ for each $j \in [n]$.

- $\mathcal{B}$ issues a digest-query to the challenger in the LFE simulation-security game with the function g, where g is as in the "real scheme" (hard-coded with the random pad r_i for each party P_i), and receives from the challenger the digest digest^g.

- $\mathcal{B}$ receives from the adversary $\mathcal{A}$ the message $\mathsf{in}_i = (\mathsf{LFE.crs}_i, \mathsf{digest}_i, z_i)$ corresponding to each corrupt party P_i, and partitions the corrupt parties in $\mathcal{C}$, such that all corrupt parties P_i in a given partition share the same common reference string and the same digest. It then chooses the partition $\mathcal{C}^*$ for which the common reference string $\mathsf{LFE.crs}^*$ and the digest digest^* match the common reference string $\mathsf{LFE.crs}$ and the digest digest that it sampled.

- $\mathcal{B}$ then issues a challenge ciphertext query to the challenger in the LFE simulation-security on the set of masked inputs

  $$(\{\widehat{z}_i\}_{i\in[n]}) = (\{z_i\}_{i\notin\mathcal{C}} \cup \{z_i\}_{i\in\mathcal{C}^*} \cup \{z'_i\}_{i\in\mathcal{C}\setminus\mathcal{C}^*}),$$

  i.e., $\mathcal{B}$ substitutes a (randomly sampled) default masked input for each corrupt party P_i that does not use the same $\mathsf{LFE.crs}$ and digest as sampled by $\mathcal{B}$ in the pre-TP call phase.

- The challenger responds with a challenge ciphertext ct^*. $\mathcal{B}$ forwards this ciphertext ct^* to the adversary $\mathcal{A}$.

- Eventually, the adversary $\mathcal{A}$ outputs some string $s_{\mathcal{A}}$, which $\mathcal{B}$ forwards to the environment $\mathcal{Z}$ to complete the simulation.

- Finally, the environment $\mathcal{Z}$ outputs a bit b. $\mathcal{B}$ outputs the same bit b.

It is easy to see that:

- When ct^* is sampled by the challenger from the "real world" distribution, i.e., $\mathsf{ct}^* = \mathsf{ct}$, the view of the adversary $\mathcal{A}$ (and hence the environment $\mathcal{Z}$) is exactly as in the real-world experiment.

- When ct^* is sampled by the challenger from the "ideal world" distribution, i.e., $\mathsf{ct}^* = \widetilde{\mathsf{ct}}$, the view of the adversary $\mathcal{A}$ (and hence the environment $\mathcal{Z}$) is exactly as in the ideal-world experiment.

Hence, if $\mathcal{A}$ (and hence $\mathcal{Z}$) has non-negligible advantage ϵ in distinguishing the real and ideal-world experiments, then $\mathcal{B}$ has non-negligible advantage $\epsilon' = \epsilon$ in breaking the simulation-security of the LFE scheme. This yields a contradiction, as desired.

B Proof of Theorem 17 (Full Security of Single-Key Succinct FE-based TP-aided MPC)

In this section, we prove Theorem 17 on the full security of our succinct single-key FE-based TP-Aided MPC protocol. As per the formal definition of full security for TP-aided MPC, our proof is divided into two cases for two distinct corruption scenarios:

- Corruption scenario-1: a semi-honest adversary $\mathcal{A}$ corrupts the TP, while the parties $P_1, \ldots, P_n$ are honest.

- Corruption scenario-2: a malicious adversary $\mathcal{A}$ corrupts a subset $\mathcal{C} \subset [n]$ of the parties $P_1, \ldots, P_n$ such that $|\mathcal{C}| \leq (n-1)$, while the TP is honest.

We tackle each of these cases separately below.

B.1 Case-1: Semi-Honest Corruption of TP

We first consider the case where the TP is corrupted by a semi-honest adversary $\mathcal{A}$, while the parties $P_1, \ldots, P_n$ are honest. In this setting, it is straightforward to construct a simulator $\mathcal{S}_\Pi$ that simulates the view of the adversary $\mathcal{A}$ without the knowledge of the inputs $x_1, \ldots, x_n$ for the honest parties.

Recall that in our protocol, the TP does not have any inputs nor does it receive any outputs at the end of the protocol; consequently, it suffices for $\mathcal{S}_\Pi$ to simulate the messages from the (honest) parties $P_1, \ldots, P_n$ to the semi-honest adversary $\mathcal{A}$ corrupting the TP. To this end, $\mathcal{S}_\Pi$ acts on behalf of the honest parties as follows:

- $\mathcal{S}_\Pi$ generates a uniform secret key $\mathsf{k}_i \leftarrow \mathsf{SKE.Gen}(1^\kappa)$ for every party P_i.
- $\mathcal{S}_\Pi$ generates $(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{FE.Setup}(1^\kappa)$.
- $\mathcal{S}_\Pi$ generates a uniformly random signing key-verification key pair $(\mathsf{sk}, \mathsf{vk})$ for the digital signature scheme.
- On behalf of each party P_i, $\mathcal{S}_\Pi$ generates the signature $\sigma_i = \mathsf{Sign}(\mathsf{sk}, (i, \mathsf{mpk}, \mathsf{k}_i))$.

On behalf of each (honest) party P_i, $\mathcal{S}_\Pi$ sends across to the semi-honest adversary $\mathcal{A}$ corrupting the TP the message

$$\mathsf{in}_i = (\mathsf{vk}, \mathsf{mpk}, \mathsf{k}_i, \sigma_i).$$

It is easy to see that the view of the semi-honest adversary $\mathcal{A}$ corrupting the TP is identical in the real and ideal worlds. Consequently, we have that for any circuit family $\mathcal{Z} = \{z_\kappa\}$ corrupting the TP P^* semi-honestly, we have

$$\left\{\mathsf{real}_{\Pi,\mathcal{C},\mathcal{A}(\mathsf{aux})}(\vec{x}, \kappa)\right\}_{\vec{x}\in(\{0,1\}^*)^n,\kappa\in\mathbb{N}} \equiv \left\{\mathsf{ideal}^{\mathsf{type}}_{f,\mathcal{C},\mathcal{S}(\mathsf{aux})}(\vec{x}, \kappa)\right\}_{\vec{x}\in(\{0,1\}^*)^n,\kappa\in\mathbb{N}},$$

B.2 Case-2: Malicious Corruption of Majority of Parties

We now consider the case where a malicious adversary $\mathcal{A}$ corrupts a subset $\mathcal{C} \subset [n]$ of the parties $P_1, \ldots, P_n$ such that $|\mathcal{C}| \leq (n-1)$, while the TP is honest. The simulation strategy in this case is substantially more involved.

We again construct an ideal-world simulator $\mathcal{S}_\Pi$, where $\mathcal{S}_\Pi$ now interacts with two additional ideal-world simulators - a simulator $\mathcal{S}_{\Pi_{\mathsf{idua}}}$ for the MPC protocol with identifiable abort, and a simulator $\mathcal{S}_{\mathsf{FE}}$ for the FE scheme. We divide the simulation strategy for $\mathcal{S}_\Pi$ into two phases

- the pre-TP call phase and the TP-call phase. Note that since the TP is honest, $\mathcal{S}_\Pi$ also acts on behalf of the TP in addition to the honest parties $\{P_i\}_{i\in[n]\setminus\mathcal{C}}$. We assume that $\mathcal{S}_\Pi$ learns from the environment $\mathcal{Z}$ the list $\mathcal{C} \subset [n]$ of all corrupt parties from among $P_1, \ldots, P_n$, along with their inputs $\{x_i\}_{i\in\mathcal{C}}$.

Pre-TP Call Phase: In this phase, $\mathcal{S}_\Pi$ acts on behalf of the honest parties among $P_1, \ldots, P_n$ (the TP is not yet involved), and interacts with the (real-world) adversary $\mathcal{A}$ (which it internally simulates) and the simulator $\mathcal{S}_{\Pi_{\mathsf{idua}}}$ for Π_{idua} (the MPC with id-abort protocol) as follows:

- $\mathcal{S}_\Pi$ receives from $\mathcal{S}_{\Pi_{\mathsf{idua}}}$ the messages corresponding to the honest parties among $P_1, \ldots, P_n$ during the MPC with abort protocol Π_{idua} and forwards these messages to the adversary $\mathcal{A}$.
- $\mathcal{S}_\Pi$ receives from $\mathcal{A}$ the messages corresponding to the corrupt parties among $P_1, \ldots, P_n$ during the MPC with abort protocol Π_{idua} and forwards these messages to $\mathcal{S}_{\Pi_{\mathsf{idua}}}$.
- At the end of the execution of the protocol, $\mathcal{S}_{\Pi_{\mathsf{idua}}}$ invokes the trusted party/ideal functionality $f_{\Pi_{\mathsf{idua}}}$ for the MPC with abort protocol Π_{idua}. At this point, it expects from $\mathcal{S}_\Pi$ the output of Π_{idua} for each party P_i such that $i \in \mathcal{C}$. $\mathcal{S}_\Pi$ responds as follows:
  - $\mathcal{S}_\Pi$ generates a default input x'_i for every (honest/corrupt) party P_i.
  - $\mathcal{S}_\Pi$ samples a secret key $\mathsf{k}_i \leftarrow \mathsf{SKE.Gen}(1^\kappa)$ for every party P_i.
  - For each corrupt party P_i, $\mathcal{S}_\Pi$ generates $e_i \leftarrow \mathsf{SKE.Enc}(\mathsf{k}_i, x_i)$.
  - For each honest party P_i, $\mathcal{S}_\Pi$ generates $\widetilde{e}_i \leftarrow \mathsf{SKE.Enc}(\mathsf{k}_i, 0^{|x_i|})$.
  - $\mathcal{S}_\Pi$ samples $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{FE.Setup}$.
  - $\mathcal{S}_\Pi$ generates $\mathsf{sk}_{\widetilde{g}} \leftarrow \mathsf{FE.KeyGen}(\mathsf{msk}, \widetilde{g})$, where the function $\widetilde{g}$ is identical to the function g in the "real" scheme *except* that it embeds $\widetilde{e}_i$ *instead of* e_i for each honest party P_i.
  - $\mathcal{S}_\Pi$ generates $(\mathsf{sk}, \mathsf{vk})$ for the digital signature scheme.

  Finally, $\mathcal{S}_\Pi$ provides to $\mathcal{S}_{\Pi_{\mathsf{idua}}}$ the following tuple for each corrupt party P_i:

  $$(\mathsf{vk}, \mathsf{mpk}, \mathsf{k}_i, \sigma_i, \mathsf{sk}_{\widetilde{g}}),$$

  where $\sigma_i = \mathsf{Sign}(\mathsf{sk}, (i, \mathsf{mpk}, \mathsf{k}_i))$.
- Eventually, $\mathcal{S}_{\Pi_{\mathsf{idua}}}$ outputs either `abort` or `continue`. If $\mathcal{S}_{\Pi_{\mathsf{idua}}}$ outputs `abort`, then $\mathcal{S}_\Pi$ also aborts the current execution and restarts the pre-TP call phase. If $\mathcal{S}_{\Pi_{\mathsf{idua}}}$ outputs `continue`, then $\mathcal{S}_\Pi$ proceeds to the next phase, as described below.

TP Call Phase: In this phase, $\mathcal{S}_\Pi$ acts on behalf of the (honest) TP, and interacts with the (real-world) adversary $\mathcal{A}$ (which it internally simulates), and the simulator $\mathcal{S}_{\mathsf{FE}}$ for the FE scheme as follows:

- $\mathcal{S}_\Pi$ receives from the adversary $\mathcal{A}$ the message $\mathsf{in}_i = (\mathsf{vk}_i, \mathsf{mpk}_i, \mathsf{k}_i, \sigma_i)$ corresponding to each corrupt party P_i. If there exists a corrupt party P_j such that $\mathsf{vk}_j = \mathsf{vk}$, $\mathsf{mpk}_j \neq \mathsf{mpk}$, and the signature σ_j verifies under vk, $\mathcal{S}_\Pi$ aborts the simulation.
- $\mathcal{S}_\Pi$ then partitions the corrupt parties in $\mathcal{C}$, such that all corrupt parties P_i in a given partition share the same verification key and master public key. It then chooses the partition $\mathcal{C}^*$ for which the verification key vk^* and the master public key mpk^* match the verification key vk and the master public key mpk that it used during phase-1.
- $\mathcal{S}_\Pi$ invokes the trusted party/ideal functionality f_Π with the corrupt set $\mathcal{C}$ and the corresponding inputs as:

$$\{\widehat{x_i}\}_{i\in\mathcal{C}} = \{x_i\}_{i\in\mathcal{C}^*} \cup \{x'_i\}_{i\in\mathcal{C}\setminus\mathcal{C}^*},$$

  i.e., $\mathcal{S}_\Pi$ substitutes the default input for each corrupt party P_i that does not use the same vk and mpk as chosen by $\mathcal{S}_\Pi$ during the pre-TP call phase.
- $\mathcal{S}_\Pi$ receives the output $\widetilde{y}$ from the ideal functionality. It then invokes the simulator $\mathcal{S}_{\mathsf{FE}}$ for the FE scheme using the tuple

$$(\mathsf{mpk}, \mathsf{sk}_{\widetilde{g}}, \widetilde{g}, \widetilde{y}, 1^{n|x_i|}),$$

  and receives the simulated FE ciphertext $\widetilde{ct}$, which it forwards to the adversary $\mathcal{A}$.
- Eventually, the adversary $\mathcal{A}$ outputs some string $s_\mathcal{A}$, which $\mathcal{S}_\Pi$ forwards to the environment $\mathcal{Z}$ to complete the simulation.

We argue that the view of the adversary $\mathcal{A}$ corrupting the TP is computationally indistinguishable in the real and ideal worlds. More formally, we claim (by extension) that for any circuit family $\mathcal{Z} = \{z_\kappa\}$ corrupting any set $\mathcal{C} \subset [n]$ of the parties $P_1, \ldots, P_n$ maliciously, we have

$$\left\{\mathsf{real}_{\Pi,\mathcal{C},\mathcal{A}(\mathsf{aux})}(\vec{x},\kappa)\right\}_{\vec{x}\in(\{0,1\}^*)^n,\kappa\in\mathbb{N}} \approx_c \left\{\mathsf{ideal}^{\mathsf{type}}_{f,\mathcal{C},\mathcal{S}(\mathsf{aux})}(\vec{x},\kappa)\right\}_{\vec{x}\in(\{0,1\}^*)^n,\kappa\in\mathbb{N}},$$

assuming that:

- The MPC protocol Π_{idua} satisfies simulation-security with id-abort.
- The succinct FE scheme satisfies (single-key) simulation-security.
- The symmetric-key encryption scheme satisfies IND-CPA security.
- The digital signature scheme satisfies existential unforgeability against chosen-message attacks.

We now establish this indistinguishability via a hybrid argument.

- **Hybrid-0:** This hybrid is identical to the real security experiment.
- **Hybrid-1:** This hybrid is identical to Hybrid-1 except that in the pre-TP call phase, we switch the execution of the MPC with identifiable abort protocol Π_{idua} from the real world to the ideal world. It immediately follows that Hybrid-1 is computationally indistinguishable from Hybrid-0 assuming that the Π_{idua} satisfies simulation-security with id-abort.

- **Hybrid-2:** In this hybrid, in the TP call phase of the simulation, the simulator $\mathcal{S}_\Pi$ aborts if it receives from the adversary $\mathcal{A}$ a message in_j on behalf of some corrupt party P_j such that $\mathsf{vk}_j = \mathsf{vk}$, $\mathsf{mpk}_j \neq \mathsf{mpk}$ and the signature σ_j verifies under vk, where vk and mpk are the signature verification key and the FE master public key sampled by $\mathcal{S}_\Pi$ in the pre TP call phase.

  We argue that the probability that $\mathcal{S}_\Pi$ aborts is negligible assuming that the digital signature scheme satisfies existential unforgeability against chosen-message attacks. To see this, observe that the view of the adversary $\mathcal{A}$ has no information about the secret signing key sk for the signature scheme used by the simulator $\mathcal{S}_\Pi$. Hence, any adversary $\mathcal{A}$ that manages to a create a message in_j on behalf of some corrupt party P_j such that $\mathsf{vk}_j = \mathsf{vk}$, $\mathsf{mpk}_j \neq \mathsf{mpk}$, and the signature σ_j verifies under vk with non-negligible probability, can be used to construct a PPT simulator $\mathcal{B}$ that successfully forges a signature on the chosen message mpk_j in an existential unforgeability game against the signature scheme with non-negligible probability (note that $\mathcal{B}$ can simulate the view of the adversary $\mathcal{A}$ using only the challenge verification key vk and the responses to its signing oracle queries in the existential unforgeability game, and without any information about the challenge signing key sk). This leads to a contradiction, as desired.

- **Hybrid-3:** In this hybrid, $\mathcal{S}_\Pi$ generates a simulated FE ciphertext $\widetilde{\mathsf{ct}}$ as in the ideal-world experiment, instead of generating the FE ciphertext ct as in the real-world experiment. We argue that the view of the adversary $\mathcal{A}$ in Hybrid-2 and Hybrid-3 are computationally indistinguishable assuming that the FE scheme satisfies (single-key) simulation security.

  To see this, suppose this is not the case, i.e., suppose there exists some $(\mathcal{Z}, \mathcal{A})$ pair such that their views in the real and ideal world experiments are distinguishable with non-negligible advantage ϵ. We construct an algorithm $\mathcal{B}$ that can break (single-key) simulation-security of the FE scheme with non-negligible advantage $\epsilon' = \epsilon$. The algorithm $\mathcal{B}$ samples a function f, receives from $\mathcal{Z}$ the inputs for the honest parties among $P_1, \ldots, P_n$, and simulates $\mathcal{S}_\Pi$ exactly as in Hybrid-2 except the following:

  – In the pre-TP call phase, instead of generating the FE master public key mpk by itself, $\mathcal{B}$ receives from the challenger in the FE simulation-security security game a master public key mpk, and uses the same mpk in the simulation.

  – In the pre-TP call phase, instead of generating the FE evaluation key sk_g by itself, $\mathcal{B}$ issues a single-key query to the challenger in the FE simulation-security game with the function g, where g is as in the "real scheme", receives from the challenger the corresponding key sk_g, and uses the same sk_g in the simulation.

  – In the post-TP call phase, instead of generating the FE ciphertext ct on behalf of the (honest) TP, $\mathcal{B}$ issues a challenge ciphertext query to the challenger in the FE simulation-security on the input

  $$(\{(k_i, b_i)\}_{i\in[n]}) = (\{(k_i, 1)\}_{i\notin\mathcal{C}} \cup \{(k_i, 1)\}_{i\in\mathcal{C}^*} \cup \{(\bot, 0)\}_{i\in\mathcal{C}\setminus\mathcal{C}^*}),$$

  i.e., $\mathcal{B}$ substitutes the default key-bit pair for each corrupt party P_i that does not use the same vk and mpk as used by $\mathcal{B}$ in the pre-TP call phase.

- The challenger responds with a challenge ciphertext ct^*. $\mathcal{B}$ forwards this ciphertext ct^* to the adversary $\mathcal{A}$.
- Eventually, the adversary $\mathcal{A}$ outputs some string $s_{\mathcal{A}}$, which $\mathcal{B}$ forwards to the environment $\mathcal{Z}$ to complete the simulation.
- Finally, the environment $\mathcal{Z}$ outputs a bit b. $\mathcal{B}$ outputs the same bit b.

It is easy to see that:

- When ct^* is sampled by the challenger from the "real world" distribution, i.e., $\mathsf{ct}^* = \mathsf{ct}$, the view of the adversary $\mathcal{A}$ (and hence the environment $\mathcal{Z}$) is exactly as Hybrid-2.
- When ct^* is sampled by the challenger from the "ideal world" distribution, i.e., $\mathsf{ct}^* = \widetilde{\mathsf{ct}}$, the view of the adversary $\mathcal{A}$ (and hence the environment $\mathcal{Z}$) is exactly as in Hybrid-3.

Hence, if $\mathcal{A}$ (and hence $\mathcal{Z}$) has non-negligible advantage ϵ in distinguishing Hybrids 2 and 3, then $\mathcal{B}$ has non-negligible advantage $\epsilon' = \epsilon$ in breaking the (single-key) simulation-security of the FE scheme. This yields a contradiction, as desired.

- **Hybrid-4:** This hybrid is identical to Hybrid-3 except that for each honest party P_i, $\mathcal{S}_\Pi$ generates $\widetilde{e}_i \leftarrow \mathsf{SKE.Enc}(\mathsf{k}_i, 0^{|x_i|})$.

  We argue that Hybrid-4 is indistinguishable from Hybrid-1 assuming that the symmetric-key encryption scheme satisfies IND-CPA security. The argument follows via a sequence of sub-hybrids $\{\text{Hybrid}_{3,j}\}_{j\in[0,n-|\mathcal{C}|]}$, where $\text{Hybrid}_{3,0}$ is identical to Hybrid-3, and in each sub-hybrid $\text{Hybrid}_{3,j}$ for $j \geq 1$, we switch e_{i_j} to $\widetilde{e}_{i_j}$ for the j-th honest party P_{i_j}. It is easy to see that the final sub-hybrid is identical to Hybrid-4.

  We now argue that for each $j \in [n - |\mathcal{C}| - 1]$, sub-hybrid $\text{Hybrid}_{3,j}$ is indistinguishable from sub-hybrid $\text{Hybrid}_{3,j+1}$. This follows immediately from the IND-CPA security of the symmetric-key encryption scheme. To see this, observe that the view of the adversary $\mathcal{A}$ has no information about the secret key $\mathsf{sk}_{i_{j+1}}$. Hence, any adversary $\mathcal{A}$ that can distinguish between $\text{Hybrid}_{0,j}$ and $\text{Hybrid}_{0,j+1}$ with non-negligible advantage ϵ can be used in a straightforward manner to construct a PPT algorithm $\mathcal{B}$ that distinguishes e_{i_j} and $\widetilde{e}_{i_j}$ in an IND-CPA security game with non-negligible advantage ϵ (note that $\mathcal{B}$ can simulate the view of the adversary $\mathcal{A}$ using its own challenge ciphertext, and without any information about the secret key sk_{i_j}). This leads to a contradiction, as desired.

C Extension of Impossibility in the Correlated Randomness Model

In this section, we extend our impossibility result for fully-secure MPC using a single call to a small TP (in the non-colluding model) from the special sub-class of protocols where the parties do not engage in any further communication post TP-call, to the more general sub-class of protocols where the parties are allowed to engage in any polynomially many rounds of communication post TP-call. Our extension relies on the following theorem:

Lemma 23 *Suppose there exists a fully secure MPC protocol in the non-colluding model (see Definition 7), where the parties have access to arbitrary correlated randomness, a single call to a TP of size* $\texttt{poly}(n, \kappa)$*, and are allowed to engage in* $r = \texttt{poly}(\kappa)$ *rounds of interaction post-TP call. Then, there exists (with probability at least* $1 - \texttt{negl}(\kappa)$*, where the probability space is over the internal randomness of the parties and the TP in the MPC protocol) a fully secure MPC protocol in the non-colluding model, where the parties have access to arbitrary correlated randomness, a single call to a TP of size* $\texttt{poly}(n, \kappa)$ *and are allowed only non-interactive computation after the TP call.*

It is easy to see that Lemma 23, together with Theorem 19, extends our negative result, as desired.

Proof of Lemma 23. To prove Lemma 23, we use a sequence of hybrid arguments as follows. Suppose that there exists a fully secure MPC protocol in the non-colluding model (see Definition 7), where the parties have access to arbitrary correlated randomness, a single call to a TP of size $\texttt{poly}(n, \kappa)$, and are allowed to engage in $r = \texttt{poly}(\kappa)$ rounds of interaction post-TP call.

Hybrid-r. Consider a malicious adversary $\mathcal{A}$ that maliciously corrupts a set $\mathcal{C}$ of t parties from among $P_1, \ldots, P_n$ for some $t > n/2$, and uses the following strategy: $\mathcal{A}$ instructs all maliciously corrupted parties to not send their round-r messages to the honest parties. We observe the following:

- Since the honest parties send out their round r-messages as dictated by the protocol, the adversary $\mathcal{A}$ manages to compute the output of the functionality as in an honest execution of the protocol.
- Since the protocol is, by definition, fully secure (i.e., has guaranteed output delivery) against malicious corruptions, all honest parties must also be able to compute the output without the messages from the corrupt parties with probability at least $1 - \texttt{negl}(\kappa)$. Here, we assume that the function computed by the protocol is such that the adversary cannot locally compute the output using its own inputs.

This implies that the views of the $(n - t)$ parties in $\overline{\mathcal{C}} = [n] \setminus \mathcal{C}$ at the end of round-$(r - 1)$ are sufficient to compute the output with probability at least $1 - \texttt{negl}(\kappa)$.

Hybrid-$(r-1)$. Now consider an adversary $\mathcal{A}$ that maliciously corrupts the set $\overline{\mathcal{C}} = [n] \setminus \mathcal{C}$ of $(n - t)$ parties that were honest in the previous hybrid, and instructs them to cease sending any messages during and after round-$(r - 1)$. We observe the following:

- By the argument in the previous hybrid, the adversary $\mathcal{A}$ manages to compute the output of the functionality using the view of the $n-t$ parties as claimed in the **Hybrid-**r.
- Again, since the protocol is, by definition, fully secure (i.e., has guaranteed output delivery) against malicious corruptions, all of the t remaining honest parties must also be able to compute the output without the corrupt parties' communication with probability at least $1-\texttt{negl}(\kappa)$.

This implies that the views of the t parties in $\mathcal{C}=[n]\setminus\overline{\mathcal{C}}$ at the end of round-$(r-2)$ are sufficient to compute the output with probability at least $1-\texttt{negl}(\kappa)$.

We build a sequence of r hybrids of the aforementioned forms down through **Hybrid-1**, where in each alternate hybrid, the adversary either corrupts the set $\mathcal{C}$ or the set $\overline{\mathcal{C}}$. Also, assume without loss of generality that r is even (the argument works similarly if r is odd). By extending the above chain of arguments, the views of t parties in $\mathcal{C}$ are enough to compute their outputs immediately after the call to the TP. In the **Hybrid-0**, the adversary corrupting $\mathcal{C}$ ceases to communicate in any of the post-TP rounds. Due to the full security guarantee, we claim that the remaining parties in $\overline{\mathcal{C}}$ must recover their output without the above messages.

Now recall that since we allow a corruption of upto $(n-1)$ parties, we can have $t=(n-1)$. This effectively implies that each party can compute its own output immediately after the TP call without engaging in any additional communication with any of the other parties. Let us assume that each party computes its own output immediately after the TP call with probability p for some $p\in(0,1)$.

Note that, each of the aforementioned hybrid arguments from **Hybrid-**r down through **Hybrid-0** holds true with probability $1-\texttt{negl}(\kappa)$, as opposed to probability 1. However, since $r=\texttt{poly}(\kappa)$, the accumulated error over all of the hybrid arguments remains $\texttt{negl}(\kappa)$; in other words, the overall sequence of argument remains correct with probability at least $1-\texttt{negl}(\kappa)$. So, we must have

$$p\geq 1-\texttt{negl}(\kappa).$$

This immediately yields, with overwhelmingly large probability, a fully secure MPC protocol in the non-colluding model, where the parties have access to arbitrary correlated randomness and a single call to a TP of size $\texttt{poly}(n,\kappa)$, and are only allowed a non-interactive computation after the TP call. This completes the proof of Lemma 23.

D TP-Aided Fair Coin-Tossing in the Colluding Model

In this section, we give evidence that any TP-aided fair protocol that computes coin-tossing securely in the colluding model is likely to involve total computation done by the TP which is as much as computing the function itself.

First, we show that any such fair protocol (that computes a special class of functions) in the colluding model can be assumed to have single call to the TP without loss of generality. For this, we present a compiler that transforms a fair protocol with $q(\kappa)$ invocations to a fair protocol with a single invocation.

Let $\pi^{q(\kappa)}$ denote the fair protocol with $q(\kappa)$ invocations. Suppose $\mathsf{in}_i^{(j)}$ denotes the message with which P_i invokes the TP during the jth call. Let $\mathsf{out}_i^{(j)}$ denote the response of jth invocation from the TP to P_i which is computed as $\mathsf{out}_i^{(j)} = \mathsf{TP}_i^j(\{\mathsf{in}_k^{(j)}\}_{k\in[n]}; r_{\mathsf{TP}}^{(j)})$ where $r_{\mathsf{TP}}^{(j)}$ denotes the randomness sampled by the TP for this invocation. [5] Let $\mathcal{T}$ denote the set of $q(\kappa)$ invocations in $\pi^{q(\kappa)}$.

We present the transformation from $\pi^{q(\kappa)}$ to π^1 (that involves single invocation to the TP) which involves the following sequence of steps:

(a) $\pi^{q(\kappa)} \rightarrow \pi^{q(\kappa)+1}$: $\pi^{q(\kappa)+1}$ is identical to $\pi^{q(\kappa)}$ except for the following change. There is an additional invocation in the beginning of the protocol which returns $\mathsf{out}_i^{(1)} = \{r_{\mathsf{TP}}^{(j)}\}_{j\in\mathcal{T}}$ for each $i \in [n]$.

(b) $\pi^{q(\kappa)+1} \rightarrow \pi_{\mathsf{bc}}^{q(\kappa)+1}$: $\pi_{\mathsf{bc}}^{q(\kappa)+1}$ is identical to $\pi^{q(\kappa)+1}$ except for the following change - Each P_i also broadcasts $\mathsf{in}_i^{(j)}$ in addition to sending it to the TP.

(c) $\pi_{\mathsf{bc}}^{q(\kappa)+1} \rightarrow \pi^1$: π^1 proceeds identical to $\pi_{\mathsf{bc}}^{q(\kappa)+1}$ except the following. Consider the j^{th} invocation to the TP where $j \geq 2$ (the first invocation to the TP remains the same). Then, instead of invoking the TP, each P_i carries out the following local computation $\mathsf{out}_i^{(j)} \leftarrow \mathsf{TP}_i^j(\{\mathsf{in}_k^{(j)}\}_{k\in[n]}; r_{\mathsf{TP}}^{(j-1)})$ where $\{\mathsf{in}_k^{(j)}\}_{k\in[n]}$ and $r_{\mathsf{TP}}^{(j-1)}$ is received via broadcast and $\mathsf{out}_i^{(1)}$ respectively. [6]

Next, we show that π^1 is fair based on our assumption that $\pi^{q(\kappa)}$ is fair by proving that fairness is preserved in each of the above steps **(a), (b)** and **(c)**.

Lemma 24 $\pi^{q(\kappa)+1}$ *achieves fairness.*

Proof: Security of $\pi^{q(\kappa)+1}$ follows directly from security of $\pi^{q(\kappa)}$ (assumed to be fair) as the view of the adversary is identical in both. This holds since the adversary can access $r_{\mathsf{TP}}^{(j)}$ for each $j \in \mathcal{T}$ since the beginning of $\pi^{q(\kappa)}$ as well. Also, note that the computation done during the first invocation of $\pi^{q(\kappa)+1}$ is still $\mathtt{poly}(\kappa)$ (as it is bounded by $q(\kappa)\cdot r(\kappa)$, where it is assumed that the computation in each call is bounded by $r(\kappa)$). $\square$

Lemma 25 $\pi_{\mathsf{bc}}^{q(\kappa)+1}$ *achieves fairness.*

Proof: We note that $\mathsf{in}_i^{(j)}$ corresponding to an honest P_i is known to the adversary in $\pi^{q(\kappa)+1}$ as well, since the adversary semi-honestly corrupts the TP. Therefore, the additional

[5]For deterministic computation by the TP, $r_{\mathsf{TP}}^{(j)}$ is simply $\perp$.
[6]Note that $r_{\mathsf{TP}}^{(j-1)}$ is used for jth call in $\pi_{\mathsf{bc}}^{q(\kappa)+1}$ as it corresponds to $(j-1)$th call in $\pi^{q(\kappa)}$.

messages broadcast by the honest parties during $\pi_{\mathsf{bc}}^{q(\kappa)+1}$ reveal no extra information to the adversary and its view remains identical across $\pi_{\mathsf{bc}}^{q(\kappa)+1}$ and $\pi^{q(\kappa)+1}$. Thus, the lemma holds due to security of $\pi^{q(\kappa)+1}$ (Lemma 24). $\square$

Lemma 26 *π^1 achieves fairness.*

Proof: Consider an execution of $\pi_{\mathsf{bc}}^{q(\kappa)+1}$ where each corrupt party P_k invokes the TP during the jth invocation using $\{\mathsf{in}_k^{(j)}\}$ that it broadcasts (for each $j \in [2, q(\kappa)+1]$). It follows from Lemma 25 that such an execution is fair. We note that $\mathsf{out}_i^{(j)}$ computed by an honest P_i in π^1 is identical to the response given by the TP during the j^{th} ($j \in [2, q(\kappa)+1]$) invocation of the above described execution of $\pi_{\mathsf{bc}}^{q(\kappa)+1}$. Thereby, security of π^1 follows from security of $\pi_{\mathsf{bc}}^{q(\kappa)+1}$. $\square$

We can thus conclude that any fair protocol in the colluding TP model can be assumed to have single call to the TP without loss of generality. Lastly, we point that the above compiler does not inflate the total computation done by the TP (because the TP computation done in π^1 only involves sampling the internal TP randomness used across the TP invocations, which was also done during $\pi^{q(\kappa)}$).

Next, we argue that any fair protocol that computes coin-tossing securely in the colluding model seems likely to involve total computation done by the TP which is as much as computing the function itself.

At a high-level, we use the above transformation to transform any generic fair coin-tossing protocol in colluding model to a fair protocol that utilizes only a single TP call in the beginning of the protocol (during the first round). We then design a series of adversarial strategies where the adversary aborts in Rounds 2 to r' sequentially (where r' denotes the total number of rounds) depending on the response of the first (and only) TP call. This allows us to show that once the TP invocation is completed, the distribution of the outputs does not change with further interaction amongst the parties. We elaborate on this below.

Consider a fully-secure 2-party protocol $\pi^{q(\kappa)}$, say comprising of $r(\kappa)$ rounds, that computes the function of coin tossing using $q(\kappa)$ invocations to the colluding TP. Following the transformation in the previous section, we can transform $\pi^{q(\kappa)}$ to π^1 (comprising of $r'(\kappa) = r(\kappa) + 1$ rounds) that has a single TP invocation in Round 1 and messages sent via broadcast channels thereafter. Note that the transformation is such that the total computation done by the TP during π^1 does not exceed the total computation done by the TP in $\pi^{q(\kappa)}$.

Next, assume for simplicity that the response of the first invocation is a single-bit $t \in \{0, 1\}$. Let $\Pr[t = 0] = p$ (probability over the random tape of TP). Let a_i ($i \in [r']$) denote the output of P_1 when P_2 acts honestly until (and including) Round i and then aborts. b_i is defined symmetrically. It follows from the correctness of the protocol that for $\mathsf{out} \in \{0, 1\}$, where out denotes the potential output of the coin-tossing protocol:

$$\Pr[a_{r'} = \mathsf{out} | t = 0] \cdot p + \Pr[a_{r'} = \mathsf{out} | t = 1] \cdot (1 - p) = \frac{1}{2}$$

We now consider a strategy $\mathcal{A}_1$ - Adversary corrupts P_2 who does the following: Participate honestly in Round 1. If $t = 0$, then stop communication Round 2 onwards. Else, participate honestly throughout the protocol. Since π^1 must maintain security even against $\mathcal{A}_1$, the following must hold for $\mathsf{out} \in \{0, 1\}$:

$$\Pr[P_1 \text{ outputs } \mathsf{out}] = \Pr[a_1 = \mathsf{out} | t = 0] \cdot p + \Pr[a_{r'} = \mathsf{out} | t = 1] \cdot (1 - p) = \frac{1}{2}$$

Comparing both the above equations, we get that $\Pr[a_{r'} = \mathsf{out} | t = 0] = \Pr[a_1 = \mathsf{out} | t = 0]$.

We capture the above in a generalized argument for $i \in [r']$. Consider strategy $\mathcal{A}_i$ where adversary corrupting P_2 does the following - Participate honestly until (and including) Round i. If $t = 0$, then abort thereafter. Else, participate honestly throughout the protocol. Since π^1 must maintain security even against $\mathcal{A}_i$, the following must hold for $\mathsf{out} \in \{0, 1\}$:

$$\Pr[P_1 \text{ outputs } \mathsf{out}] = \Pr[a_i = \mathsf{out} | t = 0] \cdot p + \Pr[a_{r'} = \mathsf{out} | t = 1] \cdot (1 - p) = \frac{1}{2}$$

Combining all the above, we get $\Pr[a_1 = \mathsf{out} | t = 0] = \Pr[a_2 = \mathsf{out} | t = 0] = \cdots = \Pr[a_{r'-1} = \mathsf{out} | t = 0] = \Pr[a_{r'} = \mathsf{out} | t = 0]$. Similarly, it can be shown that $\Pr[a_1 = \mathsf{out} | t = 1] = \Pr[a_2 = \mathsf{out} | t = 1] = \cdots = \Pr[a_{r'-1} = \mathsf{out} | t = 1] = \Pr[a_{r'} = \mathsf{out} | t = 1]$.

This implies that once the TP invocation is completed, the distribution of the outputs does not change. Intuitively, this seems to imply that the interaction post TP invocation does not help in output computation and gives evidence that it is likely that the computation done by the TP is as much as computing the function itself.

E Fully Secure MPC using Stateful TP

In this section, we study the problem of achieving fully secure MPC protocols in the *stateful* TP model. We sketch an MPC protocol that achieves full security against fail-stop corruption of a majority of parties while relying on Fully Homomorphic Encryption (FHE) [Gen09, BGV12] and a small stateful TP that is invoked twice during the protocol. Recall that FHE is a form of encryption that permits computations directly over encrypted data without decrypting it first, where the result of each such computation is also in the encrypted form.

Protocol Sketch. We now sketch an MPC protocol involving parties $P_1, \ldots, P_n$ that achieves full security against fail-stop corruption of a majority of parties. At a high level, the protocol proceeds as follows:

- **First TP-call:** Each party P_i sends its input x_i to the stateful TP. The TP collects all of the inputs (while substituting a default input for each party that does not send across its input), and encrypts each of these inputs under some FHE public key pk that it generates uniformly at random (along with the corresponding secret key sk). The TP then sends across to each party P_i the FHE public key pk and the FHE ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$ encrypting the original/default inputs for all parties under pk. The TP retains the FHE secret key sk as part of its internal state.

- **Local Computation:** Each party P_i locally computes a *homomorphic evaluation* of the function f over the FHE ciphertexts $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$ received from the TP, and obtains a ciphertext ct_i^*.
- **Second TP-call:** Each P_i then invokes the TP using its homomorphically evaluated ciphertext ct_i^*, and receives from the TP the corresponding decryption (recall that the TP retains sk as part of its internal state). The party P_i then outputs the decryption received from the TP. If a party does not invoke the TP during the second TP-call, the TP sends $\perp$ to this party.

Correctness of the protocol follows immediately from the correctness of the underlying FHE scheme. Note that this protocol requires the TP to maintain its internal state across the two calls. Also, it is only secure against fail-stop corruption of a majority of the parties, and does not provide security against a malicious corruption of parties. We leave it as an open question to construct, given only an FHE scheme, an MPC protocol that makes two calls to a small stateful TP and achieves full security against malicious corruption of majority of the parties. We also leave it as an open question to realize, given only an FHE scheme, a fully secure MPC protocol that achieves full security against fail-stop/malicious corruption of a majority of parties while making at most two calls to a *stateless* TP.