Secure Protocol Transformations

Yuval Ishai[1], Eyal Kushilevitz[1], Manoj Prabhakaran[2], Amit Sahai[3], and Ching-Hua Yu[2]

[1] Technion, Haifa, Israel, and University of California, Los Angeles. {yuvali,eyalk}@cs.technion.il
[2] University of Illinois, Urbana-Champaign. {mmp,cyu17}@cs.illinois.edu
[3] University of California, Los Angeles. sahai@cs.ucla.edu

Abstract. In the rich literature of secure multi-party computation (MPC), several important results rely on "protocol transformations," whereby protocols from one model of MPC are transformed to protocols from another model. Motivated by the goal of simplifying and unifying results in the area of MPC, we formalize a general notion of black-box protocol transformations that captures previous transformations from the literature as special cases, and present several new transformations. We motivate our study of protocol transformations by presenting the following applications.

- Simplifying feasibility results:
  - Easily rederive a result in Goldreich's book (2004), on MPC with full security in the presence of an honest majority, from an earlier result in the book, on MPC that offers "security with abort."
  - Rederive the classical result of Rabin and Ben-Or (1989) by applying a transformation to the simpler protocols of Ben-Or et al. or Chaum et al. (1988).
- Efficiency improvements:
  - The first "constant-rate" MPC protocol for a constant number of parties that offers full information-theoretic security with an optimal threshold, improving over the protocol of Rabin and Ben-Or;
  - A fully secure MPC protocol with optimal threshold that improves over a previous protocol of Ben-Sasson et al. (2012) in the case of "deep and narrow" computations;
  - A fully secure MPC protocol with near-optimal threshold that improves over a previous protocol of Damgård et al. (2010) by improving the dependence on the security parameter from linear to polylogarithmic;
  - An efficient new transformation from passive-secure two-party computation in the OT-hybrid and OLE-hybrid model to zero-knowledge proofs, improving over a recent similar transformation of Hazay and Venkitasubramaniam (2016) for the case of static zero-knowledge, which is restricted to the OT-hybrid model and requires a large number of commitments.

Finally, we prove the *impossibility* of two simple types of black-box protocol transformations, including an unconditional variant of a previous negative result of Rosulek (2012) that relied on the existence of one-way functions.

1 Introduction

Secure multi-party computation (MPC) is one of the central topics around which modern cryptography has been shaped. Research in MPC has led to major innovations in cryptography, including effective definitional approaches (e.g., simulation-based security [16,15]), powerful and vastly applicable algorithmic techniques (starting with secret-sharing [28] and garbling schemes [30]), sharp impossibility results (e.g., [8]) and even several cryptographic concepts ahead of their time (like fully-homomorphic encryption [29]). Significantly, in recent years, some of these results have started moving from theory to practice, spurring significant further theoretical and engineering effort to optimize their performance and usability.

Over 35 years of active research, MPC has grown into a rich and complex topic, with many incomparable flavors and numerous protocols and techniques. Indeed, just cataloguing the state of the art results is a non-trivial research project in itself, as exemplified by the recent work of Perry et al. [25], which proposes classifying the existing protocols using 22 dimensions.

This diversity of models and questions forms a wide spectrum of possible tradeoffs between functionality, security, and efficiency, which partially explains the massive amount of research in the area. But this diversity also poses the risk of misdirected research efforts. For instance, if a new technique is introduced in order to obtain an efficiency improvement in one model, it is not clear a-priori to which other models the same technique may apply; and even when the same technique directly applies to other models, one typically needs to manually modify protocols and their analysis to ensure it.

While developing and maintaining a systematic database like the one in [25] is certainly helpful, we propose a complementary approach to taming the complex landscape of MPC protocols. Our approach is to relate the various flavors of MPC problems to each other by means of general *protocol transformations*. More concretely, our work studies the following high level question:

> *To what extent can results in one MPC model be "automatically" transformed to other models?*

This question is motivated by the following goals.

- *Simplicity.* The current proofs of the main feasibility results in the area of MPC are quite involved, and results for different models share few common ingredients. We would like to obtain a simpler and more modular *joint* derivation of different feasibility results from the 1980s [31,15,2,6,26,24], which were originally proved using very different techniques.

– *Efficiency.* Despite a lot of progress on the efficiency of MPC, there are still significant gaps between the efficiency of the best known protocols in different models. For instance, viewing the number of parties n as a constant, n-party protocols that offer full-security (with guaranteed output delivery) against $t < n/2$ malicious parties [26,9] are asymptotically less efficient compared to similar protocols with security against $t < n/3$ parties [2], or even to protocols that offer "security with abort" against $t < n$ malicious parties [21].

A classical example of a general protocol transformation is the well known "GMW compiler," [15], which transforms any MPC protocol that offers security against passive corruptions into one that offers security against active corruptions, with the help of zero-knowledge proofs. Considering that this transformation has been behind several subsequent feasibility results, one may legitimately consider that *the GMW transformation is as important as – if not more important than – the GMW protocol itself is, as an object of study.* More recent examples include the IKOS transformation using "MPC-in-the-head" [19] and the IPS transformation that combines player-virtualization with "watchlists" [21]. Common to all these techniques is the idea that they generically transform any set of protocols that are secure for some ("easier") flavors of MPC into a protocol that is secure for another ("harder") flavor.

While these previous results demonstrate the plausibility of general MPC protocol transformations in some interesting cases, they are still far from covering the space of all desirable transformations between different MPC models and leave open several natural questions.

In this work, we initiate a systematic study of such MPC protocol transformations. We define a framework to formalize these transformations, and present a few positive and negative results. We are interested in obtaining conceptually simpler alternative proofs for known feasibility results by means of new transformations, as well as in obtaining new results. We now discuss the goals of this research in more detail.

The main theoretical motivation for studying protocol transformations is that they highlight the *essential new challenges* presented in a harder flavor of MPC compared to an easier flavor. For instance, the GMW-transformation distilled out verifying claims in zero-knowledge as the essential challenge in moving from semi-honest security to security against active corruption. As another example, in this work, we present a new transformation, that can recover the classical feasibility result of Rabin and Ben-Or [26] regarding security with guaranteed output delivery with an honest majority, from two simpler feasibility results (both of which were solved in [2,6]): (i) security against passive corruption with an honest majority and (ii) security with guaranteed output delivery

but only with an arbitrarily large fraction of honest parties. We identify achieving an intermediate security notion – security with partially identifiable abort – as the key challenge in this transformation.

As noted above, another important motivation behind studying protocol transformations is the possibility of *efficiency improvements*. On the face of it, protocol transformations are not ideal for obtaining *efficient* protocols, as one can hope to obtain extra efficiency by engineering fine details of the protocols as applicable to the specific flavor of MPC. While that may indeed be true, a protocol transformation can leverage advances in one flavor of MPC to obtain efficiency improvements in another flavor. As it turns out, this lets us obtain several *new asymptotic efficiency results* based on a single new transformation. Considering that efficiency of MPC is a well-studied area, obtaining several new result at once illustrates the power of such transformations.

There are other practical and theoretical motivations that led to this work, which we mention below.

– From a pragmatic point of view, understanding the connections across flavors of MPC will help in *modular implementations* of protocols. Indeed, the implementation of a transformation from one flavor to another would tend to be significantly simpler than an entire protocol in the latter flavor, specified and implemented from scratch.

– Roles of important techniques can often be *encapsulated as transformations* among appropriate intermediate security notions (e.g., "player elimination" can be encapsulated as implementing a transformation from "identifiable-abort-security" to full-security). In the absence of such abstraction, these techniques remain enmeshed within more complex protocols, and may not benefit from research focus that a transformation can attract.

– More generally, transformations are important in *reducing duplicated research effort*. For instance, if a new technique is introduced in order to obtain an efficiency improvement in one model, it is not clear *a priori* to which other models the same technique may apply; and even when the same technique directly applies to other models, one typically needs to manually modify protocols and their analysis to ensure it. On the other hand, if generic transformations are available across models, techniques can be easily adapted across models.

– Finally, a theoretical framework is necessary to understand the *limitations of protocol transformations*, via formal impossibility theorems. Indeed, without a rigorous notion of "black-box" transformations, it is not clear how to rule out the possibility of a "transformation" which simply discards the protocol it is given and builds one from scratch. This is especially the case for unconditional security, where the standard notions of black-box use of computational

assumptions are not helpful in differentiating a legitimate transformation from one which builds its own (unconditionally secure) protocol from scratch.

A Motivating Example. As an illustration of the use of protocol transformations in simplifying the landscape of MPC protocols, we consider two protocol schemes from Goldreich's book [14, Chapter 7]. The first one obtains (stand-alone) security-with-abort against arbitrary number of corruptions by an active, probabilistic polynomial time (PPT) adversary[4] (under standard cryptographic assumptions), for general function evaluation, in a model with broadcast channels only. The second one obtains full-security (i.e., guaranteed output delivery) in the same setting, but restricting the adversary to corrupt less than half the parties. Both these protocol schemes are obtained using the GMW transformation. However, *the latter feasibility result does not take advantage of the former*, but instead uses verifiable secret-sharing (VSS) and several other techniques to achieve full-security, while retaining certain elements from the previous construction.

We point out that in fact, one could avoid the duplicated effort by giving a protocol transformation from the former flavor to the latter flavor of MPC. For this, we abstract out a slightly stronger security guarantee provided by the first protocol: while it allows an adversary to abort the protocol after learning its own input, aborting always leads to identification of at least one party that is corrupted by the adversary. This notion of security is often referred to as security with identifiable-abort [20]. In Section 4.1, we show that one can easily transform such a protocol into a protocol with full-security.

Security Augmentation and Efficiency Leveraging. Typically, an MPC protocol transformation falls into one of two broad (informally defined) classes: *security augmentation* and *efficiency leveraging*. Security augmentation refers to building MPC protocols with strong security guarantees by transforming MPC protocols with weaker security guarantees. The IPS compiler [21] is an instance of security augmentation. Efficiency leveraging, on the other hand, aims to improve the efficiency of MPC protocols, without necessarily increasing their security guarantee. In such a transformation, the original (inefficient) protocol will typically be used on a "small" sub-computation task, in combination with other cheaper (but less secure) protocols applied to the original "large" computation task. The goal of the sub-computation task is usually to ensure that the strong attacks on the final protocol has the effect of weak attacks on an execution of the cheaper, less secure protocol. An instance of efficiency leveraging is given by Bracha's transformation [4], in which the strength of the security guarantee cor-

[4] One may consider static or adaptive corruption here. By default, we shall consider adaptive adversaries in all constructions in this paper.

responds to the corruption-threshold (i.e., what fraction of parties are corrupted) that can be tolerated.

1.1 Our Contributions

Framework. Firstly, we formalize the notion of a Black-Box Transformation (BBT) from protocol schemes satisfying some security (or efficiency) requirements to a protocol scheme satisfying some other requirements.[5] Towards this, we formalize notions like protocol schemes (which map functionalities to protocols) and security definitions (which are just sets of pairs of functionalities and protocols), all in a fairly abstract fashion. A BBT itself is modeled using a circuit that describes a protocol's structure as a program built from various components.

The framework is general enough to cast all of the above mentioned transformation (GMW, Bracha, IKOS and IPS) as instances of BBT.

We remark that we treat security notions highly abstractly, and do not impose any conditions on how security is proven. However, in all our positive results and examples, security definitions use a simulation paradigm, and one could define a "fully" blackbox transformation by requiring that the simulator of the protcol resulting from the transformation be constructed in a black-box manner from the simulators of the given protocols. For the sake of simplicity, and to keep the focus on the structure of the constructions rather than on the proofs of security, we do not formally include this restriction in our definition of BBT. We also point out that this strengthens our impossibility results.

New Transformations and Consequences. We present a new transformation which can be used to obtain known and new results about (information-theoretically) secure MPC for general function evaluation, with guaranteed output delivery, given an honest-majority and a broadcast channel. Our transformation yields such an MPC scheme starting from two protocol schemes – one achieving full-security, but for a lower threshold (βn corruption threshold, for some $\beta > 0$) and one achieving semi-honest security under honest-majority (Corollary 1). (See the next section for an overview of the transformation, and the various intermediate transformations that lead to it.) From this transformation we obtain the following results:

[5] The term "Black-Box" refers to the fact that (the next-message function of) the resulting protocol uses (the next-message function of) all the constituent protocols and the functionality itself as oracles; however, note that the constituent protocols themselves may depend on their functionalities in a non-black-box manner.

1. We readily obtain the result of Rabin and Ben-Or [26] as a consequence of the earlier work of Ben-Or et al. and Chaum et al. [2,6], via the above transformation.
2. We obtain the first "constant-rate" MPC protocol scheme with guaranteed output delivery against corruption of less than $n/2$ parties, provided the number of parties is constant (Corollary 2). That is, the total communication in this protocol is at most $c_n|C|$, where C is the circuit representation of the function, and c_n is a constant independent of the security parameter and C but dependent only on the number of parties. This result is obtained – following the lead of [21][6] – by applying our transformation to the scheme of [11] (combined with a secret-sharing scheme due to [7]) and the semi-honest secure scheme of [2].
3. Next, we present an *efficiency leveraging* transformation, which is designed to improve the efficiency of a protocol scheme with full-security, by combining it with a (cheaper) protocol which achieves security-with-abort (Theorem 8). By applying this transformation to the above protocol with full-security and an efficient protocol with security-with-abort from [13], we obtain a "scalable" MPC protocol with full-security and optimal corruption-threshold – i.e., tolerating corruption of less than $n/2$ parties (Corollary 3).[7] For an arguably natural class of functions (namely, sequential computations, where the size of a circuit implementing the function is comparable to its depth), this is the first scalable protocol with full-security and optimal threshold (complementing a result of [3], which obtains similar efficiency for circuits which are of relatively low depth).
4. We present an efficient new transformation from two-party protocols in the OT-hybrid or OLE-hybrid model that offer security against passive corruptions to zero-knowledge proofs in the commitment-hybrid model, improving over a recent similar transformation of Hazay and Venkitasubramaniam [17] for the case of static zero-knowledge. (We note that the IKOS transformation for protocols in such hybrid models requires at least 3 parties.) The transformation from [17] cannot be applied in the OLE-hybrid model, and when applied to natural protocols in the OT-hybrid model such as the GMW protocol, it requires several separate commitments for each gate in the circuit. Our transformation for the OLE-hybrid model can be applied towards efficient zero-knowledge proofs for *arithmetic* circuits and in both hybrids our transformation requires just a constant number of commitments overall (for

[6] In [21], these two protocol schemes were combined to obtain a similar constant-rate protocol, but in the oblivious-transfer (OT) hybrid model and with security-with-abort.

[7] Here the term "scalable" denotes that for evaluating large circuits C, the *communication complexity per party* scales as $\tilde{O}(|C|)$ (up to polylog multiplicative factors and polynomial additive terms of the security parameter and the number of parties).

a constant soundness error). This transformation may have relevance to the recent line of work on practical zero-knowledge proofs initiated in [23]. In contrast to [17], we do not consider here the goal of adaptive zero-knowledge in the plain model.

5. Our final application considers the problem of relaxing the corruption threshold from the optimal $n/2$ to $n(1/2 - \epsilon)$, for any constant $\epsilon > 0$. In this case, we obtain a *highly scalable protocol* in which the *total* communication for evaluating a circuit C is $\tilde{O}(|C|)$, ignoring additive terms that depend on the number of parties, but not the size of the circuit (Corollary 4). This improves over a result of [12].[8]

   For this, we apply Bracha's transformation [4] to one of the above protocols. Specifically, we use Bracha's transformation to combine an outer protocol that has a relatively low corruption threshold but is highly scalable with respect to communication and computation (in our case the one from [12]), and an inner protocol with optimal threshold (in our case, the one from item 2 above), to obtain a protocol with a near-optimal threshold.

Impossibility Results. One may ask if security against active corruption can solely be based on security against semi-honest adversaries. Such questions can be formalized as questions about the existence of a BBT. We present two impossibility results:

1. We consider the question of functionally-black-box protocol schemes, introduced by Rosulek [27]. (This is a special case of protocol transformations where no protocol scheme is provided to the transformation.) Rosulek demonstrated a two-party functionality family for which there is no functionally black-box protocol, *assuming the existence of one-way functions*. We present an unconditional version of this result (Theorem 1).
2. We show a functionality family – namely, zero-knowledge proof functionalities – for which there is no BBT from semi-honest security to security (with abort) against active adversaries (Theorem 2).

We remark that the proof of our second result breaks down if we expanded the family of functionalities from ZK functionalities to all efficient functionalities. We leave it as an important open problem to prove broader impossibility results for *general* computation (in which the family considered is the family of all functionalities).

[8] In [12], in the absence of broadcast channels, the near-optimal threshold of $n(\frac{1}{3} - \epsilon)$ was considered. We can extend our result to this setting by implementing broadcast channels among a constant number of parties, with a constant factor blow-up in communication.

1.2 Technical Overview

Black-Box Transformations. We make precise a notion of a black-box transformation among protocol schemes. Given a functionality f, a black-box transformation can define new functionalities (which are syntactically just programs) that access f in a black-box manner. Then, it can invoke a given protocol scheme on any such functionality, to obtain a protocol (which is, again, a program). The transformation can repeat these steps of defining new functionalities in terms of programs it already has, and of invoking given protocol schemes on such functionalities any number of times. At the end, it outputs one of the programs as its protocol.

We point out that the "protocol step" (invoking a protocol scheme on a functionality) is *not* limited to using the functionality as a black-box. However, it is a black-box step in the sense that the transformation can be instantiated with *any* protocol scheme with the requisite security guarantees.

Example: IPS Transformation. An example of a black-box transformation (that we shall build on later) is the IPS transformation [21]. We shall graphically represent a transformation using a circuit diagram like the one in Figure 1.

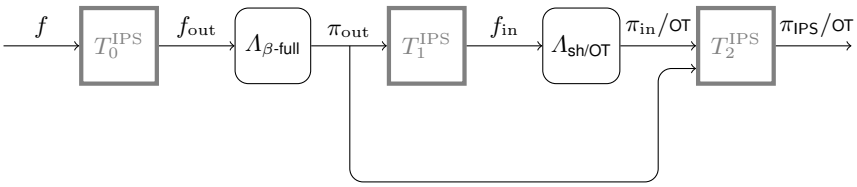

Fig. 1: Black-Box Transformation in the IPS compiler

Here, each rectangular node (labeled T_0^{IPS}, T_1^{IPS} and T_2^{IPS}) outputs a program which makes black-box access to one or more programs input to that node. T_0^{IPS} converts an n-party functionality f into a functionality f_{out} involving n "clients" and N "servers". T_1^{IPS} defines f_{in} to be an n-party functionality in which the trusted party carries out the program of a server in the protocol π_{out}. The bulk of the compiler is part of the transformation T_2^{IPS}, which combines the programs of two protocols π_{out} and π_{in} in a black-box way to define the final protocol.

The diagram also shows two other nodes, labeled $\Lambda_{\beta\text{-full}}$ and $\Lambda_{\mathsf{sh/OT}}$, each of which take as input a functionality (f_{out} and f_{in} resp.) and produces a protocol (π_{out} and π_{in} resp.). The labels on the nodes indicate the security guarantees required of these protocols (security against active corruption of strictly less than a $\beta > 0$ fraction of the parties, and security against semi-honest corruption, in the OT hybrid model resp.). [21] show that irrespective of what protocol

From	To	Theorem	Notes
id_α-security, $t < \alpha n$	full, $t < \alpha n$	Theorem 3, Theorem 4	Using player-elimination. Theorem 4 relies on a non-blackbox decomposition of the function, and yields efficiency close to the non-abort-case efficiency of the given protocol.
(sh-security, $t < \alpha n$) and (full-security, $t < \beta n$)	id_α, $t < \alpha n$	Theorem 5, Theorem 6	An honest-majority version of the IPS transformation. Any $\beta > 0$ suffices. Theorem 6 saves a factor of n using an expander graph-based watchlist scheme.
(sh-security, $t < \alpha n$) and (full-security, $t < \beta n$)	full, $t < \alpha n$	Corollary 1	Combining the above two.
(abort-secure π_1, $t < \alpha n$) and (id_α-secure π_2, $t < \alpha n$)	id_α, $t < \alpha n$	Theorem 7	Efficiency Leveraging: resulting protocol almost as efficient as π_1 when there is no abort.[9]
(abort-secure π_1, $t < \alpha n$) and (full-secure π_2, $t < \alpha n$)	full, $t < \alpha n$	Theorem 8	Efficiency Leveraging: resulting protocol is almost as efficient as π_1. From Theorem 7 and Theorem 4. Relies on a non-blackbox decomposition of the function.

Table 1: A summary of the main black-box transformations in this paper. The first column lists the type of the protocol scheme(s) given, and the second column lists the type of protocol scheme obtained. t stands for the number of parties that can be corrupted. id_α-security denotes partially-identifiable-abort security, in which, in the event of an abort, a set of parties, at least α fraction of which are corrupt, is identified by all honest parties. sh-security stands for security against semi-honest corruption, abort and full-security stand for security against active corruption, with the latter having guaranteed output delivery.

schemes are used to define the protocols produced by these nodes, as long as those schemes meet the required security conditions, the resulting protocol will be a protocol for f with security against active corruption of any number of parties.

New Transformations. We present several new transformations, some of which are summarized in Table 1. In particular, we show how to transform a low-threshold fully-secure protocol scheme and a high/optimal-threshold semi-honest secure protocol scheme to a high/optimal-threshold protocol with full-security (presented as Corollary 1). The main step is to achieve a weaker notion of security (called "security with partially-identifiable-abort") against the same high

[9] Note that a naïve protocol which runs π_1 first and in the event of an abort, runs π_2 for the same functionality does not work. If π_1 aborting is considered as an abort event, then it gives the same efficiency guarantee, but is not an id_α-secure scheme, because if π_2 completes without an abort, the protocol fails to identify an α-corrupt set. If π_1 aborting is not considered an abort event, the protocol fails to meet the efficiency guarantee.

fraction of corruption. Then, we show how a protocol with partially-identifiable-abort security can be transformed to one with full-security.

The second of these two transformations turns out to be easy, using "Error-Correcting Secret-Sharing" or ECSS (also known as robust secret-sharing) [5], which can be realized easily using ordinary Secret-Sharing and one-time message authentication codes (MAC) (see the full version). Partially-identifiable-abort-security allows us to perform, in case of an abort, a *player elimination* process, so that an honest majority is maintained. By carrying this out not on the original function, but on a function which accepts ECSS-shared inputs and produces ECSS-shared outputs, we show how to obtain full-security. The more challenging transformations is obtaining partially-identifiable-abort-security in the first place, as discussed below.

Obtaining Partially-Identifiable-Abort Security. This transformation is based on the IPS transformation [21] which, however, was not designed for the setting with an honest majority. Hence, it relied on an OT-hybrid model, and could obtain only "security with abort." We modify this transformation in a couple of ways to obtain partially-identifiable-abort security in the honest-majority setting, in the plain model (with a broadcast channel). There are two major modifications we introduce, summarized below.

Watchlist Channels in the Plain Model. An important aspect of the IPS transformation is a collection of "watchlist channels" used by each party to monitor secretly chosen instances of a semi-honest secure inner protocol. In the IPS transformation, Rabin OT is used to implement the watchlist channel. Instead, we rely on a weaker variant, $\widetilde{\mathsf{OT}}$, which we can directly implement in the honest-majority setting (without even broadcast channels), using Shamir's secret-sharing. $\widetilde{\mathsf{OT}}$ allows an adversary to selectively cause aborts when there is no erasure. The reason this suffices for building a watchlist channel is that this functionality will be applied to random inputs, and when an abort occurs, we can safely identify a pair of inconsistent parties – at least one of which is corrupt – by having all parties reveal their views in the protocol (over a broadcast channel).[10]

Obtaining Partially-Identifiable Abort Instead of Abort. In the original IPS transformation, even if the outer protocol has security with guaranteed output delivery, the final protocol offers only security with abort (without any identification of the corrupt parties). This is due to the fact that when a party detects an inconsistency, it simply aborts the protocol. In the setting with honest majority, we show how to modify the IPS transformation, so as to obtain partially-identifiable

[10] When no abort occurs, the adversary can indeed learn some information (i.e., that an erasure occurred), but this can happen only in a small number of instances before an abort occurs.

abort, such that a set of two parties can be identified of which at least one is guaranteed to be corrupt.

Consider when P_i detects an inconsistency in the messages reported over a watchlist channel that it has access to, in an inner protocol session. In this case, P_i cannot exactly identify the source of inconsistency, but only localize it to a pair of parties P_{i_1}, P_{i_2}, one of which is corrupt. However, since P_i itself could be a corrupt party, at this point the honest parties can agree on one of (P_i, P_{i_1}, P_{i_2}) being corrupt. But being able to identify a set in which only $1/3$ fraction is guaranteed to be corrupt falls below our required guarantee of 1 out of 2 being corrupt.

To further localize corruption, we require all the parties to broadcast their views in the inner-protocol session in which an inconsistency was detected, as they had earlier communicated over the watchlist channel to P_i. If an inconsistency is detected among the broadcast views, then all parties can identify a pair (P_{i_1}, P_{i_2}) which are inconsistent with each other. On the other hand, if all the views that are broadcast are consistent with each other, then, if P_i had indeed observed an inconsistency earlier, it can point out one party P_{i_1} which reported a view over the watchlist channel different from the one it reported over the broadcast channel. Then P_i is required to broadcast this party's identity, and all parties agree on the pair (P_i, P_{i_1}).

To see that this transformation retains security, note that by causing an abort, the adversary can cause at most one server's computation to be revealed over the broadcast channel. This corresponds to the adversary corrupting one extra server in the outer protocol. Since the choice of parameters in the IPS compiler leaves a comfortable margin for the number of server corruptions, this does not affect the overall security.

Efficiency Improvements. When considering a non-constant number of parties, there are a couple of major sources of inefficiency in the transformation above, which we can address.

Firstly, in the transformation from partially-identifiable-abort security to full security, the protocol could be restarted $\Theta(n)$ times. To avoid this overhead, we require the function to be given in the form of a composition of $\Theta(n)$ functions (for instance, a layered circuit with $\Theta(n)$ layers), each one of approximately the same size complexity. Then, one can restrict the duplicated effort for each restart to correspond to a single component, and can ensure that overall $O(n)$ restarts can only about double the cost.

Secondly, in the IPS compiler, every party can potentially watch every inner protocol session. This requires that all the communication in each inner-protocol session is sent out (encrypted with one-time pads) to all the n parties. To avoid this overhead, we can use an expander graph to define which parties may watch

the execution of which servers. Specifically, we can use an expander graph between the set of parties and the set of servers in the outer protocol, in which *the degree of each server is a constant*, but any subset of $n/2$ parties has in its neighborhood (i.e., will potentially watch) almost all of the servers. Thus, the communication in each inner-protocol session (corresponding to the servers in the outer protocol) is sent out to only a constant number of parties.

Efficiency Leveraging: Transformations for Improving Efficiency. We present a new instance of efficiency leveraging, in which an MPC protocol scheme with full-security is "extended" by leveraging the efficiency of cheaper MPC protocols which only offer security with abort. Specifically, we show how to combine a protocol which guarantees only security with abort given an honest majority (e.g., from [13]) and a protocol with full-security given honest majority (like the one we constructed above) to obtain one which approaches the efficiency of the former protocol while enjoying full-security like the latter.

The basic idea is simple. We can obtain a protocol with $1/2$-identifiable-abort security as follows: given a functionality, we will run a protocol with security-with-abort to compute it; if the protocol terminates without aborting (as confirmed with the help of broadcast messages), then our protocol terminates successfully. If it aborts, then we run an (inefficient) MPC protocol with full-security for a functionality which accepts the views in the first protocol and detects a pair of parties with conflicting views, at least one of which is corrupt (if no conflict is detected, then a party who aborted in the first place can be identified as a corrupt party, since, as part of the security guarantees, we shall require zero probability for abort if all parties run honestly). To make this idea work, we need to ensure that the inefficient MPC is called only on a small piece of computation. With appropriate parameters for decomposition of the function, this indeed gives new asymptotic results (for relatively "narrow" circuits).

Negative Results. We prove two negative results. Firstly, we show that there is a function family $\mathcal{F}$ such that there is no "functionally blackbox" protocol scheme [27] for $\mathcal{F}$ (even for semi-honest security). The family $\mathcal{F}$ consists of boolean functions of the form f_α, where $\alpha \in \{0,1\}^k$ and $f_\alpha(x,y) = 1$ if and only if $x \oplus y = \alpha$.

Our second negative result shows a function family $\mathcal{G}$ such that semi-honest secure protocol schemes for $\mathcal{G}$ cannot be converted in a blackbox manner to protocols with active security (with abort). We choose $\mathcal{G}$ to be the family of zero-knowledge proofs for a class of relations. Then, there is a semi-honest secure protocol for $\mathcal{G}$ which only accesses the given functionality $f \in \mathcal{G}$ in a blackbox manner. Hence, a blackbox transformation from semi-honest secure protocol schemes to schemes with active security translates to a functionally blackbox protocol scheme for $\mathcal{G}$ with active security.

To complete the proof, we show how to define $\mathcal{G}$ (assuming the existence of a pseudorandom function) such that there is no active secure, functionally blackbox protocol scheme for $\mathcal{G}$.

1.3 Organization of the Paper

The rest of the paper is organized as follows (with some of the details deferred to the full version). Section 2 includes several basic definitions of the framework, and Section 3 defines the notion of a blackbox transformation. In Section 4, we give some simple transformations, including a new transformation that improves on a recent result by [17]. Section 5 presents two impossibility results regarding blackbox transformations. Section 6 through Section 8 present several transformations, which are summarized in Table 1. Section 9 presents the results we obtain by applying these transformations to protocol schemes in the literature.

2 Preliminaries

The basic objects in our framework are *protocols*. Technically, a protocol is specified by a single program (say, Turing Machine) for the "next-message function" of all the parties in the protocol (formally defined in the full version). We shall write Π to denote the set of all protocols.

A *functionality* is technically just a special instance of a protocol, involving a trusted party. We often abuse our notation and refer to the trusted party's program as the functionality. We shall often refer to a *functionality family* $\mathcal{F}$, which is simply a set of functionalities, i.e., $\mathcal{F} \subseteq \Pi$. We denote the family of all probabilistic polynomial time computable secure function evaluation functionalities by $\mathcal{F}^*$ (represented by circuits).

We use a *synchronous model* of communication (with rushing adversaries), so that all parties in a protocol proceed in a round-by-round fashion. Note that this is applicable to ideal functionalities too. However, typically we are not interested in the exact number of rounds in the ideal functionality, as long as it finishes within a polynomial number of rounds.

2.1 Security Definitions

Technically, a *security definition* for a functionality family $\mathcal{F}$ is formalized as a relation $\Lambda \subseteq \mathcal{F} \times \Pi$. The intention is that $(f, \pi) \in \Lambda$ iff π is a secure protocol for f. For a security notion named secure, the corresponding relation will typically be written as $\Lambda_{\mathsf{secure}}$.

$\Lambda^{\mathcal{F}}_{\mathsf{secure}}$	(f,π) s.t. $f \in \mathcal{F}$ and π meets the definition secure (for a polynomial-round version of f). If $\mathcal{F} = \mathcal{F}^*$, the family of all probabilistic polynomial time function evaluation functionalities, we simply write $\Lambda_{\mathsf{secure}}$.		
α-secure	secure, restricted to corruption of strictly less than α fraction of the parties.	secure/F	protocol is in the F-hybrid model. e.g., secure/BC denotes protocols using broadcast channels.
sa	standalone security (default is UC security).	ppt	adversary is PPT (default is unbounded adversary).
sh	semi-honest adversary.	full	active adversary (with guaranteed output delivery).
abort	adversary may learn its output and then decide which honest parties get their outputs and which do not.	id_θ	same as abort, but on abort, honest parties agree on a non-empty set of parties, at least a θ fraction of which is corrupt. We shall abbreviate α-id_α as α-id.

Table 2: Terminology used for guarantees from protocols.

In Table 2 we name some of the main security definitions considered in our results. For instance, $\Lambda^{\mathcal{F}}_{\alpha\text{-full/BC}}$ includes all pairs (f,π) such that f is a functionality in the family $\mathcal{F}$, and π is a UC-secure protocol with guaranteed output delivery (within a polynomial number of rounds), against computationally unbounded adversaries who may adaptively corrupt strictly less than α fraction of the parties, and BC means that the protocol uses a broadcast channel. In all our security notions, for simplicity of our transformations, we require that an honest party aborts the protocol only if there is no possible honest execution of the protocol that is consistent with its view. We also define a security notion generalizing the notion of security with identifiable abort:

Security with θ-Identifiable Abort. Given a functionality f, we define a functionality $f^{\langle \mathsf{id}_\theta \rangle}$ to formalize the notion of security with θ-identifiable abort. As defined in the full version, we require the functionalities to be in a normal form, involving a computation phase and an output delivery phase. $f^{\langle \mathsf{id}_\theta \rangle}$ internally runs f and interacts with Adv as follows.

1. Accept the inputs from all parties (including honest parties and parties corrupted by Adv) and forward to f. (If there is no input from P_i, substitute it with a dummy input.) Set the output vector as set by f.
2. If Adv sends getoutput, then send the corrupted parties' outputs to Adv.
3. If Adv sends $(\mathsf{corrupt}, T)$ s.t. T is a subset of parties in which at least a θ fraction are corrupt, then change the output of all honest parties to be $(\mathsf{corrupt}, T)$.
4. **Output phase:** Deliver the (current) output to all parties.

2.2 Protocol Schemes

A *protocol scheme* maps a functionality to a protocol (with a desired security property).

Definition 1 (Λ-scheme). *$\mathcal{P} : \mathcal{F} \to \Pi$ is said to be a* Λ-scheme *if $\mathcal{F}$ is a functionality family such that $\Lambda \subseteq \mathcal{F}^* \times \Pi$, and for every $f \in \mathcal{F}$, $(f, \mathcal{P}(f)) \in \Lambda$.*

For example, the semi-honest BGW-protocol scheme is a $\Lambda^{\mathcal{F}}_{\alpha\text{-}\mathsf{sh}}$-scheme where $\mathcal{F}$ is the family of all circuit-evaluation functionalities and $\alpha = \frac{1}{2}$. Typical protocol schemes are *uniform*, in that there is a Turing Machine which, on input a standardized description of f, for $f \in \mathcal{F}$, outputs the code of $\mathcal{P}(f)$.

Complexity Notation. To discuss asymptotic efficiency guarantees of protocol schemes, we augment the notation for security definitions to include protocols' communication (and sometimes, computational) cost. Typically, a protocol's complexity is measured as a function of some complexity measure of the functionality f that it is realizing, as well as the number of parties n and the security parameter k of the protocol execution. For each functionality family, we shall require a cost measure $\mathsf{size} : \mathcal{F} \to \mathbb{Z}^+$, that maps $f \in \mathcal{F}$ to a positive integer. We stress that a functionality f denotes a specific implementation (of a trusted party in a protocol), and so there can be different $f \in \mathcal{F}$ which are all functionally equivalent, but with differing values of $\mathsf{size}(f)$.

To capture the typical efficiency guarantees in the literature, we define a p-$\Lambda^{\mathcal{F}}_{\mathsf{secure}}$ scheme as a $\Lambda^{\mathcal{F}}_{\mathsf{secure}}$ scheme $\mathcal{P}$ such that for any $f \in \mathcal{F}$, $\mathcal{P}(f)$ is a protocol whose communication cost (for n parties, and security parameter k) is

$$O(p(n,k) \cdot \mathsf{size}(f) + \mathrm{poly}(n,k)). \tag{1}$$

For typical functionality families $\mathcal{F}$, a functionality $f \in \mathcal{F}$ is represented as a circuit C_f, and $\mathsf{size}(f)$ is the size of C_f. The function $p(n,k)$ reflects the multiplicative overhead of secure computation, on top of the size of the (insecure) computation.

Often, protocol schemes which offer a smaller value for $p(n,k)$ incur additive costs. To denote protocol schemes with such complexities, we use a more detailed notation: $(p,q,r;\mathsf{D})$-$\Lambda^{\mathcal{F}}_{\mathsf{secure}}$ schemes are $\Lambda^{\mathcal{F}}_{\mathsf{secure}}$ schemes $\mathcal{P}$ such that for all $f \in \mathcal{F}$, the communication cost of $\mathcal{P}(f)$ is $O(p(n,k) \cdot \mathsf{size}(f) + \mathrm{poly}(n,k) \cdot \mathsf{D}(f))$, its *computation cost* is $O(q(n,k) \cdot \mathsf{size}(f) + \mathrm{poly}(n,k) \cdot \mathsf{D}(f))$, and its *randomness cost* is $O(r(n,k) \cdot \mathsf{size}(f) + \mathrm{poly}(n,k) \cdot \mathsf{D}(f))$. Here D is a secondary cost measure – typically the depth of the circuit C_f – which is often much smaller than $\mathsf{size}(f)$. We omit D to indicate that $\mathsf{D}(f)$ is a constant and omit q and/or r to leave them as unspecified $\mathrm{poly}(n,k)$ functions. We

omit $\mathcal{F}$ if it equals $\mathcal{F}^*$, the family of all probabilistic polynomial time function evaluation functionalities.

For functionality families using circuit representation, a traditional choice for D is depth: $\mathsf{depth}(f)$ denotes the depth of the circuit C_f representing f. We shall find it useful to define another function width, defined as follows. For any topological sorting of the gates in the circuit, define a sorted-cut as a partition of the gates into two sets so that all the gates in one part appear before any gate in the other part, in the topologically sorted order; the max-sorted-cut for a sort order is the maximum number of wires crossing a sorted-cut. $\mathsf{width}(f)$ is the value of the max-sorted-cut of C_f minimized over all topological sorts of C_f. (Alternately, we could require the topological sort to be part of the circuit specification. In this case, an appropriate model of computation would be a *linear bijection straight-line program* [1], and width would correspond to the number of "registers" in the program.)

For protocol schemes providing partially-identifiable security, like α-id-schemes, we sometimes want to distinguish the cost of an execution without an abort event and that with an abort event (and identification): a $\langle \gamma, \delta \rangle$ - $\Lambda_{\alpha\text{-}\mathsf{id}}$ scheme denotes a $\Lambda_{\alpha\text{-}\mathsf{id}}$ scheme $\mathcal{P}$ such that the communication cost of $\mathcal{P}(f)$ is $O(\gamma(n,k) \cdot \mathsf{size}(f) + \mathrm{poly}(n,k))$ without abort events and $O(\delta(n,k) \cdot \mathsf{size}(f) + \mathrm{poly}(n,k))$ with abort.

Finally, we write $(p, q, r; \mathsf{D})\text{-}\widetilde{\Lambda}^{\mathcal{F}}_{\mathsf{secure}}$ instead of $(p, q, r; \mathsf{D})$ - $\Lambda^{\mathcal{F}}_{\mathsf{secure}}$ and so on, if we intend to use $\widetilde{O}(\cdot)$ instead of $O(\cdot)$ in the above costs.[11] The notation is summarized in Table 3.

$(p, q, r; \mathsf{D})$ - $\Lambda_{\mathsf{secure}}$	$\Lambda_{\mathsf{secure}}$ scheme $\mathcal{P}$ s.t. the communication cost of $\mathcal{P}(f)$ is $O(p(n,k) \cdot \mathsf{size}(f) + \mathrm{poly}(n,k) \cdot \mathsf{D}(f))$, the computation cost is $O(q(n,k) \cdot \mathsf{size}(f) + \mathrm{poly}(n,k) \cdot \mathsf{D}(f))$ and randomness cost is $O(r(n,k) \cdot \mathsf{size}(f) + \mathrm{poly}(n,k) \cdot \mathsf{D}(f))$.
$(p, q; D)$ - $\Lambda_{\mathsf{secure}}$	$(p, q, r; \mathsf{D})$ - $\Lambda_{\mathsf{secure}}$, where $r(n,k)$ is $\mathrm{poly}(n,k)$.
(p, q) - $\Lambda_{\mathsf{secure}}$	$(p, q; \mathsf{D})$ - $\Lambda_{\mathsf{secure}}$, where $D(f)$ is a constant
$(p; D)$ - $\Lambda_{\mathsf{secure}}$	$(p, q; \mathsf{D})$ - $\Lambda_{\mathsf{secure}}$, where $q(f)$ is $\mathrm{poly}(n,k)$
p - $\Lambda_{\mathsf{secure}}$	$(p, q; \mathsf{D})$ - $\Lambda_{\mathsf{secure}}$, where $D(f)$ is a constant and $q(f)$ is $\mathrm{poly}(n,k)$
$\langle \gamma, \delta \rangle$ - $\Lambda_{\alpha\text{-}\mathsf{id}}$	$\Lambda_{\mathsf{secure}}$ scheme $\mathcal{P}$ s.t. the communication cost of $\mathcal{P}(f)$ is $O(\gamma(n,k) \cdot \mathsf{size}(f) + \mathrm{poly}(n,k))$ without abort events and $O(\delta(n,k) \cdot \mathsf{size}(f) + \mathrm{poly}(n,k))$ with abort.
(params)$\text{-}\widetilde{\Lambda}_{\mathsf{secure}}$	Similar to (params) - $\Lambda_{\mathsf{secure}}$ scheme, but with $\widetilde{O}(\cdot)$ instead of $O(\cdot)$.

Table 3: Additional notation for protocol schemes (for n parties, and security parameter k).

[11] $\widetilde{O}(h)$ denotes $O(h \cdot \mathrm{polylog} h)$.

2.3 Error-Correcting Secret-Sharing

Some of our transformations rely on a simple variant of secret-sharing that has been referred to as robust secret-sharing or as honest-dealer VSS [26,10,5]. To clarify the nature of this primitive, we shall call it *Error-Correcting Secret-Sharing (ECSS)*, and define it formally below.

Definition 2 (Error-Correcting Secret Sharing). *A pair of algorithms* (share, reconstruct) *is said to be an* (n, t)*-Error-Correcting Secret Sharing (ECSS) scheme over a message space* $\mathcal{M}$ *if the following hold:*

1. **Secrecy:** *For all* $s \in \mathcal{M}$ *and* $N_c \subseteq [n], |N_c| < t$*, the distribution of* $\{\sigma_i\}_{i \in N_c}$ *is independent of* s*, where* $(\sigma_1, ..., \sigma_n) \leftarrow \mathsf{share}(s)$.
2. **Reconstruction from upto** t **erroneous shares:** *For all* $s \in \mathcal{M}$*, and all* $(\sigma_1, ..., \sigma_n)$ *and* $(\sigma'_1, ..., \sigma'_n)$ *such that* $\Pr[(\sigma_1, ..., \sigma_n) \leftarrow \mathsf{share}(s)] > 0$ *and* $|\{i \mid \sigma'_i = \sigma_i\}| \geq n - t$*, it holds that* $\mathsf{reconstruct}(\sigma'_1, ..., \sigma'_n) = s$.

3 Defining Black-Box Transformations

In this section, we present our framework of black-box transformations, which operates on protocol schemes (Definition 1). More specifically, a black-box transformation defines a Λ-scheme in terms of Λ'-schemes, for one or more other security notions Λ'. We present our definition in two parts – first the syntax of a transformation, followed by its security requirements.

Definition 3 (Black-Box Transformation (BBT): Syntax). *A* BBT for a functionality family $\mathcal{F}$ *is defined as a circuit* C *with*

- *a single input wire taking a functionality* $f \in \mathcal{F}$*,*
- *a single output wire outputting a protocol* $\pi \in \Pi$*,*
- *one or more* black-box nodes *labeled with oracle TMs* $T_1, \cdots, T_s$*,*
- *one or more* protocol nodes *labeled with relations* $\Lambda_1, \cdots, \Lambda_t$ *where* $\Lambda_i \subseteq \mathcal{F}_i \times \Pi$ *for some functionality family* $\mathcal{F}_i$*.*

For a black-box node labeled with T_i *we require that the number of oracles accessed by* T_i *is equal to the number of input wires to that node. For a protocol node, we require that there is only one input wire.*

Given such a circuit C and protocol schemes $\mathcal{P}_1, \cdots, \mathcal{P}_t$ such that each $\mathcal{P}_i$ is a Λ_i-scheme, we define $C^{\mathcal{P}_1,...,\mathcal{P}_t}(f) \in \Pi$ as follows. We shall set the value on each wire in C to be a protocol in Π (possibly a functionality), starting with the input wire and ending with the output wire, which is taken as the value

$C^{\mathcal{P}_1,\ldots,\mathcal{P}_t}(f)$. First, set the value on the input wire to be f. Then, for any black-box node with all its input wires' values already set to values $\pi_1, \cdots, \pi_d$, set its output wire's value to $T_i^{\pi_1,\cdots,\pi_d}$, where T_i is the label on the node. For any protocol node with its input wire's value set to π, set its output wire's value to $\mathcal{P}_i(\pi)$, where i is the index of the protocol node in C (if $\mathcal{P}_i(\pi)$ is undefined, then $C^{\mathcal{P}_1,\ldots,\mathcal{P}_t}(f)$ is undefined).

Definition 4 (Black-Box Transformation (BBT)). *We say that a BBT C, for a functionality family $\mathcal{F}$, is a* BBT from $\{\Lambda_1, \cdots, \Lambda_t\}$ to Λ, *if C has t protocol nodes labeled with $(\Lambda_1, \cdots, \Lambda_t)$ and, for all $f \in \mathcal{F}$ and all $(\mathcal{P}_1, \cdots, \mathcal{P}_t)$ such that each $\mathcal{P}_i$ is a Λ_i-scheme, we have $(f, C^{\mathcal{P}_1,\ldots,\mathcal{P}_t}(f)) \in \Lambda$.*

4 Examples of Black-Box Transformations

In the full version, we illustrate how several important constructions from the literature are in fact BBTs from simpler security notions or simpler function families, to more demanding ones. This list includes Bracha's compiler [4] (from high-threshold (and low-efficiency) security and low-threshold (and high-efficiency) security to a high-threshold (and high-efficiency) security), the IKOS compiler [19] (from semi-honest secure MPC and and honest-majority secure MPC to active security for Zero-Knowledge proofs) and the IPS compiler [21] (as above, but for arbitrary MPC). The GMW compiler [15] could also be viewed as a BBT (from semi-honest security and active security specialized to zero-knowledge functionality, to active security).

It is helpful to visualize these transformations using "circuit diagrams." An example of the IPS transformation was given in Figure 1. Similar diagrams for the other examples mentioned above are given in the full version.

Below we discuss two new simple BBTs, which yield much simpler alternatives to more complex constructions in the literature.

Improving Over [17]. Very recently, Hazay and Venkitasubramaniam [17], presented an IKOS-like transformation that starts from any (semi-honest) two-party protocol *in the OT-hybrid model* and gives a zero-knowledge proof system in the commitment-hybrid model. We present a different transformation that has several advantages over [17]: our transformation may start with a two-party protocol in the OLE-hybrid model,[12] whereas the one from [17] seems inherently restricted to the OT-hybrid model. Perhaps more importantly, to achieve

[12] OLE stands for Oblivious Linear function Evaluation. It is a generalization of Oblivious Transfer where a sender has (a, b) in a field $\mathbb{F}$ and the receiver has $x \in \mathbb{F}$. At the end of the protocol, the receiver will learn $ax + b$ while the sender learns nothing. OLE-based protocols are useful for arithmetic computation. Such protocols are obtained in [22] by generalizing the OT-based GMW protocol [15].

a constant level of soundness our transformation uses only a constant number of commitments (to long strings), compared to the protocol in [17] that uses as many commitments as the number of OT calls. For the simplest case of the GMW protocol applied to a boolean circuit of size s, our protocol requires only 6 commitments whose total length is $O(|C|)$ whereas the protocol from [17] requires $O(|C|)$ separate bit-commitments. These features of our transformation make it appealing for the design of practical ZK protocols based on OT-hybrid and OLE-hybrid protocols such as GMW.

Our transformation, as well as the IKOS transformation on which it is based, are presented in the full version. At a high-level, we give a simple BBT from a 2-party semi-honest MPC protocol scheme in the OLE-hybrid model to a 3-party 1-private MPC protocol scheme in the plain model; this transformation is then readily composed with the IKOS transformation (which can be applied to a 1-private protocol) to obtain our full transformation.

4.1 A Pedagogical Application

One of the results from Goldreich's textbook [14] can be simplified using a BBT. In [14], two separate protocols for $\Lambda_{\mathsf{abort\text{-}ppt\text{-}sa\text{-}id}}$ (i.e., security-with-identifiable-abort) and $\Lambda_{1/2-\mathsf{full\text{-}ppt\text{-}sa}}$ (i.e., security with guaranteed output delivery, with an honest majority) are presented, with the latter relying on VSS. Below, we give a BBT from $\Lambda_{\mathsf{abort\text{-}ppt\text{-}sa\text{-}id}}$ to $\Lambda_{1/2-\mathsf{full\text{-}ppt\text{-}sa}}$, that uses ECSS (see Section 2.3) instead of VSS.

To evaluate an n-party function f, each party shares its input using an $\lceil n/2 \rceil$-out-of-n error-correcting secret-sharing (ECSS) scheme (see Section 2.3), and sends the resulting shares to the n parties. We remark that an ECSS is much simpler than, say, a VSS protocol, and can be constructed readily by adding message authentication code (MAC) tags to the shares of any threshold secret sharing scheme (such as Shamir's scheme). Then, the parties use a protocol π from the protocol scheme with security-with-identifiable-abort to evaluate a function f', which takes shares as its inputs, reconstructs them to get inputs for f, evaluates f and reshares the outputs among all parties, again using ECSS. If the shares given as inputs have fewer than $n/2$ errors, f' can error-correct and recover the original input being shared; otherwise it defines the reconstructed value to be a default value (this corresponds to the shares not being generated correctly in the first place). If the protocol π for f' does not abort, then all the parties are expected to redistribute the shares they received from π, so that each party gets all the shares of its output; due to the error-correcting property, and since the adversary can corrupt less than $n/2$ of the shares received by each honest party, every honest party will be able to correctly recover its output. On the other hand, if the protocol π aborts, due to the identifiable-abort security

guarantee, all honest parties will agree on the identity of one corrupt party. Note that at this point, even though the adversary may learn its outputs from π (i.e., outputs of f'), these carry no information and can be efficiently simulated (by a simulator running the protocol with arbitrary inputs for the honest parties). Hence, the parties can simply eliminate the identified party (and still retain honest majority), and restart the entire protocol on a smaller functionality in which the eliminated party's input is replaced by a default value. This process must eventually terminate, after at most $\lceil n/2 \rceil$ attempts, guaranteeing output for all honest parties.

An ad-hoc use of the above "player elimination" technique was made in several previous MPC protocols (see, e.g., [18] and references therein). In contrast, our use of this technique yields a *completely general transformation* from a weaker flavor of MPC to a stronger one.

5 Impossibility of Black-Box Transformations

In this section, we present some impossibility results for BBT. Before proceeding, we emphasize that in the definition of BBT, we *do not* require the security proofs to be black-box in any form. In particular, the simulators used to define security can arbitrarily depend on the functionality in a non-black-box manner. As such, the impossibility results on BBT are of a rather strong nature.

Our first impossibility results relates to an interesting special case of a BBT, namely, BBT from $\emptyset$ to Λ. This corresponds to the notion of a *functionally-black-box* protocol introduced by Rosulek [27], wherein there is an oracle TM such that for all $f \in \mathcal{F}$, T^f is a secure protocol (according to Λ) for f. Rosulek demontrated a two-party functionality family for which there is no functionally black-box protocol, *assuming the existence of one-way functions*. We present an unconditional version of this result.

Theorem 1. *There exists a two-party functionality family $\mathcal{F}$ such that there is no BBT from $\emptyset$ to $\Lambda^{\mathcal{F}}_{\mathsf{sh}}$. In particular, there is no BBT from $\emptyset$ to $\Lambda^{\mathcal{F}^*}_{\mathsf{sh}}$.*

The detailed proof is given in the full version. Here we sketch the main ideas of the proof.

Proof sketch: The family $\mathcal{F}$ we shall use to prove the theorem consists of boolean functions of the form f_α, $\alpha \in \{0,1\}^k$, where $f_\alpha(x,y) = 1$ if and only if $x \oplus y = \alpha$. To show that there can be no secure protocol for f_α, in which the two parties access the function only in a blackbox manner, we consider the following experiment. Pick x, y, α uniformly and independently at random, and run the protocol for f_α with inputs x, y. Then we argue that the probability for both of the following events should be negligible:

(A) Either party queries their oracle with (p, q) such that $p \oplus q = \alpha$.

(B) Either party queries their oracle with (p, q) such that $p \oplus q = x \oplus y$.

The probability of event A is negligible since α is chosen uniformly at random, and the parties make only a polynomial number of queries. The reason for the probability of event B being negligible is the security of the protocol: in an ideal world, since $x \oplus y \neq \alpha$, a corrupt party (simulator), even given α, can learn only a negligible amount of information about the other party's input. Now, we consider a "coupled" experiment in which instead of α, we pick $\alpha^* = x \oplus y$, and run the same protocol but now for f_{α^*}. It can be argued that for the random tapes in the protocol for which events (A) and (B) does not occur in the first case, they will not occur in the second run too. Thus with high probability, both the executions produce the same output, violating the correctness of the protocol. □

Also, we consider the question of showing impossibility of BBT from semi-honest security to active security. We present such a result conditioned on the existence of one-way functions.

Theorem 2. *Assuming the existence of one-way functions, there exists a two-party functionality family $\mathcal{G}$ such that there is no BBT from $\{\Lambda^{\mathcal{G}}_{\mathsf{sh}}\}$ to $\Lambda^{\mathcal{G}}_{\mathsf{abort}}$.*

We present the intuition behind the proof below, and defer the detailed proof to the full version. *Proof sketch:* We will let $\mathcal{G}$ to be the family of zero-knowledge proofs for a class of relations. Then, there is a semi-honest secure protcol for $\mathcal{G}$ which only access the given functionality $f \in \mathcal{G}$ in a blackbox manner. Hence, a blackbox transform from semi-honest secure protocol schemes to schemes with active security translates to a functionally blackbox protocol scheme for $\mathcal{G}$ with active security. To show that this does not exist, we assume the existence of a pseudorandom function F and define $\mathcal{G}$ as follows. The relations associated with $\mathcal{G}$ are $R_s = \{(x, w) \mid F_s(w) = x\}$, where F_s denotes F with seed s.

To show that there can be no ZK protocol for this relation in which the parties only have blackbox access to an oracle for the relation R_s (but the simulator may depend on s), we consider a cheating prover as follows. When given (x, w) and access to R_s, it uses a wrapper around R_s to turn it into relation which accepts (x, w) (and does not accept (x', w) for $x' \neq x$), but otherwise behaves like R_s. Then the cheating prover runs the honest prover with access to the modified oracle. Using the ZK property we can argue that an honest verifier, when given a random x, cannot detect the difference between interacting with the real prover and the cheating prover. Thus, if the protocol is complete, the cheating prover will be able to break soundness. □

6 A BBT from Partially-Identifiable-Abort to Full Security

We present a simple black-box transformation from *partially-identifiable abort security* (formalized using $\Lambda_{\alpha\text{-id}}$ below) to full security. This will be an important ingredient in our applications in Section 9. First, we present a simple but general version of this transformation (which suffices for feasibility results); in Theorem 4, we shall present a more efficient variant.

Theorem 3. *For any $0 \leq \alpha \leq 1/2$, there exists a BBT from $\Lambda_{\alpha\text{-id/BC}}$ to $\Lambda_{\alpha\text{-full/BC}}$. Specifically, there is a BBT from p - $\Lambda_{\alpha\text{-id/BC}}$ to $(np; \mathsf{D})$ - $\Lambda_{\alpha\text{-full/BC}}$, where $\mathsf{D}(f)$ is the input plus output size of f.*

Our tools behind this construction are relatively simple. In particular, we do not use verifiable secret-sharing (VSS), but instead use the much simpler primitive Error-Correcting Secret-Sharing (ECSS) (see Section 2.3), which can be realized easily using ordinary Secret-Sharing and one-time message authentication codes (MAC).

Here we give a high level overview of the construction, with a complete description defered to the full version. The idea behind this BBT is that if we have a protocol which either completes the computation or identifies a set of parties such that at least α fraction of which are corrupt, then, in the event of an abort, we can remove the identified set of parties from active computation and restart the computation. Note that this preserves the corruption threshold of α (i.e., strictly less than α fraction remains corrupt) among the set of "active" parties.

For this idea to work, we need to keep the outputs secret-shared (so that by aborting, the adversary does not learn any useful information, even though it receives its outputs from the computation), and after the computation finishes, guarantee reconstruction. Further, we need to use secret-sharing to let all the parties deliver their inputs to the set of active parties. All this will be achieved using ECSS in a straightforward manner, for $\alpha \leq 1/2$.

A More Efficient Variant. In the above BBT, we restarted the entire computation in the event of an abort. To avoid this, we rely on having access to a "layered representation" of the function. Formally, consider a parametrized functionality $\hat{f}$, parametrized by an index $i \in \{1, \cdots, d\}$, such that $f = \hat{f}[d] \circ ... \circ \hat{f}[1]$, such that $\mathsf{size}(\hat{f}[i]) = O(\mathsf{size}(f)/d)$, for all i. We define $\mathsf{width}_d(f)$ to be the smallest number w such that there exists a decomposition of f into d layers, each of size $O(\mathsf{size}(f)/d)$, such that the number of output wires from any layer is at most w. We shall typically take d to be a polynomial $d(n, k)$. Note that $\mathsf{width}(f)$ defined in Section 2.2 is an upper-bound on $\mathsf{width}_d(f)$ for all d.

Since decomposing f into $\hat{f}$ is not a black-box operation, we require a "protocol scheme" that carries out this decomposition. For this we define a $\Lambda_{\mathsf{layer}[d]}$ scheme to be one which maps f to a parametrized function $\hat{f}$ such that

$$f = \hat{f}[d] \circ \cdots \circ \hat{f}[1],$$

and $\forall i \in [d]$, $\mathsf{size}(\hat{f}[i]) = O(\mathsf{size}(f)/d)$ and the number of bits output by $\hat{f}[i] \leq \mathsf{width}_d(f)$.

Then, as shown in the full version, we obtain the following efficiency improvement over Theorem 3.

Theorem 4. *For any $0 < \alpha \leq 1/2$, there exists a BBT from $\{\Lambda_{\mathsf{layer}[d]}, \langle \gamma, \delta \rangle\text{-}\Lambda_{\alpha\text{-}\mathsf{id}}\}$ to $(\gamma; \mathsf{D})\text{-}\Lambda_{\alpha\text{-}\mathsf{full}}$, where $d(n,k) = n \cdot \frac{\delta(n,k)}{\gamma(n,k)}$ and $\mathsf{D}(f) = \mathsf{width}_d(f)$.*

7 A BBT From $\{\Lambda_{\alpha\text{-}\mathsf{sh}}, \Lambda_{\beta\text{-}\mathsf{full}}\}$ to $\Lambda_{\alpha\text{-}\mathsf{id}}$

Our goal in this section is to obtain a BBT that increases the corruption threshold of a fully secure protocol, by combining it with a semi-honest protocol which has the higher threshold. Given Theorem 3, it suffices to obtain a protocol with partially-identifiable-abort against the higher corruption threshold. Formally, we shall prove the following theorem, which is interesting when $\beta < \alpha$

Theorem 5. *For any $0 < \alpha, \beta \leq 1/2$, there exists a BBT from $\{\Lambda_{\alpha\text{-}\mathsf{sh}}, \Lambda_{\beta\text{-}\mathsf{full}}\}$ to $\Lambda_{\alpha\text{-}\mathsf{id/BC}}$.*

This BBT (detailed in the full version) resembles the IPS compiler, but achieves $1/2$-identification in case of abort, and also avoids the use of OT in watchlists. For this, it replaces T_2^{IPS} in IPS (see Figure 1) with a black-box transformation T_2. Figure 2 compares T_2^{IPS} and T_2. T_2^{IPS} consists of a "core" compiler $\mathrm{IPS}_{\mathrm{core}}$, which produces a protocol in a "watchlist-channel hybrid" model (also using OT if it is needed by the inner protocol). Separately, a watchlist-channel functionality $\mathcal{W}$ was realized using a protocol w_{IPS} in the OT-hybrid model. Finally, the former was composed with the latter to obtain a protocol in the OT-hybrid model.

In T_2, firstly the OT used in the watchlist protocol is replaced with a functionality $\widetilde{\mathsf{OT}}$, which is then implemented by a protocol $\pi_{\widetilde{\mathsf{OT}}}$ in the honest-majority setting; further this watchlist protocol is modified in a simple manner to achieve $1/2$-identification. The functionality of the resulting protocol is captured by $\mathcal{W}^*$. Next,the protocol generated by $\mathrm{IPS}_{\mathrm{core}}$ is modified to facilitate $1/2$-identification (even if given the watchlist functionality $\mathcal{W}^*$ instead of $\mathcal{W}$), following the outline sketched in Section 1.2 (see paragraph *Obtaining Partially-Identifiable-Abort Security*). The final protocol is obtained by composing this protocol with the watchlist protocol for $\mathcal{W}^*$.

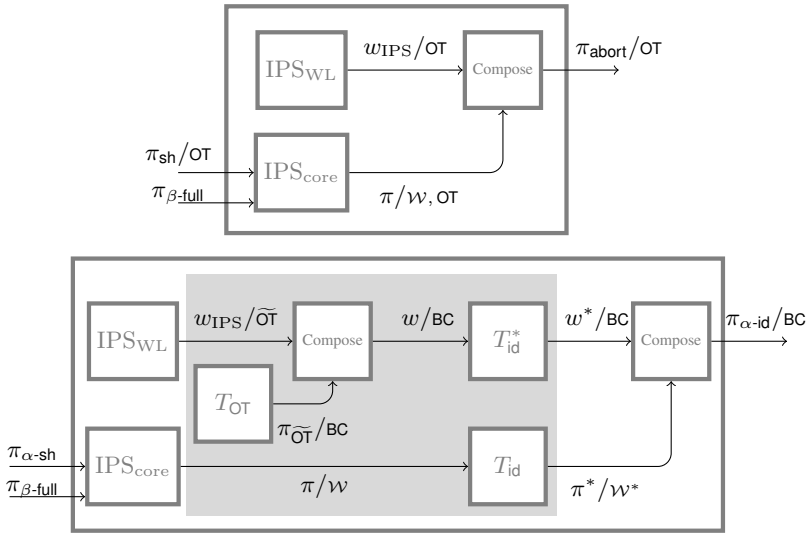

Fig. 2: T_2^{IPS} and T_2. The shaded region shows the new components in T_2. Note that T_2 retains $\mathrm{IPS}_{\mathrm{core}}$ and $\mathrm{IPS}_{\mathrm{WL}}$ from T_2^{IPS} as it is.

7.1 Using a Sparse Watchlist

The BBT in Theorem 5 is in fact a BBT from $\{(p_{\mathrm{in}}, q_{\mathrm{in}}, r_{\mathrm{in}})\text{-}\Lambda_{\alpha\text{-sh}}, (p_{\mathrm{out}}, q_{\mathrm{out}})\text{-}\Lambda_{\beta\text{-full}}\}$ to $p\text{-}\Lambda_{\alpha\text{-id/BC}}$, where $p = n^2 \cdot (p_{\mathrm{in}} + r_{\mathrm{in}}) \cdot (q_{\mathrm{out}} + n \cdot p_{\mathrm{out}})$. But by exploiting the honest majority guarantee which was absent in the setting of [21], we can state the following version.

Theorem 6. *For any $0 < \alpha, \beta \leq 1/2$, and polynomials $p_{\mathrm{in}}, q_{\mathrm{in}}, r_{\mathrm{in}}, p_{\mathrm{out}}, q_{\mathrm{out}}$, there exists a BBT from $\{(p_{\mathrm{in}}, q_{\mathrm{in}}, r_{\mathrm{in}})\text{-}\Lambda_{\alpha\text{-sh}}, (p_{\mathrm{out}}, q_{\mathrm{out}})\text{-}\Lambda_{\beta\text{-full}}\}$ to $p\text{-}\Lambda_{\alpha\text{-id/BC}}$, where $p = n \cdot (p_{\mathrm{in}} + r_{\mathrm{in}}) \cdot (q_{\mathrm{out}} + n \cdot p_{\mathrm{out}})$.*

The above result saves a factor of n compared to the previous transformation. The efficiency improvement comes from a sparser watchlist mechanism (using an expander graph to define which parties may watch the execution of which servers) in the BBT from $(\Lambda_{\beta\text{-full}}, \Lambda_{\alpha\text{-sh}})$ to $\Lambda_{\alpha\text{-id/BC}}$. We present the details in the full version.

8 Efficiency Leveraging

Bracha's transformation is a classical example of efficiency leveraging. It was originally proposed in the context of byzantine agreement [4], and later applied to MPC protocols (see, e.g., [12]). Below, we record a version of this result that is sufficient for our applications.

Proposition 1 (Bracha's Transformation [4]). *Let* $0 < \epsilon, \beta \leq \alpha \leq 1/2$*, and let* $p'(n,k) = c_n$ *be independent of* k*. Then, for each* $\mathsf{secure} \in \{\mathsf{sh}, \mathsf{abort}, \mathsf{full}\}$ *and any function* D*, there exists a BBT from* $\{(p,q;\mathsf{D})\text{-}\Lambda^{\mathcal{F}}_{\beta\text{-}\mathsf{secure}}, p'\text{-}\Lambda_{\alpha\text{-}\mathsf{secure}}\}$ *to* $(p'';\mathsf{D})\text{-}\Lambda^{\mathcal{F}}_{(\alpha-\epsilon)\text{-}\mathsf{secure}}$*, where* $p''(n,k) = p(n,k) + q(n,k)$*.*

In this section, we present a new instance of efficiency leveraging for full-security: a simple BBT from $\{\Lambda_{\alpha\text{-}\mathsf{abort}}, \Lambda_{\alpha\text{-}\mathsf{full}}\}$ to $\Lambda_{\alpha\text{-}\mathsf{full}}$, in which the resulting protocol's efficiency is comparable to that of the protocol in $\Lambda_{\alpha\text{-}\mathsf{abort}}$.

First we present a efficiency leveraging transformation for $\Lambda_{\alpha\text{-}\mathsf{id}}$ which can then be combined with Theorem 4 to obtain efficiency leveraging for $\Lambda_{\alpha\text{-}\mathsf{full}}$. In our efficiency leveraging transformation for $\Lambda_{\alpha\text{-}\mathsf{id}}$ the efficiency of the resulting protocol, *when there is no abort event*, is comparable to that of a cheaper $\Lambda_{\alpha\text{-}\mathsf{abort}}$ protocol. Formally, we have the following theorem.

Theorem 7. *For any* $0 \leq \alpha \leq 1/2$*, and functions* $p, q, p' \in \mathrm{poly}(n,k)$*, there exists a BBT from* $\{(p,q)\text{-}\Lambda_{\alpha\text{-}\mathsf{abort}}, p'\text{-}\Lambda_{\alpha\text{-}\mathsf{id}}\}$ *to* $\langle\gamma,\delta\rangle\text{-}\Lambda_{\alpha\text{-}\mathsf{id}}$*, where* $\gamma = p$ *and* $\delta = p' \cdot (p+q)$*.*

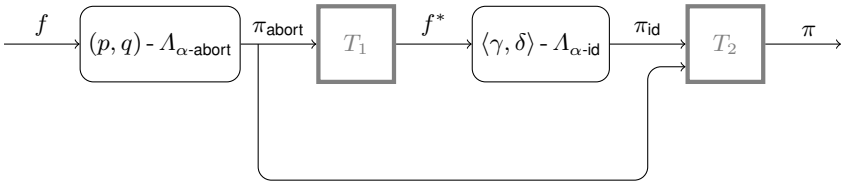

Fig. 3: Black-Box Transformation from $\{(p,q)\text{-}\Lambda_{\alpha\text{-}\mathsf{abort}}, p'\text{-}\Lambda_{\alpha\text{-}\mathsf{id}}\}$ to $\langle\gamma,\delta\rangle\text{-}\Lambda_{\alpha\text{-}\mathsf{id}}$, where $\gamma = p$ and $\delta = p' \cdot (p+q)$.

The protocol scheme claimed in Theorem 7 is shown in Figure 3. The first node is a protocol node of $p\text{-}\Lambda_{\alpha\text{-}\mathsf{abort}}$, which converts a functionality f into a protocol π_{abort}.

The second node is a black-box node T_1, which converts the protocol π_{abort} to an (n-party) functionality f^*, in which the trusted party takes the view of each party in an execution of π_{abort} as the input, carries out the execution of π_{abort}, and identifies a set of two parties which have inconsistent views, if it exists.[13] When there is none, it outputs $\emptyset$. The third node $\Lambda_{\alpha\text{-}\mathsf{id}}$ compiles f^* into a protocol π_{id}.

Finally, a black-box node T_2 combines π_{abort} and π_{id} together and transforms them into a protocol π, which works as follows: initially the parties execute π_{abort} on the given input, and on finishing this execution successfully, each

[13] Recall that the view of a party involves its initial input, the randomness, and all the received messages.

party broadcasts "done." If all parties broadcast "done," then each party outputs the output from the execution of π_{abort} and terminates. If not, they execute π_{id} with their views in the execution of π_{abort} as input. If this latter execution itself aborts, π_{id} identifies a set of parties S at least an α fraction of which is corrupt (where $\alpha \leq 1/2$). otherwise (i.e., if π_{id} finishes without an abort event), then all parties agree on the output of f^*, namely a set S of two parties at least one of which is corrupt, or the emptyset $\emptyset$; if the output is $\emptyset$, the parties set S to be the singleton set consisting of the lexicographically smallest party who did not broadcast "done" after the execution of π_{abort}. In all cases, if π_{abort} resulted in an abort, the honest parties agree on a set of parties S of which at least an α fraction is corrupt.

We verify that the complexity of π is as claimed in the theorem. When there is no abort event, the communication cost is essentially the same as that of π_{abort}, namely $p(n,k)$; otherwise, there is an additional the cost from π_{id}, which is $\widetilde{O}(p(n,k) + p'(n,k) \cdot \mathsf{size}(f^*))$, where $\mathsf{size}(f^*) = \widetilde{O}((p(n,k) + q(n,k)) \cdot \mathsf{size}(f))$. Hence the whole scheme is in $\langle \gamma, \delta \rangle$ - $\Lambda_{\alpha\text{-}\mathsf{id}}$ with $\gamma = p$ and $\delta' = p' \cdot (p+q)$.

Combining Theorem 7 with Theorem 4 we get the following result. Here we state it as efficiency leveraging for full-security; however, the result holds as a BBT from $\{\Lambda_{\mathsf{layer}[d]}, (p,q)\text{-}\Lambda_{\alpha\text{-}\mathsf{abort}}, p'\text{-}\Lambda_{\alpha\text{-}\mathsf{id}}\}$ as well.

Theorem 8. *For all $0 \leq \alpha \leq 1/2$, and for all functions $p, q, p' \in \mathrm{poly}(n,k)$, there exists a BBT from $\{\Lambda_{\mathsf{layer}[d]}, (p,q)\text{-}\Lambda_{\alpha\text{-}\mathsf{abort}}, p'\text{-}\Lambda_{\alpha\text{-}\mathsf{full}}\}$ to $(p;\mathsf{D})\text{-}\Lambda_{\alpha\text{-}\mathsf{full}}$, where $d = \frac{n \cdot p' \cdot (p+q)}{p}$ and $\mathsf{D}(f) = \mathsf{width}_d(f)$.*

9 Applications

In Section 4.1, we already saw a pedagogical application of BBT, in simplifying the exposition of security with guaranteed output delivery (with computationally bounded adversaries). In this section, we give several interesting examples regarding how to use the BBTs in the previous sections for deriving both feasibility and efficiency results.

∘ **Rabin-Ben Or without honest-majority VSS.** As our first example, we reproduce the classic feasibility result of Rabin and Ben-Or [26] for fully secure MPC for corruption against $t < n/2$ parties. The core new tool developed in this paper (and used in subsequent results in this regime of corruption) was Verifiable Secret-Sharing (VSS) that is secure against corruption of $t < n/2$ parties. Interestingly, our construction by-passes the need for an explicit VSS protocol for this corruption regime, instead showing that one can directly use fully secure

MPC from prior work [2,6]. Our construction is based on the following direct corollary of Theorem 5 and Theorem 3.

Corollary 1. *For any $0 < \alpha, \beta \leq 1/2$, there exists a BBT from $\{\Lambda_{\alpha\text{-sh}}, \Lambda_{\beta\text{-full}}\}$ to $\Lambda_{\alpha\text{-full/BC}}$.*

To obtain the result of [26] we simply apply Corollary 1 to the protocols in [2,6].

∘ **Constant-Rate MPC with Full-Security for Small Number of Parties.** Our first quantitative result is a "constant-rate" honest-majority MPC protocol with guaranteed output delivery, *when the number of parties involved is constant*. That is, as the size of the function grows, the communication complexity of the protocol grows linearly at a rate that is independent of the security parameter. For MPC of large circuits, against the optimal corruption threshold $n/2$, this gives an amortized complexity of $O(1)$ per gate, compared to $O(k)$ per gate in the previously best result from [3].

Corollary 2. *There exists a p-$\Lambda_{1/2\text{-full/BC}}$-scheme, where $p(n,k) = c_n$ is independent of k.*

This result is obtained as a corollary of Theorem 6[14] and Theorem 3. First we obtain a p-$\Lambda_{1/2\text{-id/BC}}$ scheme by applying the BBT from Theorem 6 to the $\Lambda_{1/2\text{-sh}}$-scheme from [2] and the constant rate $\Lambda_{\beta\text{-full}}$-scheme (for some $\beta > 0$) that is obtained by instantiating the protocol scheme from [11] using the constant-rate ramp scheme of [7]. (The same "outer protocol" was used in [21] to obtain a constant-rate $\Lambda_{\text{abort/OT}}$-scheme.) Then by further applying the BBT fromTheorem 3, we obtain the p-$\Lambda_{1/2\text{-full/BC}}$ protocol as claimed.

∘ **Scalable MPC with Full-Security, Optimal Threshold.** Our next result is a "scalable" honest-majority MPC protocol with guaranteed output delivery. We define the function class $\mathcal{F}_{\text{arith}}$ of functions represented as arithmetic circuits over a field $\mathbb{F}$ such that $\log|\mathbb{F}| > k$. For $f \in \mathcal{F}_{\text{arith}}$, $\mathsf{size}(f)$ refers to $\log|\mathbb{F}| \cdot |C_f|$, where $|C_f|$ is the number of gates in the circuit C_f representing f. Equivalently, $\mathsf{size}(f)$ measures the number of binary wires in the circuit C_f; similarly $\mathsf{width}(f)$ measures the width of C_f in bits.

Corollary 3. *There exists a $(p;\mathsf{D})$-$\Lambda^{\mathcal{F}_{\text{arith}}}_{\frac{1}{2}\text{-full/BC}}$-scheme, where $p(n,k) = n$ and $\mathsf{D} = \mathsf{width}(f)$.*

[14] The construction leading to Theorem 5 also suffices here. We point to Theorem 6 only because it makes the parameters explicit; the optimization in Section 7.1 is not important for this result.

That is, for MPC of large arithmetic circuits over a large field, with security against the optimal corruption threshold $n/2$, we get an amortized communication cost of $O(n)$ bits per binary wire in the circuit. This result is obtained as a corollary of Theorem 7 and Theorem 4, by applying the BBTs to the $\Lambda^{\mathcal{F}_{\text{arith}}}_{1/2\text{-abort}}$-scheme from [13] and the p-$\Lambda_{1/2\text{-id}}$-scheme from Corollary 2. Note that we have used $\mathsf{width}(f)$ as an upper-bound on $\mathsf{width}_d(f)$ over all d.

Our result complements a similar result of Ben-Sasson et al. [3] in which the secondary complexity measure is depth, instead of width. We remark that a natural regime for scalable MPC involves long sequential computations (carried out by a small or moderate number of parties), so that a circuit for the computation would be deep and narrow. In such a regime, the above result, which yields a cost of $O(n \cdot \mathsf{size}(f) + \mathrm{poly}(n,k))$, compares favorably to the protocols of [3] which yield a cost of $\widetilde{\Omega}(n \cdot \mathsf{size}(f) + n^2 \cdot \mathsf{depth}(f) + \mathrm{poly}(n,k))$.

∘ **Highly Scalable MPC with Full-Security, Near Optimal Threshold.** Our final application considers the problem of relaxing the corruption threshold from the optimal $\alpha = 1/2$ to $\alpha = 1/2 - \epsilon$, for any constant ϵ.

Corollary 4. *For every $\epsilon > 0$, there exists a $(p_\epsilon; \mathsf{D})$-$\Lambda_{(\frac{1}{2}-\epsilon)\text{-full/BC}}$-scheme, where $p_\epsilon(n,k) = c_\epsilon$ is independent of n and k and $\mathsf{D}(f) = \mathsf{depth}(f)$.*

This generalizes a result in [12], which obtained a similar result (without using a broadcast channel) for the threshold $\frac{1}{3} - \epsilon$. We obtain this result by applying Proposition 1 (Bracha's efficiency leveraging transformation) to our c_n-$\Lambda_{\frac{1}{2}\text{-full/BC}}$ scheme from Corollary 2 and the $(c_1, c_2; \mathsf{depth})$-$\Lambda_{\beta\text{-full}}$ scheme from [12] (for, say, $\beta = 1/6$ and c_1, c_2 being constants), with $\alpha = 1/2$.

References

1. M. Ben-Or and R. Cleve. Computing algebraic formulas using a constant number of registers. *SIAM Journal on Computing*, 21(1):54–58, 1992.
2. M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. 20th STOC*, pages 1–10. ACM, 1988.
3. E. Ben-Sasson, S. Fehr, and R. Ostrovsky. Near-linear unconditionally-secure multiparty computation with a dishonest minority. In *Proc. 32th CRYPTO*, pages 663–680. Springer, 2012.
4. G. Bracha. An o(log n) expected rounds randomized byzantine generals protocol. *J. ACM*, 34(4):910–920, 1987.
5. A. Cevallos, S. Fehr, R. Ostrovsky, and Y. Rabani. Unconditionally-secure robust secret sharing with compact shares. In *Proc. 31th EUROCRYPT*, pages 195–208. Springer, 2012.
6. D. Chaum, C. Crépeau, and I. Damgård. Multiparty unconditionally secure protocols. In *Proc. 20th STOC*, pages 11–19. ACM, 1988.
7. H. Chen and R. Cramer. Algebraic geometric secret sharing schemes and secure multi-party computations over small fields. In *CRYPTO*, pages 521–536. Springer, 2006.

8. R. Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In *STOC*, pages 364–369. ACM, 1986.
9. R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient multiparty computations secure against an adaptive adversary. In *EUROCRYPT '99*, pages 311–326, 1999.
10. R. Cramer, I. Damgård, and S. Fehr. On the cost of reconstructing a secret, or VSS with optimal reconstruction phase. In *Proc. 21th CRYPTO*, pages 503–523. Springer, 2001.
11. I. Damgård and Y. Ishai. Scalable secure multiparty computation. In *Proc. 26th CRYPTO*, pages 501–520. Springer, 2006.
12. I. Damgård, Y. Ishai, and M. Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In *Proc. 29th EUROCRYPT*, pages 445–465. Springer, 2010.
13. D. Genkin, Y. Ishai, M. M. Prabhakaran, A. Sahai, and E. Tromer. Circuits resilient to additive attacks with applications to secure multiparty computation. In *The Proceedings of the 46th Annual Symposium on the Theory of Computing (STOC)*, 2014.
14. O. Goldreich. *Foundations of Cryptography: Basic Applications*. Cambridge University Press, 2004.
15. O. Goldreich, S. Micali, and A. Wigderson. How to play ANY mental game. In ACM, editor, *Proc. 19th STOC*, pages 218–229. ACM, 1987. See [14, Chap. 7] for more details.
16. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. In *Proc. 17th STOC*, pages 291–304. ACM, 1985.
17. C. Hazay and M. Venkitasubramaniam. On the power of secure two-party computation. Cryptology ePrint Archive, Report 2016/074, http://eprint.iacr.org/2016/074, 2016. To appear in *Proc. Crypto 2016*.
18. M. Hirt and J. B. Nielsen. Upper bounds on the communication complexity of optimally resilient cryptographic multiparty computation. In *Advances in Cryptology - ASIACRYPT 2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4-8, 2005, Proceedings*, pages 79–99, 2005.
19. Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Zero-knowledge from secure multiparty computation. In *STOC*, pages 21–30. ACM, 2007.
20. Y. Ishai, R. Ostrovsky, and V. Zikas. Secure multi-party computation with identifiable abort. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, pages 369–386, 2014.
21. Y. Ishai, M. Prabhakaran, and A. Sahai. Founding cryptography on oblivious transfer - efficiently. In *CRYPTO*, pages 572–591. Springer, 2008.
22. Y. Ishai, M. Prabhakaran, and A. Sahai. Secure arithmetic computation with no honest majority. In O. Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 294–314. Springer, 2009.
23. M. Jawurek, F. Kerschbaum, and C. Orlandi. Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 955–966, 2013.
24. J. Kilian. Founding cryptography on oblivious transfer. In *STOC*, pages 20–31. ACM, 1988.
25. J. Perry, D. Gupta, J. Feigenbaum, and R. N. Wright. Systematizing secure computation for research and decision support. In *Proc. 9th SCN*, pages 380–397. Springer, 2014.
26. T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proc. 21st STOC*, pages 73–85. ACM, 1989.
27. M. Rosulek. Must you know the code of f to securely compute f? In *Proc. 32th CRYPTO*. Springer, 2012.
28. A. Shamir. How to share a secret. *Communications of the ACM*, 22(11), Nov. 1979.
29. A. Shamir, R. L. Rivest, and L. M. Adleman. Mental poker. Technical Report LCS/TR-125, Massachusetts Institute of Technology, April 1979.
30. A. C. Yao. Protocols for secure computation. In *Proc. 23rd FOCS*, pages 160–164. IEEE, 1982.
31. A. C. Yao. How to generate and exchange secrets. In *Proc. 27th FOCS*, pages 162–167. IEEE, 1986.