Constructing Secure Multi-Party Computation with Identifiable Abort

Nicholas-Philip Brandt, Sven Maier, Tobias Müller, and Jörn Müller-Quade

Abstract. Statistical Multi-Party Computation (MPC) protocols based on two-party Oblivious Transfer (OT) have one severe drawback: the adversary can abort the protocol without repercussions. [IOZ14] introduced the notion of Identifiable Abort (IA). We extend the work of [FGM+01] and investigate, under which conditions \( n \)-party MPC can be constructed from smaller functionalities in the setting of IA. Previous work already contains an impossibility result for two-party functionalities [IOS12] and a universal \( n \)-party setup [IOZ14].

We thus investigate setup functionalities of size between 3 and \((n - 1)\). In this paper we give novel upper bounds for the sizes of functionalities needed for IA. In particular, we find out, that it is possible to construct \( n \)-party MPC with IA from an \( (n - 1) \)-party setup and a broadcast, if at least 3 parties are honest. We achieve our result by using a new and innovative technique called conflict graph and its complementary association graph, which uses a broadcast channel to model the knowledge of honest parties regarding the identity of malicious parties.

Keywords: Multi-Party Computation · Simulation-Based Security · Identifiable Abort · Conflict Graph

1 Introduction

Minimal Cardinality. [Kil88; IPS08] proved that two-party primitives like OT can be used to construct general \( n \)-party MPC secure against arbitrary adversaries, but only in the setting of Anonymous Abort, also known as security with abort. Surprisingly, this result does not carry over to the setting of Identifiable Abort [IOS12]. This raises the question, how strong the setup has to be in order to construct \( n \)-party MPC with IA. The construction of [IOZ14] requires Correlated Randomness, which can be modeled as an \( n \)-party functionality and as such poses a relatively strong assumption. We say that the Correlated Randomness-functionality is of size \( n \). In general, we would like to compose general \( n \)-party MPC solely from functionalities of smaller size and a global broadcast. Following [FGM+01], we call the minimal size of a functionality that enables \( n \)-party MPC with minimal complete cardinality \( k^* \) of a primitive.
Identifiable Abort. Ishai, Prabhakaran, and Sahai [IPS08] classify protocols for secure Multi-Party Computation (MPC) into two categories: (1) Protocols which are only secure with an honest majority of parties, and (2) Protocols which are secure against arbitrarily many malicious parties. The former implies some level of trust in the other parties, which can be used for techniques such as majority votes. For the latter, it was stated by the authors, that certain security guarantees cannot be made. In particular, an adversary can always abort the computation, without revealing information which allows to identify corrupted parties. We call this property Anonymous Abort (AA). Since this is an undesirable property for the real world, Ishai, Ostrovsky, and Zikas [IOZ14] introduced a compromise between what we would want for real-world protocols and what is technically feasible, which they called Identifiable Abort (IA). It still allows the adversary to abort any protocol at will, but in doing so, the identity of at least one malicious party is revealed to all participants. In our work, we try to find bounds on the minimal complete cardinality in the setting of IA. We refer to ideal functionalities $F$ with IA as $I$.

Adversarial Restrictions. In order to circumvent impossibility results or raise the efficiency, security statements often consider only a certain sub-class of adversaries. One common limitation is the computational power of the adversary. Most protocols are only secure against adversaries with polynomial runtime, where security can be based on the computational complexity of well-investigated problems. Another common limitation is on the number of parties that an adversary can corrupt. Ben-Or, Goldwasser, and Wigderson [BGW88] showed that if we only consider adversaries who corrupt less than one third of all parties, $k^* = 2$ is the minimal complete cardinality. This result already indicates some relation between the number of corrupted parties and the minimal complete cardinality. Our investigation is focused on this particular relation.

Summary. In this work, we investigate upper and lower bounds of the minimal complete cardinality $k^*$ in the style of [FGM+01], but with Identifiable Abort. As a helpful tool, we formalize the novel concept of the conflict graph, which contains information on all conflicts between parties that have occurred during a protocol execution. This tool requires the existence of a global broadcast channel, which is of size $n$, but still a relatively weak assumption. The conflict graph aids the honest parties in identifying a set of malicious parties, when too many conflicts arise.

We use the conflict graph to present a construction that uses $(n - 1)$-party Secure Function Evaluation and $n$-party broadcast to construct $n$-party SFE, without losing the properties of Identifiable Abort.

1.1 Related Work

The question regarding requirements and cardinalities for hybrid functionalities to achieve general $n$-party MPC has been investigated for several different scenarios. Such research is motivated by the fact, that hybrid functionalities of smaller
cardinality can be realized more easily using physical assumptions (see e.g. [GIS+10; CK88; Cré97]).

One of the earliest insights in that area comes from the seminal work of Kilian [Kil88] (which was later improved by Ishai, Prabhakaran, and Sahai [IPS08]), which works in the setting of Anonymous Abort (AA). In this setting, the adversary can abort the protocol anonymously – that is, without revealing information regarding corrupted parties – if the state is considered unfavorable. They proved that in this setting, actively secure n-party MPC is possible using only Oblivious Transfer (OT), which is of cardinality $2^n$.

In the setting of Universal Composability by Canetti [Can01], it was shown by Canetti and Fischlin [CF01], that without an additional n-party set-up (like a trusted third party, who creates a common reference string from a given distribution), general MPC cannot be achieved. They even proved that not even (MPC-incomplete) commitments can be securely realized in this setting.

Our work focuses on the setting of Identifiable Abort. This setting was first investigated (under the name Cheater Identification) by Ishai, Ostrovsky, and Seyalioglu [IOS12], although only in the honest majority setting. It turned out that even under such seemingly loose restrictions, no composition of functionalities with cardinality $2^n$ (such as OT) exists that leads to n-party MPC with cheater identification and therefore there is no canonical extension of the protocol by Kilian [Kil88], which adds identifiable abort using only OT hybrid functionalities.

The information-theoretic setting of IA was first investigated by Ishai, Ostrovsky, and Zikas [IOZ14]. There, the authors proved that the results of Ishai, Ostrovsky, and Seyalioglu [IOS12] do not hold if there is a correlated randomness setup; which doesn’t contradict the statement, since correlated randomness is considered a n-party functionality.

Combining these two results, we know that hybrid functionalities of cardinality $n$ suffice for n-party MPC with IA, whereas functionalities of cardinality $2^n$ do not. Inspired by the work of Fitzi, Garay, Maurer, and Ostrovsky [FGM+01], where the question was investigated without abort, this raises the question whether we can achieve n-party MPC with IA using only hybrid functionalities of cardinality $k$ with $2 < k < n$ and, more importantly, what the minimal cardinality $k^*$ for hybrid functionalities is, from which n-party MPC with IA can be build from. To the best of our knowledge, there is no previous work investigating this question in the setting of IA.

In contrast, most research in the area of Identifiable Abort has been directed towards the efficiency of concrete MPC protocols (such as [DPS+12; SF16; BOS16]). However, we focus on the theoretical minimal setup for general n-party MPC.

### 1.2 Contribution

Our main contributions are:

**New Oblivious Transfer variant.** In Section 3, we introduce a multi-party extension of OT called Fully Committed Oblivious Transfer (FCOT) which fits
well into the setting of Identifiable Abort. We extend OT to the multi-party case, such that all parties obtain a receipt if the OT has shown to be successful. The parties consist of one sender and one receiver, both of which have secret inputs as in the classical OT, and \((n-2)\) witnesses without any input. After the OT, both the sender and the receiver are committed to their inputs independently. They can unveil their inputs later on. More specifically, if the sender unveils its input \(m_0\) (resp. \(m_1\)), then all parties obtain \(m_0\) (resp. \(m_1\)) simultaneously. Note that the sender can unveil \(m_0\) and \(m_1\) independently of each other. If the receiver unveils its input, all parties obtain its choice bit \(c\). Note that the unveiling-step for both parties is purely optional.

In the appendix, we prove equivalence of Secure Function Evaluation and Fully Committed Oblivious Transfer in the setting of Identifiable Abort. That is, we show how to instantiate \(n\)-party SFE in the FCOT-hybrid-model and how to realize \(n\)-party FCOT using only a SFE hybrid functionality.

**Conflict Graph.** As a tool for our analysis, we introduce the conflict graph in Section 4. Although we only use it as a tool to prove security of our construction, we believe it to be of independent interest, like the improvement of efficient MPC-protocols with IA.

Formally, a conflict graph is an undirected graph \(G = (P, E)\), where \(P\) is the set of parties. An edge \(e = \{P, P'\} \in E\) for \(P, P' \in P\) and \(P \neq P'\) corresponds to a conflict, that is, an accusation of misconduct between the two parties \(P\) and \(P'\); an honest party \(P\) accuses a malicious party \(P'\), if either \(P'\) notably deviated from the protocol, or if \(P'\) aborted a hybrid functionality with IA in which \(P\) participated. In that case, we call \(P'\) a disruptor. Note that malicious parties can accuse any party of being a disruptor. Furthermore, we call the complement graph of the conflict graph association graph \(A = (P, E')\). Its edges \(\{P, P'\} \in E'\) indicate that parties \(P\) and \(P'\) are not (yet) in conflict with each other.

The conflict graph helps to identify disruptors. It models every parties knowledge on misbehavior of other parties. Using the conflict graph, protocols that realize ideal functionalities with IA can reduce complex properties of the protocol to easily verifiable graph conditions. Under certain conditions, the graph allows to extract an explanation of the conflicts. That is a subset of parties \(P' \subseteq P\) such that all conflicts can be explained assuming that these parties are corrupted. Technically, an explanation is a Vertex Cover of the conflict graph.

We can use ambiguous explanations to obtain information regarding disruptors. For example, if we know that the adversary can only corrupt up to \(t\) parties and a party \(P\) is part of every explanation \(P'\) of size \(|P'| \leq t\), then \(P\) is guaranteed to be a disruptor. We generally differentiate explanations as internal and external explanations. The main difference is that for internal explanations, a party takes into account that it knows it behaved according to the protocol, meaning that the party itself is honest. For external explanations, the conflict graph allows even parties who do not participate in the protocol to identify a corrupted party as a disruptor.

An example conflict graph can be found in Fig. 1: It showcases a 4-party MPC protocol with an adversary capable of corrupting at most \(t = 2\) parties.
The conflict graph has three different explanations (Fig. 1a, 1b and 1c). The graph does not provide external explanation: For each party $P_i$, there is at least one possible explanation where $P_i$ is honest.

Assuming that $P_3$ is honest, we can deduce that only the explanation from Fig. 1a can apply, implying that $P_2$ is honest, whereas $P_1$ and $P_4$ are disruptors. However, $P_3$ has no way of convincing $P_2$ of that fact, as for $P_2$, the explanations from Fig. 1b and Fig. 1c are equally possible; the latter explanation would imply $P_3$ to be a disruptor. Hence, the honest parties are unable to agree on a common disruptor in this scenario.

Both internal and external explanations of the conflict graph can be transferred into graph properties. We thus point out the necessary and sufficient conditions of a conflict graph to unambiguously identify a misbehaving party. We therefore introduce two terms: First, we define what it means for a graph to be $t$-settled, which is required for $G$ to provide an external explanation. A conflict graph $G$ is $t$-settled, iff (1) the minimal explanations have a size of at most $t$ and (2) the same party $P$ exists in every minimal explanation. If this property is fulfilled, even external parties can be convinced that the party $P$ is a disruptor.

For a formal treatment of internal explanations, we define the property of a biseparated conflict graph $G$. The property states that there is a subset of edges $E' \subseteq E$ such that $G' = (P, E')$ is a complete bipartite graph. This means that we can partition the graph in two subsets $P_1$ and $P_2$, such that (1) $P = P_1 \cup P_2$ and (2) all parties in $P \in P_1$ have an edge $\{P, P'\} \in E$ for every $P' \in P_2$ and vice versa. This property implies that any party can safely assume that all parties in the other partition are disruptors. However, external parties don’t necessarily know which partition belongs to the honest parties.

The conflict graph of an $n$-party computation is maintained and created by a functionality $\mathcal{I}^b_{CG}$. The functionality accepts as inputs messages of the type $(\text{conflict},P')$ from any party $P$. We call the chronologically ordered list of accusations the conflict graph transcript.

Based on this transcript, $\mathcal{I}^b_{CG}$ infers the conflict graph $G$. For each message $(\text{conflict},P')$ from any party $P$, $\mathcal{I}^b_{CG}$ adds a new edge $\{P, P'\}$ to $G$. After having created such an intermediate graph, $\mathcal{I}^b_{CG}$ performs a deduction step using an
algorithm called DeduceCG (see Algorithm 1). This algorithm adds new edges \( \{P, P'\} \) to \( G \), if the only logical conclusion that \( P \) can draw from \( G \) is that \( P' \) is corrupted, but no conflict between \( P \) and \( P' \) has been announced. A more formal description of DeduceCG can be found in Section 4.

**Constructing \( n \)-party SFE from SFE of lesser cardinality.** The main focus of our work is in Section 5, where we investigate the minimal cardinality of hybrid functionalities to realize \( n \)-party MPC with IA.

We investigate security against adversaries \( A \) who can corrupt all but three parties, that is, we set \( t \leq (n - 3) \). There, we use the conflict graph to show that it is theoretically impossible for the adversary to abort arbitrarily often, without causing disruptors to be identified by all honest parties. We then provide a construction, which expands \( (n - 1) \)-party SFE to \( n \)-party SFE with IA. In our setting, we show in the supplementary material (Appendix B), that \( n \)-party SFE is as powerful as \( n \)-party FCOT. Hence, our problem of instantiating \( n \)-party SFE from SFE-hybrids of cardinality \( (n - 1) \) comes down to instantiating \( n \)-party FCOT using only hybrid functionalities of \( (n - 1) \)-party FCOT.

In the first step, we show how a global commitment \( T_{\text{COM}}^v \) can be expanded, such that \( n \)-party commitments \( T_{\text{COM}}^v \) can be constructed from \( (n - 1) \)-party commitments \( T_{\text{COM}}^{v-1} \) and \( T_{\text{BC}} \). Essentially, the protocol lets the sender commit the same bit to each hybrid functionality (for each excluded party), and each recipient awaits the receipt from all hybrid functionalities it participates in, and then
broadcasts the receipt. After the opening phase, the parties accept, if the opening information received from the subfunctionalities are sufficiently consistent.

As a second step, we use the \( n \)-party commitment constructed in the first step to construct the actual expansion of FCOT from \( I_{FCOT}^{n-1} \) and \( I_{BC}^{n} \) to \( I_{FCOT}^{n} \).

In conclusion, we show that for any number of parties \( n \), we can build \( n \)-party SFE from \((n-1)\)-party SFE with IA. This expansion counts as our main contribution, which extends the graphic in Fig. 2 for all dishonest majority settings.

2 Setting

When constructing MPC-protocols, it is crucial to give a precise specification what its security guarantees are and under which circumstances they apply. In the following we give the relevant properties of the adversary and the composition model.

We consider statistical security, where the adversary is computationally unbounded. Moreover, our only assumption is the existence of hybrid functionalities. This leaves the means of the realization of these hybrid functionalities up to the user, e.g. via physical means such as trusted hardware [GIS+10; SSW10] or noisy channels [CK88; Cré97], or again from computational assumptions with better efficiency [Bon98; Reg05].

We focus on static adversaries that maliciously corrupt an arbitrary number of parties, as the honest majority case is historically well researched [BGW88].

We don’t explicitly assume additional secure channels between each pair of parties. The only means of communication between the protocol parties are via hybrid functionalities. Though, pairwise private secure channels can be realized by hybrid functionalities of cardinality \( \geq 2 \).

We focus our investigation in a synchronous network model, as our conflict graph requires that any conflict sent by a party \( P \) will eventually be received by all other parties. In an asynchronous model, the adversary could drop all messages [CM89; BCG93], thus getting something similar to Anonymous Abort. This would render Identifiable Abort essentially useless. In the synchronous model, however, the adversary can only either let the functionality terminate, or abort at the cost of unveiling the identity of at least one malicious party.

We further assume that the adversary can neither send a message to a functionality in the name of an honest party, nor alter or even read such a message. That is, communication between parties and ideal functionalities are authenticated.

In our setting, all parties have access to an \( n \)-party broadcast, which we model as ideal functionality \( I_{BC}^{n} \). This broadcast is mainly used to announce conflicts. For more details see Section 3.2.

2.1 Security Framework

For our analysis we consider simulation-based security [GMW87]. The central idea behind simulation-based security is that a real execution of a protocol is
compared to an idealized execution in which a trusted party exists which performs the computation. The behavior of this party is specified by a \textit{functionality} $F$. In the real world, a set of mutually distrustful parties execute a protocol $\pi$, which is said to \textit{realize} the functionality $F$, if it can be shown to be indistinguishable from the ideal world, in which the whole transcript is generated by a simulator who works agnostically from the parties inputs and the computation is performed by the honest party according to $F$. More precisely, the transcripts of both worlds must be indistinguishable. The transcript includes the output of all parties and the respective adversary. Indistinguishability implies that the real adversary cannot learn anything from the real protocol execution that the simulator cannot contrive without knowing the private inputs. We require statistical indistinguishability, i.e. the transcripts are indistinguishable even for computationally unbounded distinguishers.

We adapt the view that the direct party-party communication can also be modeled as a two-party hybrid functionality. Therefore, in our hybrid protocol, honest parties only ever communicate with hybrid functionalities in an authenticated manner; the adversary cannot manipulate messages from honest parties to hybrid functionalities. We work in a synchronous model, where we obtain Denial-of-Service (DoS)-protection for free.

We generally assume that the simulator gets notified whenever any party passes input to any functionality. The simulator doesn’t learn anything regarding the parties secret inputs. It only learns, that a party passed input to the functionality.

For the composition of functionalities we use a short notation.

\textbf{Notation 1 (Functionalties)} We call the number of interacting parties of a functionality the \textit{cardinality} $[FGM^{+}01]$ and denote it superscript, e.g. $F^n$.

\textbf{Definition 1 ((SFE-)Complete functionalities).} For $n,m \in \mathbb{N}_+$, we call a functionality $F^m$ of cardinality $m$ (SFE-)complete, iff there exists a $\{F^m\}$-hybrid protocol that securely realizes $F^n_{\text{SFE}}$ with the same abort property.

\textbf{Definition 2 (Minimal functionality).} We call a functionality $F^m$ with cardinality $m$ minimal, iff no functionality of lesser cardinality is complete.

\subsection{Identifiable Abort}

When constructing any protocol, an abort property must be specified. Intuitively, the most desirable property is \textit{guaranteed output}, where an abort is impossible. Unfortunately, guaranteed output requires a setup of full cardinality $[FGM^{+}01]$ when the number of corrupted parties and the computational power of the adversary is not limited. On the other extreme, the weaker notion of Anonymous Abort leaves the adversary capable of stopping any computation without repercussions and thus is an undesirable property for many real-world scenarios.

A protocol with IA enables all honest parties to eventually expel all malicious parties, if the protocol is aborted too often. Thereby it suffices that, during an unsuccessful protocol run, all honest parties agree on at least one disruptor. Then
the adversary can abort the protocol at most \((n - 1)\) times, before all malicious parties are excluded.

To clarify the abort property of a given functionality, we use the following notation:

**Notation 2 (Functionalities with IA)** A functionality \(\mathcal{I}^n\) is a functionality with Identifiable Abort and \(n\) participants.

Note that the original work [IOZ14] uses the notation \(\mathcal{F}_{\text{ID}}^\perp\).

**Definition 3 (Identifiable Abort).** Let \(\mathcal{I}^n\) be an ideal \(n\)-party functionality with parties \(P\) and malicious subset \(C \subseteq P\). \(\mathcal{I}^n\) has **Uni-Identifiable Abort**, iff all (honest) parties yield output \((\text{abort}, D)\) when the adversary sends \((\text{abort}, D)\) to \(\mathcal{I}^n\). If \(D \not\in C\), the message from the adversary is ignored. \(\mathcal{I}^n\) has **Multi-Identifiable Abort**, iff all (honest) parties yield output \((\text{abort}, C')\) when the adversary sends \((\text{abort}, C')\) to \(\mathcal{I}^n\). If \(C' \not\subseteq C\), the message is ignored.

Additional care has to be taken into the protocol design, if the protocol does not have fairness; the adversary may learn sensitive information in the first protocol run, which it can leverage in the next run. The honest parties neither learn their output, nor have a precise estimate, how sensitive the data of the adversary is. By design, our functionality is implicitly fair. Therefore, we can compose and repeat functionalities without excluded parties.

### 3 Functionalities

In this section, we introduce the ideal functionalities we use. We note that all following functionalities have the output property of Identifiable Abort.

#### 3.1 Secure Function Evaluation

First we give a formal variant of a Secure Function Evaluation (SFE)-functionality.

**Functionality \(\mathcal{T}^n_{\text{SFE}}\)**

\(\mathcal{T}^n_{\text{SFE}}\) proceeds as follows, running with security parameter \(\lambda\), parties \(P = \{P_1, \ldots, P_n\}\), input set \(\mathcal{I} \subseteq \{1, \ldots, n\}\), malicious parties \(C \subseteq P\), adversary \(S\) and function \(f: ((x_i)_{i \in I}) \rightarrow ((y_j)_{j \in \{1, \ldots, n\}}, \Delta)\) with private input \(x_i\) and output \(y_j\) for \(P_j\) and common output \(\Delta\). Messages not covered here are simply ignored.

- When receiving \((\text{input}, x_i)\) from \(P_i\) with \(x_i \in \{0, 1\}^\lambda\) and \(i \in I\), store \((i, x_i)\). Ignore further messages from \(P_i\).
- When there are \((i, x_i)\) in store for all \(i \in I\), then send \((\text{output}, y_j, \Delta)\) to each party \(P_j\) and \((\text{output})\) to \(S\), then terminate.

When receiving \((\text{abort}, C')\) from \(S\) with \(C' \subseteq C\), then \(\mathcal{T}^n_{\text{SFE}}\) outputs \((\text{abort}, C')\) to all parties, and then terminates.
Each party receives a private output $y_i$ and a common output $\Delta$. We additionally use an input set $I$ to model the fact, that not all parties have to provide input. Otherwise, certain functionalities such as broadcast cannot be realized because receiving parties, that do not provide input, may stall the protocol execution. This can be used to realize certain functionalities such as commitments using a function in SFE, where only the committer $C$ has to provide input, whereas the receiver $R$ only obtains output.

Although $T^\text{SFE}$ is not well-formed, it only requires knowledge about the corrupted parties upon abort. In particular, functionalities with IA need to know the set of malicious parties $C \subseteq P$, since it must check whether the set $C'$ in the abort is indeed a subset of $C$. In this sense, our functionalities are as well-formed as possible in the setting of IA.

### 3.2 Global Commitment and Broadcast

Global Commitment is the natural extension of two-party commitment, in which a party $C$ is committed towards $(n-1)$ other parties.

<table>
<thead>
<tr>
<th>Functionality $T^\text{COM}_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T^\text{COM}<em>n$ proceeds as follows, running with security parameter $\lambda$, parties $P = {C, R_1, \ldots, R</em>{n-1}}$, malicious parties $C \subseteq P$ and adversary $S$. Messages not covered here are ignored.</td>
</tr>
<tr>
<td>- When receiving $(\text{commit}, m)$ with $m \in {0, 1}$ from party $C$, store $m$ and send $(\text{receipt commit})$ to all parties and to $S$. Ignore further messages of the type $(\text{commit}, \cdot)$ from $C$.</td>
</tr>
<tr>
<td>- When receiving $(\text{unveil})$ from party $C$ and if $m$ is stored, send $(\text{unveil}, m)$ to all parties and to $S$, then terminate.</td>
</tr>
<tr>
<td>When receiving $(\text{abort}, C')$ from $S$ with $C' \subseteq C$, then $T^\text{COM}_n$ outputs $(\text{abort}, C')$ to all parties, and then terminates.</td>
</tr>
</tbody>
</table>

We also make use of an ideal broadcast functionality, which is a relatively weak building block:

<table>
<thead>
<tr>
<th>Functionality $T^\text{BC}_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T^\text{BC}<em>n$ proceeds as follows, running with security parameter $\lambda$ and parties $P = {S, R_1, \ldots, R</em>{n-1}}$, malicious parties $C \subseteq P$ and adversary $S$. Messages not covered here are simply ignored.</td>
</tr>
<tr>
<td>- When receiving a message $(\text{input}, m)$ with $m \in {0, 1}^*$ from party $S$, send $(\text{output}, m)$ to all parties and to $S$. Then terminate.</td>
</tr>
<tr>
<td>When receiving $(\text{abort}, C')$ from $S$ with $C' \subseteq C$, then $T^\text{BC}_n$ outputs $(\text{abort}, C')$ to all parties, and then terminates.</td>
</tr>
</tbody>
</table>
Although $I^n_{BC}$ is a relatively weak functionality, it suffices as sole setup to realize the conflict graph functionality $I^n_{CG}$ from Section 4. We provide an instantiation of $I^n_{CG}$ using $I^n_{BC}$ in our supplementary material (Appendix D), thus proving that $I^n_{CG}$ too is a relatively weak setup.

Note that a broadcast $I^n_{BC}$ can be constructed from a global commitment $I^n_{COM}$, by letting the sender $S$ commit to his message and then directly unveil it.

### 3.3 Fully Committed Oblivious Transfer

Next, we introduce our novel primitive called Fully Committed Oblivious Transfer (FCOT), which we use for our construction in Section 5:

<table>
<thead>
<tr>
<th>Functionality $I^n_{FCOT}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I^n_{FCOT}$ proceeds as follows, running with security parameter $\lambda$ and parties $P = {S, R, W_1, ..., W_{n-2}}$, malicious parties $C \subseteq P$ and adversary $S$. Messages not covered here are simply ignored.</td>
</tr>
<tr>
<td>• When receiving $(\text{messages}, m_0, m_1)$ from $S$ with $m_0, m_1 \in {0, 1}$, store $m_0, m_1$. Ignore further messages of the type $(\text{messages}, \cdot, \cdot)$ from $S$.</td>
</tr>
<tr>
<td>• When receiving $(\text{choice}, c)$ from $R$ with $c \in {0, 1}$, store $c$. Ignore further messages of the type $(\text{choice}, \cdot)$ from $R$.</td>
</tr>
<tr>
<td>• When both messages from $S$ and $R$ have been received, send $(\text{receipt transfer}, \bot)$ to $S$ and to all parties except $R$, and send $(\text{receipt transfer}, m_c)$ to $R$.</td>
</tr>
<tr>
<td>• When receiving $(\text{unveil message}, b)$ from $S$ with $b \in {0, 1}$ and $m_0, m_1$ are stored, send $(\text{unveil message}, b, m_b)$ to $S$ and to all parties. Ignore further messages $(\text{unveil message}, b)$ from $S$.</td>
</tr>
<tr>
<td>• When receiving $(\text{unveil choice}, c)$ from $R$ and $c$ is stored, send $(\text{unveil choice}, c)$ to $S$ and to all parties. Ignore further messages from $R$.</td>
</tr>
</tbody>
</table>

When receiving $(\text{abort}, C')$ from $S$ with $C' \subseteq C$, then $I^n_{FCOT}$ outputs $(\text{abort}, C')$ to all parties, and then terminates.

The idea is not completely new; Crépeau [Cré90] introduced a committed variant of OT under the name of Verifiable OT, which was later renamed to Committed Oblivious Transfer (COT) [CvT95]. There, the sender inputs two committed messages $(m_0, m_1)$, and the receiver inputs the choice bit $c$. The receiver obtains the committed $m_c$, and can use this for zero-knowledge proofs, without knowing $m_1-c$ and without revealing $c$ in the process.

In Fully Committed Oblivious Transfer (FCOT), the sender $S$ is committed to both messages $m_0$ and $m_1$, and the receiver $R$ is committed to $c$. Our ideal functionality $I^n_{FCOT}$ is a novel extension of conventional OT, which has proven to be very useful in the setting of Identifiable Abort.

This $n$-party extension of OT is motivated by the insight that the results of [Kil88; IPS08] do not work with IA. In our supplementary material (Appendix B),
we show that FCOT is SFE-complete with IA. The proof is based on the construction from [IPS08], but replaces (2-party) OT-calls with (n-party) FCOT-calls. When a party notices any misbehavior, it can demand other parties to open their inputs, thus enabling all parties to retrace the disruptor’s misbehavior without leaking any information on the parties inputs due to their secret sharing.

4 Conflict Graph

Our goal is to construct protocols, which either output the correct value, or identify at least one malicious party. As an innovative and useful tool to help us in that regard, we introduce the conflict graph. It administrates observed misbehavior during the execution and helps honest parties to agree on a misbehaving party. Honest parties behave according to the protocol specification, hence the correctness property states that only corrupted parties can be identified. We call actively deviating parties disruptors. If a functionality \( I \) is aborted with output \( \text{abort}, P_i \), the corrupted party \( P_i \) is a disruptor. It is then a known fact to all other participants of \( I \), that \( P_i \) is malicious. We also refer to disruptors as parties who notably deviate from the protocol description; this, however, is protocol-dependent. Consider, for example, a protocol that specifies that each party initially sends the message \( \text{OK} \) via \( F^2_{\text{AUTH}} \) to all other parties. If party \( P_j \) receives \( \perp \) from \( P_i \) via \( F^2_{\text{AUTH}} \), it is clear to \( P_j \) that \( P_i \) must be malicious.

In the statistical setting, a party \( P_i \) can interact with the conflict graph only by publicly announcing that it witnessed a misbehavior of party \( P_j \). We call such an announcement a declaration of a conflict between \( P_i \) and \( P_j \). Note that malicious parties can declare a conflict with anybody. If a conflict is based on an Identifiable Abort, conflicts between two honest parties can never occur by definition of IA. However, conflicts based on deviations of the protocol can have many different reasons; it is vital to ensure during protocol design, that no conflicts between two honest parties can occur due to protocol deviations.

A visual example of a conflict graph is given in Fig. 1. It showcases the uncertainty that arises in such scenarios. The view of \( P_1 \) can only be explained, if \( P_3 \) is corrupted. Yet \( P_1 \) cannot expect \( P_2 \) to agree on that, since the view of \( P_2 \) could also be explained if \( P_1 \) and \( P_4 \) are corrupted. The precise condition of a conflict declaration based on protocol deviations is highly protocol-dependent; we defer the specifics of our constructions to Section 5.

We now provide an informal description of the conflict graph and introduce graph-based properties that enable identification of a disruptor. We then introduce an ideal functionality \( I^\text{CG}_n \) to manage the conflict graph; this functionality can be realized using only the broadcast functionality \( I^\text{BC}_n \), which is a relatively weak assumption. With \( I^\text{CG}_n \), protocols only have to take care of conflict declarations; all logic associated with the formal deduction of the conflict graph is part of the ideal functionality.

For a set of parties \( P = \{P_1, \ldots, P_n\} \), we model each conflict between two parties \( P_i \) and \( P_j \) as undirected edge \( \{P_i, P_j\} \in E \); a conflict between \( P_i \) and \( P_j \) trivially also causes a conflict between \( P_j \) and \( P_i \), hence directed edges would
contain a lot of redundancy. We define the graph \( G := (P, E) \) for parties \( P \in P \) and conflicts \( \{P_i, P_j\} \in E \) as the conflict graph. We also denote the set of all possible edges by \( \bar{E} := \{\{P, P'\} \mid P, P' \in P \land P \neq P'\} \). Honest parties can use this conflict graph to identify disruptors, once sufficiently may conflicts have been declared. To formalize this idea, we introduce the term explanation of a conflict graph. Intuitively, an explanation is a vertex cover of \( G \); that is, a subset \( P' \subseteq P \) of parties which explains all declared conflicts, if we assume that all \( P \in P' \) are corrupted. Let \( t \) be the maximum number of corrupted parties. If a party \( P \) exists, which is in all explanations \( P' \) with \( |P'| \leq t \), \( P \) is identified as disruptor.\(^5\) More formally, even non-participants can identify disruptors in a conflict graph \( G \), if \( G \) is \( t \)-settled.

**Definition 4 \((t\text{-settled conflict graph})\).** Let \( t \) for \( 0 \leq t \leq n \) be the maximum number of corrupted parties. Let \( G = (P, E) \) be the conflict graph of a protocol \( \pi \). Let

\[
M(G, t) := \{P' \mid P' \subseteq P \text{ is a MVC of } G : |P'| \leq t\}
\]

\(^5\) Note that the same condition also holds for a set of parties \( P' \subseteq P \).
be the set of all Minimal Vertex Covers (MVCs) of $G$ with size $t$ or less.

$$G \text{ is } t\text{-settled} \iff M(G, t) \neq \emptyset \land X(G, t) \neq \emptyset.$$  \hspace{1cm} (1)

That is, $G$ is $t$-settled, if all MVCs of $M(G, t)$ contain the same party $P$. We need the additional condition $M(G, t) \neq \emptyset$, as the intersection of zero sets $\bigcap_{P' \in \emptyset} P' = \emptyset$ is the universe by definition. Furthermore, we call $X(G, t)$ the settled set of $G$ with up to $t$ malicious parties.

In Fig. 3a, we provide a graph which is 1-settled; the only explanation for $t = 1$ implies $\{P_1\}$ to be corrupted. In Fig. 3d, the smallest possible explanation is of size 2. However, even if we assume $t = 2$, it is impossible for external observers to determine which parties are malicious. Yet, any participant has only one explanation of size 2, since it knows it is honest. This motivates a new condition on $G$, which allows participants to detect disruptors:

**Definition 5 (Biseparated conflict graph).** A conflict graph $G = (P, E)$ is called biseparated, iff a subset of edges forms a complete bipartite graph (biclique) on $P$

$$\exists E' \subseteq E: G' = (P, E') \text{ is a complete bipartite graph}.$$  \hspace{1cm} (2)

Figure 3b shows an example conflict graph which is biseparated; the edges that form the bipartite graph are thickened. It partitions $P$ into $\{P_1, P_2\}$ and $\{P_3, P_4, P_5\}$. Figure 3e does not contain any bipartite graph over all parties, as $P_2$ is not in conflict with $P_5$ and hence would not be in the same partition as $P_1$.

Finally, we provide a definition for a requirement on $G$ to detect disruptors:

**Definition 6 (t-semisettled conflict graph).** Let $0 \leq t \leq n$ be the maximum number of corrupted parties. Let $G$ be a conflict graph. $G$ is called t-semisettled, iff $G$ is $t$-settled or $G$ is biseparated.

None of the $t$-settled-property and biseparated-property automatically imply each other. However, we now show under which conditions $t$-settled graphs imply biseparated graphs:

**Lemma 1.** Let $t \geq (n - 2)$. If $G$ is a $t$-settled graph, $G$ is biseparated.

**Proof.** If $G$ is $t$-settled, then there can only be either one or two honest parties. We first consider the case where one honest party $P \in P$ is in conflict with all $P' \in P \setminus P$, that is, $X(G, (n - 2)) = \{P\}$. The conflict graph is trivially biseparated on $X(G, (n - 2))$ and $P \setminus X(G, (n - 2))$.

Next, we consider the case where two or more parties are honest: $P_0, P_1 \in X(G, (n - 2))$. Assume for the sake of contradiction, that $G = (P, E)$ is not biseparated. This implies that there is at least one pair of parties $\{P, P'\}$, such that party $P \in \{P_0, P_1\}$ and party $P' \in P \setminus \{P_0, P_1\}$ have no conflict. Therefore, the vertex cover given by $P' := (X(G, (n - 2)) \setminus P) \cup ((P \setminus X(G, (n - 2))) \setminus P') = P \setminus \{P, P'\}$ serves as a viable explanation of cardinality $\leq (n - 2)$ for the conflict graph. In particular, we have $P \notin P'$, and, as a contradiction, we get $P \notin X(G, (n - 2))$.

Thus, for $t \geq (n - 2)$, the $t$-semisettled condition is equivalent to the biseparation of the graph. \hfill $\Box$
Now that we have established the conditions on $G$ relevant for IA, we need a mechanism to manage the conflict graph. We define a functionality for this purpose:

**Functionality $T^\lambda_{CG}$**

$T^\lambda_{CG}$ proceeds as follows, running with security parameter $\lambda$ and parties $P = \{R_1, \ldots, R_n\}$, malicious parties $C \subseteq P$ and adversary $S$. Messages not covered here are ignored.

- Upon first activation, initiate the set of conflict edges $E := \emptyset \subseteq \tilde{E}$.
- When receiving a message $(\text{conflict}, P_i)$ from $P_j$, append the new conflict edge $\{P_i, P_j\}$ to the set of conflict edges $E$ and send $(\text{conflict}, P_j, P_i)$ to the adversary.
- When receiving a message $(\text{query})$ from $P_i$, deduce the current conflict graph $G^* := \text{DeduceCG}(P, E, t)$ and output $G^*$ to $P_i$.

When receiving $(\text{abort}, C')$ from $S$ with $C' \subseteq C$, then $T^\lambda_{CG}$ outputs $(\text{abort}, C')$ to all parties, and then terminates.

As stated in Section 3.2, we emphasize that this functionality can be realized using only one instance of an $n$-party broadcast functionality. The detailed construction can be found in our supplementary material (Appendix D).

We call $G^*$ the *deduced conflict graph*. It remains to provide the algorithm DeduceCG, which we first motivate:

Consider Fig. 3c with $t = 3$ and take the position of $P_1$. This graph is neither 3-settled nor biseparated. Using the inference rule of DeduceCG, a biseparated graph can be deduced. In this stage $P_1$ has already identified its neighbors $N(P_1) = \{P_5, P_6\}$ as malicious. Thus the rest of the graph must be explicable with only $t - |N(P_1)| = 1$ malicious party. Hence, from $P_1$’s viewpoint, $P_4$ must also be malicious, since this is the only explanation for the remaining conflict graph, resulting in the graph from Fig. 3f.

The algorithm DeduceCG from Algorithm 1 applies a special inference rule for each party. DeduceCG takes the position of each party $P \in P$ and removes $P$ and all neighbors $P'$ with $\{P, P'\} \in E$ from the graph. If the remaining graph is $(t - |N(P)|)$-settled, then all conflicts between $P$ and the settled set of the remaining graph are appended to the conflict edges. This rule mirrors what party $P$ can infer from its own knowledge about the corruption of its neighbors.

Next, we present the main result regarding our concept of the conflict graph, stating that for any protocol $\pi$ that implements a functionality with IA, a deduced $t$-semisettled conflict graph $G^*$ is necessary and sufficient upon abort.

**Lemma 2 (t-semisettled Identifiable Abort).** Let $n \geq 3$ and $m \in \mathbb{N}$. Let $t$ with $0 \leq t \leq n$ be the maximum number of corrupted parties. Let $\pi$ be a protocol that securely realizes a functionality $T^\pi$ with Identifiable Abort in a $\{T_t^\pi, T^\lambda_{CG}\}$-hybrid model. Upon abort, the abort outputs of all honest parties must be equal; if
Algorithm 1 DeduceCG\((P, E, t)\)

1: \(\text{changed} := 1\)
2: while \(\text{changed} = 1\) do
3: \(\text{changed} := 0\) \(\triangleright\) no change in this iteration yet
4: for all \(P \in P\) do
5: \(\tilde{P} \coloneqq P \setminus (\{P\} \cup N(P))\) \(\triangleright\) reduced party set
6: if \(\tilde{P}\) is \((t - |N(P)|)\)-settled then
7: \(P_f := X((\tilde{P}, E \setminus 2^{\tilde{P}}), t - |N(P)|)\) \(\triangleright\) settled set from Definition 4
8: for \(P' \in P \times 1\) do
9: \(E := E \cup \{P, P'\}\) \(\triangleright\) append inferred conflicts
10: \(\text{changed} := 1\) \(\triangleright\) mark change
11: end for
12: end if
13: end for
14: end while
15: return \(G^* := (P, E)\) \(\triangleright\) deduced conflict graph

the conflict graph \(G\) of \(\pi\) is not \(t\)-semisettled upon abort, then \(\pi\) cannot securely 
realize \(I^n\).

Proof. We carry out our proof for Uni-Identifiable Abort, however, the same reasoning applies for Multi-Identifiable Abort. We prove that a conflict graph that is not \(t\)-semisettled must lead to incorrect abort outputs against some environment \(Z\). Together with the fact that no simulator can abort either \(I^m\) or \(I^n\) with an honest party, this directly contradicts the presupposition that the protocol \(\pi\) securely realizes \(I^n\).

Let \(\pi\) be a protocol that realizes some functionality \(I^n\) with parties \(P = \{P_1, \ldots, P_n\}\). Also, let the conflict graph \(G\) of \(\pi\) be not \(t\)-semisettled upon abort; that is, \(G\) is neither \(t\)-settled nor biseparated. For the sake of contradiction, assume that there is a selector function with which honest parties can still agree on a common disruptor \(P^*\). We show that this agreement cannot exist by constructing a different environment creating the same conflict graph transcript, but where \(P^*\) is honest.

Let \(S_\pi : P \times \tau \to P\) with \((P, \tau) \mapsto P'\) for \(P' \subseteq P\) be the selector function that specifies the identified disruptor for each (honest) party \(P\), given the ordered set of inputs to \(T^G_{\text{CG}}\) as \(\tau\). For the sake of simplicity, we focus here on the special case \(|P'| = 1\), that is, \(S_\pi\) outputs a single party \(P'\) (Uni-Identifiable Abort). However, the proof still holds if \(S_\pi\) outputs \(P'\) with \(|P'| > 1\). The selector \(S_\pi\) must only depend on the transcript of all conflicts and the identity of the given party itself. Otherwise, the environment \(Z\) could induce two different abort-parties for two honest parties.

We now show that for a special Environment \(Z^0\) there exists a different environment \(Z^1\) where the same transcript would lead to an accusation of an honest party. Let the first environment \(Z^0\) corrupt up to \(t\) parties \(C^0 \in M(G, t)\). Denote the complement set of honest parties by \(H^0 := P \setminus C^0\). Environment \(Z^0\) produces a transcript \(\tau^0\) until the protocol aborts with output \((\text{abort}, P^*)\), where
\( P^* \in S_\pi(P, \tau^0) \) for each \( P \in H^0 \). The environment \( Z^0 \) only lets corrupted parties broadcast conflicts as a retaliation, that is, it broadcasts conflicts \((\text{conflict}, P)\) against an honest party \( P \) in the name of \( P' \) only after \( P \) has publicly declared a conflict \((\text{conflict}, P')\).

We now show that if there exists another corrupted MVC \( C^1 = P \setminus H^1 \in M(G, t) \) with \( C^0 \cup C^1 \neq P \), or equivalently \( H^0 \cap H^1 \neq \emptyset \), and \( P^* \in S_\pi(P, \tau^0) \subseteq C^0 \cap H^1 \) for all \( P \in H^0 \), then \( \pi \) cannot securely realize \( I_n \). Here the corrupted MVC \( C^1 \) is chosen such that the identified party against \( Z^0 \) is explicitly excluded. Because there exists a party \( P \) in \( H^0 \setminus H^1 \), it will select \( P^* \in S_\pi(P, \tau^0) = S_\pi(P, \tau^1) \) against both environments, though \( P^* \) is honest when playing with \( Z^1 \). Specifically, there exists an honest party \( P \in H^0 \cap H^1 \) which selects \( S_\pi(P, \tau^0) \) against \( Z^0 \) but selects \( S_\pi(P, \tau^1) \subset H^1 \) against \( Z^1 \). Since both transcripts \( \tau^0 \) and \( \tau^1 \) are equal, the two selected parties must also be equal. Fig. 4 depicts an exemplary visualization of the corruption sets.

This leads to the contradiction that against the environment \( Z^1 \) an honest party is identified by at least one honest party. Thus, we know that if the above conditions hold, that is, there exists a MVC \( C^1 \) with \( H^0 \cap H^1 \neq \emptyset \) and \( P^* \in C^0 \cap H^1 \), then \( \pi \) cannot securely realize \( I_n \).

Next, we show that such an MVC actually exists. It only does not exist if for all feasible explanations \( C^1 \in M(G, t) \), it holds that \( C^0 \cup C^1 = P \), or if for all \( P \in H^0 \), it holds that \( S_\pi(P, \tau^0) \subseteq H^0 \cup C^1 \). The former is equivalent to \( C^1 \supseteq H^0 \). This, however, contradicts the initial assumption that \( G \) is not \( t \)-semisettled. If \( C^1 = H^0 = P \setminus C^0 \) are both MVCs, this would imply a biseparated conflict graph with partitions \( C^0 \) and \( C^1 \). If \( C^1 \supseteq H^0 \), it would follow that \( H^0 \) is a valid explanation, that is, \( H^0 \in M(G, t) \). Hence, we get that \( C^1 \) is not minimal and thus is not contained in \( M(G, t) \). Both cases lead to a contradiction with the initial assumption.

**Fig. 4**: Visualization of an exemplary set of corrupted parties and an assumed honest parties agreement on \( P^* \) for an environment \( Z^0 \) on the left. The right side shows another, specifically constructed set of corrupted parties by an environment \( Z^1 \) which leads to honest-honest-accusations despite having the same conflict graph transcript.
Now that we have shown that there exists $C^1 \in \mathcal{M}(G,t)$ that fulfills $C^0 \cup C^1 \neq P$ and $S_e(P,\tau^1) \subseteq C^0 \cap H^1$ for all $P \in H^0$, we can define an environment $Z^1$ that corrupts $C^1$ and acts such that it produces $\tau^1$ before the abort.

We conclude our proof by showing that $Z^1$ really can create an equivalent transcript to $\tau^0$. The transcript $\tau^0$ of the conflict graph $G = (P,E)$ is essentially an ordered list of (directed) edges $(e_j)_{j=1..m}$. To keep the information which party broadcasts the conflict, the edges in the transcript are directed, while the edges in the conflict graph are undirected. This mirrors the fact that the process of the conflict graph creation yields more information than the final conflict graph. Regardless of the environment, each edge of the transcript $e_j = (u_j,v_j)$ contains at least one corrupted party. When a conflict $e_j$ arises in the protocol execution with $Z^0$, the environment $Z^1$ behaves as follows:

**If** $u_j \in C^0$ **and** $u_j \in C^1$, $Z^0$ and $Z^1$ behave identically.

**If** $u_j \in C^0$ **and** $u_j \in H^1$, then by assumption, $Z^0$ will only declare conflict $e_j$ as a retaliation, that is, after $(v_j,u_j)$ has been issued, to match the behavior of an honest party. Therefore $Z^1$ lets $v_j$ broadcast $(\text{conflict},u_j)$ such that the retaliation $e_j$ follows subsequently from the honest party $u_j$.

**If** $u_j \in H^0$ **and** $u_j \in C^1$, $Z^1$ lets $u_j$ broadcast $(\text{conflict},v_j)$. If $v_j$ is honest, it will retaliate; if it is not, $Z^1$ lets $v_j$ broadcast the retaliation $(\text{conflict},u_j)$.

**If** $u_j \in H^0$ **and** $u_j \in H^1$, $Z^0$ and $Z^1$ behave identically.

The transcript produced by $Z^1$ is therefore identical to $\tau^0$, which concludes our contradiction. □

We use this result in our later proofs, where it now suffices to show that a certain type of protocol never establishes a $t$-semisettled graph and does not produce the correct output on termination. No simulator can then abort the functionality such that two honest parties yield different identified parties. Again, we remark that our result finds application in any simulation-based framework, e.g. standalone or universally composable.

For a $I^*_{GC}$-hybrid protocol, we require all honest parties to declare conflict with the disruptor of any hybrid functionality when it is aborted. Thus, any aborted hybrid functionality $I^k$ running with the subset $P' \subseteq P$ with $|P'| = k$, the subgraph on $P'$ becomes $t$-semisettled. Intuitively, each abort of a hybrid functionality (new subset of parties) adds new conflicts until at least one disruptor can be identified.

We call the complement graph of the conflict graph the *Association Graph* $A = (P,E')$. The edges $E'$ are defined as follows:

$$\{P,P'\} \in E' \iff \{P,P'\} \in \bar{E} \land \{P,P'\} \notin E$$

That is, $E'$ contains all possible edges between any two parties $P$ and $P'$, which do not have an edge in the conflict graph $G$.

**Remark 1.** A conflict graph $G = (P,E)$ is biseparated, iff its association graph is disconnected.

This directly follows from the fact that the complement of a biclique contains no path from one partition to the other.
In this chapter, we present the main result of our work: we show how $n$-party MPC with IA can be based on $(n-1)$-party MPC with IA and an $n$-party broadcast, if at least three parties are honest. To that end, we utilize our novel tool, the conflict graph (see Section 4), which enables honest parties to share information about malicious parties. These requirements are relatively mild, since broadcast is easily physically realizable. Previous results [GMW87; BGW88] already fail against a constant fraction of malicious parties. Our result can tolerate almost all parties being malicious. Still, it is an interesting question whether our results can be extended to the case where all parties might be corrupted ($0 \leq t \leq n$).

More formally, we construct $I_{FCOT}^n$ from $I_{FCOT}^{n-1}$ and $I_{BC}^n$. Since FCOT is MPC-complete, this corresponds to an expansion of general MPC.

Our proof is structured in three lemmata. Each provides a limit on the maximum number of hybrid subfunctionalities that can be aborted. This implies a guarantee that some subfunctionalities cannot be aborted. Using this guarantee, we provide a protocol that uses $n$-party broadcast to expand $(n-1)$-party global commitments $I_{COM}^{n-1}$ to $n$-party global commitments $I_{COM}^n$. We then use this $n$-party commitment as a tool in the expansion of FCOT from $(n-1)$ parties to $n$ parties.

We note that our result holds for any simulation-based framework that offers at least parallel composition, and hence has applicability even in the Universal Composability Framework [Can01]. Intuitively, if the smaller functionalities can be composed parallel, then the constructed $n$-party functionality can be composed in the same fashion.

Lemma 3 (General subfunctionality abort). Let $t$ for $0 \leq t < n$ be the maximum number of parties the adversary can corrupt. Let $\pi$ be an $\{I^{n-1}\}$-hybrid protocol that uses the conflict graph technique ($I_{CG}^n$). If the adversary $A$ aborts more than $t$ subfunctionality instances of cardinality $(n-1)$, then the conflict graph from $I_{CG}^n$ is biseparated.

Proof. Denote the set of $n$ parties by $P = \{P_1, \ldots, P_n\}$. Let $t' \leq t$ be the actual number of parties corrupted by the adversary $A$. W.l.o.g., let $H = \{P_1, P_2, \ldots, P_{n-t}\}$ be the set of honest parties and let $C = \{P_{n-t+1}, \ldots, P_n\}$ be the set of corrupted parties.

We now show that, if $A$ aborts too many subfunctionalities, a set of parties must be separated from all others in the association graph, thus leaving the corresponding conflict graph biseparated. Note that the association graph is initially a complete graph and loses edges, when subfunctionalities are aborted. Since honest parties are never in conflict, their mutual edges are never removed. We only consider subfunctionalities of cardinality $(n-1)$; thus, we have one subfunctionality instance for each excluded party $P \in P$.

---

6 For simplicity, neglect the fact that edges could also be removed, if a party obviously deviates from the protocol.
We call the set of corrupted parties that aborts a subfunctionality instance the disruptor set $D$. By $D_i$ we denote the disruptor set for the subfunctionality instance that excludes $P_i$, and by $Z$ the set of corrupted parties that disrupted no subfunctionality so far. The disruptor sets corresponding to honest parties $D_1, \ldots, D_{n-t'}$ alongside with $Z$ partitions $C$, meaning that $D_1 \cup \cdots \cup D_{n-t'} \cup Z = C$. $Z$ is disjoint with any $D_i$. If there was a nonempty intersection between two different disruptor sets $D_i$ and $D_j$, then this intersection $I_{ij} := D_i \cap D_j$ would be in conflict with all parties who participated in $I_{n-1}^{P}$ and with the ones who participated in $I_{n-1}^{P_j}$, which is all parties. Thus, $I_{ij}$ would be completely separated in the association graph. Consequently, the association graph loses all edges except the ones between $Z$ and $H$ and for each $i \in [n-t']$, the ones between $P_i$ and $D_i$. The respective sets internally form a complete graph. See the left side of Fig. 5 for the case $t' = (n-3)$. Until now, only functionalities that omitted honest parties were aborted.

Recall that the association graph must be connected in order for its complement graph not to be biseparated (Remark 1). Therefore, any subfunctionality $I_{n-1}^{P}$ for any omitted party $P \in D_i \cup Z$ can only be aborted by other parties within the same set. Otherwise, it would be disconnected from its own set $D_i$ or $Z$, respectively. Since it already is disconnected from $H$, $Z$ and all other $D_j$, it would be completely isolated in the association graph.

Consider, for example, the case where $D' \subseteq D_1 \setminus \{P\}$ disrupts $I_{n-1}^{P}$ for $P \in D_1$. This causes all remaining parties $P' \in D_1 \setminus (\{P\} \cup D')$ to declare a conflict with $D'$. Thus, those parties would be separated from $D'$. Since $P$ was excluded from this subfunctionality, it remains connected to both $D_1$ and $D'$. This turns the association graph into a tree containing subsets of parties as nodes, where $H$ is the root node containing all honest parties. The tree has one leaf that contains all non-disruptors $Z$ and $(n-t')$ branches, one for each honest party. Again, both $H$ and $Z$ internally form a complete graph. The abort of any $I_{n-1}^{P}$ for $P \in D_i$ for an arbitrary $i$ thus leads to an extension of the respective branch. However,
the adversary cannot abort the subfunctionality corresponding to the leaf node of each branch; there would have to be some party left that could abort the subfunctionality, which would then cause a conflict, thus separating a subset of the branch from the rest. As a consequence, at least \((n - t')\) subfunctionalities must succeed. The adversary can thus abort at most \(t'\) subfunctionalities, before the conflict graph becomes biseparated.

We consider two special cases: one where we have at least three honest parties \( ((t \leq (n - 3)) \) and one where the adversary can corrupt all but one parties \( ((t \leq (n - 1))\).

In the former case, where \( (t \leq (n - 3))\), only strictly less than \(t\) subfunctionalities of cardinality \((n - 1)\) can be aborted when using the conflict graph technique. For this case, Lemma 3 tightens to:

**Lemma 4 (Strong subfunctionality abort).** Let \( t \leq (n - 3)\). If \(t\) or more subfunctionalities of cardinality \((n - 1)\) are aborted, then the conflict graph is biseparated.

**Proof.** It suffices to consider the case for \( t\) aborted subfunctionalities. Assume that \( t\) subfunctionalities are aborted. Each of the \( t\) disruptors \( D'\) can have at most two associates, namely the party \( P\) who was omitted in the functionality \( T^{n-1}_P\) that \( D'\) aborted and the party \( P^*\) that disrupted \( T^{n-1}_{D'}\). However, honest parties \( P\) can still have up to three associates, namely two actual honest parties and one disruptor who aborted \( T^{n-1}_{D'}\). Honest parties can use this fact to determine its malicious associate: at least \((n - t - 1)\) of its associates form a clique, while the malicious associate has only two associates of its own. Thus, the honest parties can declare a conflict with the party outside of their clique. This leaves the conflict graph such that all parties who are part of the clique in the association graph form one partition, whereas all disruptors form the second partition. Thus, the conflict graph is biseparated.

\(\Box\)

Next, we consider the second case, namely \( (t \leq (n - 1))\). Here, Lemma 3 yields:

**Lemma 5 (Weak subfunctionality abort).** Let \( 0 \leq t < n\). Either \((n - 2)\) or less subfunctionalities of cardinality \((n - 1)\) are aborted, or the conflict graph is biseparated.

**Proof.** We only proof the case of a single honest party \( P\). If more parties are honest, less subfunctionalities can be aborted.

Denote the set of parties by \( P = \{P_1, \ldots, P_n\}\); w.l.o.g., assume \( P = P_1\). Furthermore, denote the corrupted parties as \( C\).

Assume \( I^{n-1}_{P_1}\) is aborted. Then the subgraph on \( P \setminus \{P_1\}\) must be \(t\)-semisettled. Otherwise, \( P_1\) can directly identify all other parties as malicious. If the subgraph is \(t\)-settled, the excluded party \( P_1\) also identifies the settled set and the entire graph is biseparated on the settled set and its complement. If the subgraph on \( P \setminus \{P_1\}\) biseparated, then there are two partitions, \( P_0\) and \( P_1\), which are completely disconnected from each other in the association graph. However, \( P_1\)
is still connected to both partitions. Each party in \( P_0 \) can only be aborted by a subset of \( P_0 \), since otherwise, a set of disruptors in \( P_0 \) of a functionality in \( P_1 \) would be in conflict with all other parties; the same argument holds for \( P_1 \). Every time a functionality omitting a party in any partition is aborted, that party is removed from the respective set, thereby shrinking the set. Hence, each set must have at least one party \( P \), whose hybrid functionality \( \mathcal{I}_P \) cannot be aborted, without creating a biseparated conflict graph.

We now have two limits on the maximal number of hybrid functionalities that the adversary can abort before causing a biseparated conflict graph. Using them, we are able to expand the (SFE-incomplete) functionality Global Commitment (GCOM) from \( \mathcal{I}_\text{COM}^{n-1} \) to \( \mathcal{I}_\text{COM}^n \).

**Lemma 6 (COM expansion).** There exists a protocol \( \pi \) in the \( \{ \mathcal{I}_\text{COM}^{n-1}, \mathcal{I}_\text{BC}^n \} \)-hybrid model that securely realizes \( \mathcal{I}_\text{COM}^n \) for \( 0 \leq t \leq (n-2) \) static malicious corruptions:

\[
\exists \pi : \pi \mathcal{I}_\text{COM}^{n-1} \mathcal{I}_\text{BC}^n \geq \mathcal{I}_\text{COM}^n
\]  

Proof. We only consider bit-commitments, which can be canonically extended to string-commitment. Let the set of parties be \( P = \{ C, R_1, ..., R_{n-1} \} \) and let \( b \) be the bit that the committer \( C \) commits to. We present a protocol that uses \( (n-1) \) COM-subfunctionality instances \( \mathcal{I}_R^{n-1} \) for each excluded receiver \( R_l \). Additionally, it uses the conflict graph \( \mathcal{I}_{\text{CG}}^n \) which can be realized using \( \mathcal{I}_{\text{BC}}^n \). Denote \( C \)'s input to \( \mathcal{I}_R^{n-1} \) by \( b_l \) and the abort set for \( \mathcal{I}_R^{n-1} \) by \( D_l \). Let \( I_0 \) be the set of subfunctionalities where \( b_l = 0 \), let \( I_1 \) be the set of subfunctionalities where \( b_l = 1 \), and let \( I_x \) be the set of subfunctionalities that have previously been aborted. Note that \( \mathcal{I}_{\text{CG}} \) ensures that \( I_x \) is public knowledge; the abort of a subfunctionality \( \mathcal{I}_{R_l}^{n-1} \) results in a \( t \)-semisettled subgraph on \( P \setminus \{ R_l \} \) and if a subgraph is \( t \)-semisettled, the corresponding subfunctionality is considered aborted. It holds that \( |I_0| + |I_1| + |I_x| = (n-1) \), since there are \( (n-1) \) instances of \( \mathcal{I}_P^{n-1} \).

In the following we give a description of the protocol. The appropriate simulator can be found in our supplementary material (Appendix C.1).

Let \( b \) be the bit \( C \) wants to commit to. For runtime restrictions, we assume that the unary security parameter \( 1^k \) is also input.

**Protocol:**

1. On input \( \text{(commit,} b \text{)} \) with \( b \in \{0, 1\} \) from \( Z \) to \( C \), \( C \) sends \( \text{(commit,} b \text{)} \) to \( \mathcal{I}_R^{n-1} \) for all \( l \in \{1, ..., n-1\} \).
2. Once \( R_l \) has received output from any \( \mathcal{I}_R^{n-1} \) for \( j \neq l \), the receiver \( R_l \) checks \( |I_x| \):
   - **If** \( t \leq n-3 \) and \( |I_x| \geq t \), then the conflict graph is biseparated due to Lemma 4. \( R_l \) aborts with the identified set of parties.
   - **If** \( |I_x| \geq n-1 \), then the conflict graph is biseparated due to Lemma 5. \( R_l \) aborts with the identified set of parties.
3. On input \( \text{(unveil)} \) from \( Z \) to \( C \), \( C \) sends \( \text{(unveil)} \) to \( \mathcal{I}_R^{n-1} \) for all \( l \in \{1, ..., n-1\} \).
4. Once $R_l$ has received output from all $T_{R_j}^{n-1}$ for $j \neq l$, $R_l$ sends $(\text{receipt, } R_l)$ to $T_{BC}^n$.

5. Once $R_l$ has received $(\text{receipt, } R_j)$ from all $R_j$ for $j \neq l$ via $T_{BC}^n$, $R_l$ checks $|I_x|$. 

   - If $|I_x| \geq n - 1$, then the conflict graph is biseparated due to Lemma 5. $R_l$ aborts with the identified set of parties.
   - If $|I_x| = n - 2$, excluded receiver $R_l$, i.e. $T_{R_l}^{n-1}$ has been aborted, concurs with the output of their (only) associate $P$. $R_l$ does so by broadcasting (help) whereupon associate $P$ broadcasts its output. If the associate is the committer, the committer broadcasts its bit $b$, otherwise the associate is a receiver who checks all of its unveiled bits for consistency. If all but one bit are equal, then this bit $b$ is the output (unveil, $b$). The associate receiver broadcasts (unveil, $b$) to signal to $R_l$ to output the same. If inconsistent bits have been received, then $P$ broadcasts (abort, $C$) and outputs the same.
   - If $|I_x| = n - 3$, essentially the same strategy as in the previous case applies. Only, here are up to three honest parties possible, therefore excluded receivers only concur with associates which have at least three associates themselves. These associates are naturally included in all subfunctionalities that have not been aborted.
   - If $|I_x| \leq n - 4$, $R_l$ checks the unveil messages:
     Since $|I_x| \leq n - 4$ each receiver gets at least three messages; there is also $T_{C}^{n-1}$ which is not used. If all but one unveilings are consistent with $b_{R_l}$, $R_l$ outputs (unveil, $b'_{R_l}$). Otherwise, $R_l$ outputs ($\bot$) and sends {conflict, $C$} to $T_{CG}^n$.

The intuition behind the protocol is the following: The committer $C$ commits its bit $b$ to all subfunctionalities, which gives honest receivers consistent opening information. The adversary has two levers to disturb the protocol: One is to abort subfunctionalities, thus increasing $|I_x|$; we have shown in Lemma 4, that this is only possible for up to $t$ aborts, before the conflict graph becomes biseparated. The other option the adversary has is to let a corrupted committer use different bits in different subfunctionalities. If at least four subfunctionalities are not aborted, then the second adversarial strategy no longer works, since all receivers will notice a sufficient inconsistency in the subfunctionalities. Therefore the adversary has to abort many subfunctionalities. This increases the honest parties knowledge about the identity of malicious parties sufficiently for honest parties to identify each other. Finally, if too many subfunctionality are aborted, such that only one is left, the conflict graph has sufficiently many edges that a identification is possible. If a receiver does not accept, one of two cases must have happened: Both in Item 2 and Item 5, $R$ rejects if $|I_x| \geq t$. In that case, it follows from Lemma 4 and Lemma 5 that the resulting conflict graph is biseparated, hence identification for an abort is possible. If $t \leq (n - 3)$ Lemma 4 applies directly, whereas if $t \geq (n - 2)$ and $|I_x| > t$ Lemma 5 applies.
We note that our strategy does not work directly for $t \leq (n - 1)$ because the honest party has no other honest party to rely on, if too many subfunctionalities are aborted.

Proving security of this protocol is straightforward; we provide a simulator and a proof in our supplementary material (Appendix C.1).

Further on, we concentrate on FCOT. Denote the parties as sender $S$, receiver $R$ and witnesses $W_1$ through $W_{n-2}$. We call type-1 subfunctionalities $I_{W_i}^{n-1}$ for all $i \in \{1, \ldots, n-2\}$ and type-2 subfunctionalities both $I_{S}^{n-1}$ and $I_{R}^{n-1}$.

**Corollary 1.** Let $\pi$ be a protocol that securely realizes $I_{FCOT}^n$. At least two type-1 functionalities cannot be aborted, otherwise the conflict graph is $t$-semisettled.

This follows from Lemma 4 with $t = (n - 3)$, which states that only strictly less than $(n - 3)$ subfunctionalities of cardinality $(n - 1)$ can be aborted without producing a biseparated conflict graph. Conversely, at least four subfunctionalities of cardinality $(n - 1)$ must succeed, two of which may be $I_{S}^{n-1}$ and $I_{R}^{n-1}$. Hence the remaining two must be of type-1.

Now, we use Corollary 1 to construct a protocol that securely realizes $I_{FCOT}^n$ from $I_{FCOT}^{n-1}$ and $I_{COM}^n$.

**Theorem 3 (FCOT expansion).** There is a protocol $\pi^n_{FCOT}$ that statistically realizes $I_{FCOT}^n$ in the $\{I_{FCOT}^{n-1}, I_{BC}^n\}$-hybrid model against any adversary that statically corrupts at most $t \leq (n - 3)$ parties: \( \exists \pi : \pi^n_{FCOT} \geq I_{FCOT}^n \) for $t \leq (n - 3)$.

**Proof.** We describe the protocol $\pi^n_{FCOT}$. Denote the parties as sender $S$, receiver $R$ and witnesses $W_1$ through $W_{n-2}$.

The protocol is iteration-based. Let $I_x$ be the set of type-1 subfunctionalities that have been aborted. There are $T \coloneqq (n - 2 - |I_x|)$ type-1 subfunctionalities that have not yet been aborted. In each iteration of the protocol enumerate these as $(I_1^{n-1}, \ldots, I_T^{n-1})$. The protocol uses $r$ sessions of each remaining subfunctionality $I_i^{n-1} \notin I_x$. We implicitly use the string variant of the commitment functionality, which can be constructed from the aforementioned bit commitment. We consider a subfunctionality to be aborted, if the conflict graph of its participants is $t$-semisettled. Note that $I_{BC}^n$ follows trivially from a $T_{COM}^n$ by immediately unveiling the commitment. Furthermore, we showed in Lemma 6, that $I_{COM}^{n-1}$ follows from $I_{FCOT}^{n-1}$. In our supplementary material, we show in Appendix E that $I_{COM}^{n-1}$ follows from $I_{FCOT}^{n-1}$. Hence, our construction works in the $\{I_{FCOT}^{n-1}, I_{BC}^n\}$-hybrid model; we still use the terms $I_{COM}^{n-1}$ and $I_{CG}$ in our proofs.

The central idea of the protocol is to use secret sharings of the sender’s messages to perform multiple FCOTs of cardinality $(n - 1)$ and to globally commit to these shares. We use a cut-and-choose trick, where some shares are unveiled to ensure that the FCOTs and Commitments are equal. After the Oblivious Transfer, the receiver commits to the received shares. The secret sharing implies that he cannot unveil a different choice bit afterwards. We assume inputs
The security proof is given in our supplementary material (Appendix C.2).

Protocol iteration:
1. If the conflict graph becomes $t$-semisettled, all honest parties output the appropriate corrupted set.
2. On input $\langle$messages, $m_0, m_1 \rangle$ with $m_0, m_1 \in \{0, 1\}$ from $Z$ to $S$, $S$ creates an additive $T$-sharing of $m_0$ and $m_1$ called $(\mu^0_j)_{j=1..T}$ and $(\mu^1_j)_{j=1..T}$, respectively. $S$ then creates an $r$-sharing $(\alpha^0_{j,i})_{i=1..r}$ for each share $\mu^0_j$ with a $(2r/3)$-threshold secret sharing scheme. $S$ inputs $(\text{messages}, \alpha^0_{j,i}, \alpha^1_{j,i})$ into the $i$-th session of $I^{n-1}_j$ for all $i \in [r]$ and $j \in [T]$ and commits globally to each share by sending $\alpha^0_{j,i}$ resp. $\alpha^1_{j,i}$ into $I^{n}_{\text{COM}}$ for each $i \in [r]$ and $j \in [T]$.
3. On input $\langle$choice, $c \rangle$ with $c \in \{0, 1\}$ from $Z$ to $R$, $R$ sends $\langle$choice, $c \rangle$ to all remaining subfunctionalities $I^{n-1}_i$ for all $l \in [T]$. Additionally, $R$ sends $c$ to $I^{n}_{\text{COM}}$.
4. If subfunctionality $I^{n-1}_l \notin I_x$ is aborted, the current iteration ends and $I^{n-1}_l$ is added to $I_x$.
5. When a party $P$ has received $\langle$receipt messages\rangle resp. $\langle$receipt choice\rangle from all $r$ sessions of all type-1 FCOT-subfunctionalities that include $P$, $P$ outputs $\langle$receipt messages\rangle resp. $\langle$receipt choice\rangle.
6. When no subfunctionality has been aborted in this iteration and all $T$ subfunctionalities have concluded their OT-phase with output $\langle$receipt transfer, $\bot$\rangle to $S$, $R$ learns $m_c$; it obtains all $r$ shares $\alpha^0_{j,i}$ of all $T$ additive shares $\mu^0_j$ of $m_c$. Each $W_i$ and $R$ verify the integrity of the sender’s commitments, by verifying that the sender’s global commitments indeed contain the correct input used for the FCOT-subfunctionalities. Each party $P \in P \setminus \{S\}$ broadcasts the set of $r/10n$ indices $i \in [r]$ per subfunctionality $l \in [T]$. For each index $(i, l)$, the sender has to unveil the corresponding global commitment and the FCOT-subfunctionality.
7. Upon receiving $\langle$unveil message, $b\rangle$ from $Z$ to $S$, $S$ sends $\langle$unveil\rangle to all $I^{n}_{\text{COM}}$. Upon receiving their shares, the parties output $\langle$unveil message, $b, m_b\rangle$.
8. Upon receiving $\langle$unveil choice\rangle from $Z$ to $R$, $R$ inputs $\langle$unveil choice\rangle into all FCOT-functionalities and $\langle$unveil\rangle into $I^n_{\text{COM}}$ to unveil $c$. The other parties first check consistency of all unveiled choice bits. If not consistent, all honest parties immediately abort with $\langle$abort, $R\rangle$. If the choice bits are consistent $c'$, then the other parties check consistency between the unveiled choice bits of the FCOT-sessions and the global commitment. If internally inconsistent or inconsistent with the global commitment, the affected FCOT is considered aborted, since all participating party are able to identify the receiver as malicious.

The security proof is given in our supplementary material (Appendix C.2). $\square$

Intuitively, security follows from the fact that each FCOT-subfunctionality unveils less than $r/10$ shares, but reconstruction requires $2r/3$ shares. A user can learn at most $11r/10$ shares, whereas $4r/3$ would be required to learn both messages. If $S$ tries to unveil a different value than its original input, it would have to deviate in more than $r/3$ shares of any one subfunctionality. For each subfunctionality, a
party can probe \( r/10n \) shares. Thus, \( S \) has negligible probability of successfully deviating from the original FCOT-input in the required number \( r/3 \) of global commitments.

The integrity check in the OT-phase ensures that the values must match the FCOT-inputs with overwhelming probability. From Corollary 1, it follows that at least for two \( \mathcal{I}_{t}^{n-1} \not\in I_{x} \), all \( r \) sessions must be unveiled. Also, \( R \) unveils the commitments to its received shares. If \( R \) tries to learn both messages, \( R \) to have input \( (1-c) \) into all sessions of at least two FCOT-subfunctionalities; in that case, \( R \) cannot learn \( m_{c} \), since \( (\mu_{j}^{c})_{j \in \{T\}} \) is an additive sharing and thus requires all shares for reconstruction.

**Corollary 2 (SFE expansion).** The functionality \( \mathcal{I}_{SFE}^{n} \) can be statistically realized in the \( \{\mathcal{I}_{SFE}^{n-1}, \mathcal{I}_{BC}^{n}\} \)-hybrid model against any static adversary that corrupts at most \( t \leq (n-3) \) parties: \( t \leq (n-3) \implies \exists \pi: \mathcal{I}_{SFE}^{n-1}, \mathcal{I}_{BC}^{n} \geq \mathcal{I}_{SFE}^{n} \).

If we additionally assume a multicast functionality were the recipient set can be chosen by the sender, then we can tighten our result for any \( t \leq n-3 \). By induction we can deduce an upper bound of the minimal complete cardinality for each number of maximal corruptions; as we have shown \( \mathcal{I}_{SFE}^{n-2} \) and \( \mathcal{I}_{BC}^{n} \) realize \( \mathcal{I}_{SFE}^{n-1} \) for \( t \leq n-3 \). But this also holds for \( t \leq n-4 \), then we have that \( \mathcal{I}_{SFE}^{n-3} \) and \( \mathcal{I}_{BC}^{n} \) realizes \( \mathcal{I}_{SFE}^{n-2} \). By induction we get \( \mathcal{I}_{SFE}^{i+2} \) and all \( \mathcal{I}_{BC}^{i} \) for \( i \in \{t+3, \ldots, n\} \) realize \( \mathcal{I}_{SFE}^{n} \).

**Corollary 3.** For up to \( t \) corruptions, it holds for the minimal complete cardinality that \( k^{*}(t) \leq t + 2 \).

It is an interesting insight that composing functionalities is possible when three parties are honest. Intuitively, this is the case because in each subfunctionality, we now have a guarantee that at least two parties are honest such that upon abort they share the same information about the disruptors.

### 6 Summary and Outlook

**Summary.** We brought the work of [FGM+01] into the framework of Identifiable Abort, and extended Uni-Identifiable Abort [IOZ14] to Multi-Identifiable Abort. For our analysis, we developed the formal concept of the conflict graph and its counterpart, the association graph, both of which only require a global broadcast \( (\mathcal{I}_{BC}^{n}) \). Then we linked identification criteria for disruptors in Identifiable Abort to easily verifiable properties of the conflict graph. Furthermore, we presented a new variant of Oblivious Transfer, namely Fully Committed Oblivious Transfer. Our main result proves a new upper bound for the minimal complete cardinality \( k^{*} \) against adversaries who can corrupt at most \( t \leq (n-3) \) parties. To that end, we provided a construction for realizing \( \mathcal{I}_{SFE}^{n} \) from \( \mathcal{I}_{SFE}^{n-1} \) and \( \mathcal{I}_{BC}^{n} \). In a first step we expanded global commitments from the \((n-1)\)-party case to \( n \) parties and then used the result to expand COT from \((n-1)\) to \( n \) parties. Our expansion results, together with the fact that COT and SFE are equally powerful, and that \( \mathcal{I}_{COM}^{n} \) can be instantiated via \( \mathcal{I}_{COT}^{n} \), implies the expansion of SFE in the presence of \( \mathcal{I}_{BC}^{n} \).
Outlook. Even the impossibility result of [IOS12] might not hold when a broadcast is available, that is, functionalities of cardinality 2 might suffice for MPC with Identifiable Abort. Also, Fig. 2 shows a discontinuity at $t = n/2$, where the upper limit jumps from 3 to approximately $n/2$. This could, however, be explained by the omission of the majority vote technique.

Our upper bounds require a full multicast $\{T_{BC}^{t+3}, \ldots, T_{BC}^{n-1}, T_{BC}^n\}$. It is interesting to see whether these multicasts of lesser cardinality are indeed necessary, or if they can be incorporated in the global broadcast $T_{BC}^n$ in some manner.

Also, our results require $n$-party broadcast. We did not find out, if a broadcast expansion is possible in the setting of IA. This would leave us with a pure SFE-expansion; we have shown, that $T_{FCOT}^{n-1}$ realizes $T_{COM}^{n-1}$, and that $T_{COM}^{n-1}$ realizes $T_{BC}^{n-1}$. Although we conjecture this to be impossible, realizing $T_{FCOT}^n$ from $T_{FCOT}^{n-1}$ and $T_{BC}^n$ alone might still be possible.

We do not know yet whether the minimal complete cardinality increases, when no global broadcast is used. Specifically the COM-expansion in Lemma 6 strengthens our intuition that an SFE-incomplete functionality like global broadcast indeed lowers the minimal complete cardinality.

Also, our construction merely provides a possibility result. We did not engage in efficiency analysis, although our protocols have polynomial runtime. Hence, the investigation of tightness of our results provides an open field of research.

Another branch of research is to lower the minimal cardinality by using quantum functionalities, as quantum states naturally enable for example the receiver of a message to transmit that message without obtaining any information about it. This property could be used to send messages via multiple smaller functionalities to obtain the behavior of a larger one. Specifically, we assume that given a pairwise authenticated secure quantum channel, $T_{COM}^n$ is SFE-complete with IA. A positive result in this area would imply efficient MPC-protocols, as the global commitment can be instantiated with a computationally secure commitment and a broadcast channel.

We only considered a deterministic notion of Identifiable Abort. We did not investigate an abort property like probabilistic IA, where a disrupter is identified with a certain probability $p$, or where a non-disruptor might be identified as disruptor with a low probability $p'$.

References


A Definitions and Notation

In general the following notation is used unless explicitly stated otherwise: For an overview see Table 1.

<table>
<thead>
<tr>
<th>Entity</th>
<th>Remark</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set</td>
<td>cursive</td>
<td>$P, H, C$</td>
</tr>
<tr>
<td>Party</td>
<td>sans-serif</td>
<td>$P, S, R, C, W$</td>
</tr>
<tr>
<td>Adversary</td>
<td>calligraphic</td>
<td>$A, S, Z$</td>
</tr>
<tr>
<td>Code words</td>
<td>typewriter</td>
<td>$\text{abort, conflict}$</td>
</tr>
<tr>
<td>Draw</td>
<td>uniformly from set</td>
<td>$v \overset{$}{\leftarrow} S, b \overset{$}{\leftarrow} {0,1}$</td>
</tr>
<tr>
<td>Functionality</td>
<td>with cardinality</td>
<td>$F^n, F^2_{OT}$</td>
</tr>
<tr>
<td>Functionality</td>
<td>with identifiable abort</td>
<td>$I^n, I^3_{BC}$</td>
</tr>
<tr>
<td>Construction</td>
<td>of functionalities</td>
<td>$I^n_{COM} \rightsquigarrow I^0_{BC}$ (see Section 2.1)</td>
</tr>
<tr>
<td>Parameter</td>
<td>Minimal cardinality</td>
<td>$k^c$</td>
</tr>
<tr>
<td>Parameter</td>
<td>Security Parameter</td>
<td>$\lambda$</td>
</tr>
<tr>
<td>Parameter</td>
<td>Maximal number of corrupted parties</td>
<td>$t$</td>
</tr>
<tr>
<td>Parameter</td>
<td>Number of Parties</td>
<td>$n$</td>
</tr>
</tbody>
</table>

Furthermore, we use the following conventions; also see the list of abbreviation and symbols.

**Notation 4** For any natural number $n \in \mathbb{N}$, we denote by $[n]$ all natural numbers between 1 and $n$, that is,

$$[n] := \{1, 2, \ldots, (n-1), n\}$$

**Definition 7 (Negligible functions).** A function $f: \mathbb{N} \rightarrow \mathbb{R}$ is called negligible, iff $f \in o(x^c)$ for all $c \in \mathbb{R}$. We later use the equivalent condition

$$\lim_{x \rightarrow \infty} \frac{\ln|f(x)|}{\ln x} = -\infty.$$ Denote the set of negligible functions with respect to $x$ by $\text{negl}(x)$.

**Definition 8 (Overwhelming functions).** A function $f: \mathbb{N} \rightarrow \mathbb{R}$ is called overwhelming, iff $1 - f$ is negligible. Denote the set of overwhelming functions with respect to $x$ by $\text{owhl}(x)$.

**Notation 5 (Construction)** We write $I^n_A, I^n_B \rightsquigarrow I^n_C$, iff there is a protocol $\pi^n_C$, that realizes $I^n_C$ in the $(I^n_A, I^n_B)$-hybrid-model. More formally:

$$I^n_A, I^n_B \rightsquigarrow I^n_C \iff \exists \pi^n_{AC, AB}: I^n_A, I^n_B \geq I^n_C$$
Usually the index is the security parameter $\lambda$, it is omitted when clear from the context.

**Definition 9 (Minimal complete cardinality).** Let $n \in \mathbb{N}$ be the number of parties and let $t \leq n$ be the maximal number of corrupted parties. We denote the cardinality of a minimal and complete functionality by $k^*(n; t)$. When clear from the context, we omit the parameters $n$ and $t$. In other words, a minimal complete functionality of cardinality $m'$ is the smallest functionality from which $n$-party SFE can be constructed.

**Definition 10 (Minimal vertex cover).** A Minimal Vertex Cover (MVC) is a vertex cover such that any strict subset is not a vertex cover.

Note the difference between a vertex cover and a minimal vertex cover. The minimal vertex cover is the vertex cover with the least number of vertices out of all possible vertex covers.

We make use of secret sharing in our constructions. Thus, we include a brief description for consistency.

**Notation 6 (Secret sharing (informal))** Let $p$ be a prime integer and $k, m \in \mathbb{N}$ integers with $k \leq m \leq p$. For a secret $s \in \mathbb{Z}_p$ a secret sharing $(\sigma_i)_{i=1}^m$ is a $(k,m)$-threshold scheme, iff $k$ or more shares reconstruct the secret $s$ but $k-1$ or less shares hide the secret $s$ information-theoretically. A prominent example is Shamir’s secret sharing [Sha79].

Also, particularly efficient are additive sharings with are always $(m,m)$-threshold schemes. Here, shares are simply uniformly distributed numbers from $\mathbb{Z}_p$ and such that $s = \bigoplus_{i=1}^m \sigma_i$.

Note that $(k,m)$-threshold schemes can tolerate up to $m - k$ manipulated shares and still reconstruct the secret correctly.

**Notation 7 (Boolean random variable)** For a boolean random variable $B$ which naturally takes values 0 and 1, we denote the probabilities as $\Pr[B] := \Pr[B = 1]$ and $\Pr[\neg B] := \Pr[B = 0]$.

## B SFE-Completeness of FCOT

In this section, we prove that Fully Committed Oblivious Transfer (FCOT) and Secure Function Evaluation (SFE) are equally powerful in the setting of IA, meaning that they can be constructed from each other. This implies that constructing $n$-party SFE from $(n-1)$-party SFE and an $n$-party broadcast comes down to the more intuitive construction of $n$-party FCOT from $(n-1)$-party FCOT and a broadcast.

**Lemma 7 (SFE $\rightsquigarrow$ FCOT).** There is a protocol $\pi_{\text{SFE}}^n$ in the $\{T_{\text{FCOT}}^n\}$-hybrid model that securely realizes $T_{\text{SFE}}^n$. 

30
Proof. Let \( n \) be the number of parties. We denote the set of all parties by \( P = \{ S, R, W_1, \ldots, W_{n-2} \} \). Further on, we use a seamless type conversion from algebraic numbers to bit strings when necessary, in the form of \( \mathbb{Z}_2^n \approx \{0,1\}^n \).

We present a protocol that lets the sender create many secret sharings of its inputs. The receiver \( R \) obtains sufficiently many shares to reconstruct one message, but not enough to reconstruct both. The witnesses \( W_i \) obtain sufficiently many shares to detect a manipulation of the shares in the unveiling-phase, but not enough to learn any message just from the OT-phase.

First, we describe the protocol \( \tau_{\text{FCOT}}^n \) that utilizes four calls to \( \mathcal{I}_{\text{SFE}}^n \). The first instance, \( \mathcal{I}_{\text{SFE}}^n[f] \), is used for the OT. The next two instances \( \mathcal{I}_{\text{SFE}}^n[g^0] \) and \( \mathcal{I}_{\text{SFE}}^n[g^1] \) are used for the unveiling of the sender’s message \( m_0 \) and \( m_1 \). The last instance \( \mathcal{I}_{\text{SFE}}^n[g_R] \) is used for the unveiling of the receiver’s choice bit. The functions are defined as follows:

\[
\begin{align*}
  f : (x_S, x_R, x_0, \ldots, x_{n-3}) &\mapsto (y_S, y_R, y_0, \ldots, y_{n-3}, \Delta) \\
  g^0_S : (u^0_S) &\mapsto (\varepsilon, \Delta' = u^0_S) \\
  g^1_S : (u^1_S) &\mapsto (\varepsilon, \Delta'' = u^1_S) \\
  gr : (vr) &\mapsto (\varepsilon, \Delta = vr)
\end{align*}
\]

for both \( b \in \{0,1\} \). The function \( f \) provides private outputs for each party and public output \( \Delta \). The inputs and outputs are defined as follows:

\[
\begin{align*}
  x_S &:= \left( \left( \mu^0_{j,i}, \mu^1_{j,i} \right)_{j=1..n,i=1..r} \right)_{i=1..r} & y_S &:= \left( \gamma v^0_{j,i} \right)_{i=1..r} \\
  x_R &:= \left( \left( \gamma_{j,i} \right)_{j=1..n,i=1..r} \right) \nu^0_R & y_R &:= \left( m_0, \left( \mu^0_{j,i}, \mu^1_{j,i} \right)_{i=1..r} \right) \\
  x_l &:= \left( \nu^1_i \right)_{i=1..r} & y_l &:= \left( \mu^0_{j,i}, \mu^1_{j,i}, \gamma v^1_i \right)_{i=1..r} \\
  u^0_S &:= \left( \mu^0_{j,i} \right)_{j=1..n,i=1..r} & \Delta_S &:= (\Delta_S, \Delta_R) \\
  v_R &:= \left( \gamma_{j,i} \right)_{j=1..n,i=1..r} & \Delta_R &:= (\Delta_S, \Delta_R)
\end{align*}
\]

For \( i \in [r] \), the sharings \( \bigoplus_{j=1}^{n} \mu^0_{j,i} := m_0 \) and \( \bigoplus_{j=1}^{n} \mu^1_{j,i} := m_1 \) distribute the two messages \( m_0 \) and \( m_1 \), and the sharing \( \bigoplus_{j=1}^{n} \gamma_{j,i} := c \) distributes the choice bit \( c \). We later chose the number of sharings \( r \) such that the detection of a share alteration by all honest parties becomes overwhelming. The common output \( \Delta_S \) takes the value 1, if the encoded bits of all r sharings of \( m_0 \) are equal \( (m_0 = \bigoplus_{j=1}^{n} \mu^0_{j,i} \text{ for all } i \in [r]) \) and all encoded bits of the sharings of \( m_1 \) are equal. Otherwise \( \Delta_S \) is 0. Analogously, \( \Delta_R \) is 1, if the sharings of \( c \) are consistent and 0 otherwise. Formally, the shares are bits \( \mu^0_{j,i}, \mu^1_{j,i}, \gamma_{j,i} \in \{0,1\} \) and the share choices are numbers \( \nu^i \in \mathbb{Z}_n \). Now that we have the definitions of the SFE-subfunctionality we proceed to describe the actual protocol. We assume inputs \( S(m_0, m_1), R(c) \) and \( W_i(\varepsilon) \) together with the implicit unary security parameter \( 1^\lambda \).
Protocol:

On input (local start) from $Z$ to $W_l$, $W_l$ draws $r$ numbers $\{\nu_i\}_{i \in [r]} \leftarrow \mathbb{Z}_n^r$, to determine which share of the other parties’ inputs will be obtained by $W_l$.

On input (messages, $m_0, m_1$) from $Z$ to $S$, $S$ produces $r$ independent, additive $n$-sharings of $m_0$ and $m_1$: $(\mu^0_{j,i}, \mu^1_{j,i})_{j \in [n], i \in [r]} \leftarrow \mathbb{Z}_n^{r \times n}$ such that for all $i \in [r]$ it holds that $\bigoplus_{j \in [n]} \mu^0_{j,i} = m_0$ and $\bigoplus_{j \in [n]} \mu^1_{j,i} = m_1$, where $j$ is the index of the share within a sharing and $i$ is the index of the sharing itself. Then, $S$ draws $r$ numbers $\{\nu^i_S\}_{i \in [r]} \leftarrow \mathbb{Z}_n^r$, which determine the shares of the receiver’s choice bit $S$ obtains. $S$ inputs $x_S$ into $I^n_{SFE}[f]$.

On input (choice, $c$) from $Z$ to $R$, $R$ produces $r$ independent, additive $n$-sharings of $c$: $(\gamma^i_{j,i})_{j \in [n], i \in [r]} \leftarrow \mathbb{Z}_n^{r \times n}$ such that for all $i \in [r]$ it holds that $\bigoplus_{j \in [n]} \gamma_{j,i} = c$, where $j$ is the index of the share within a sharing and $i$ is the index of the sharing itself. Then, $R$ draws $r$ numbers $\{\nu^i_R\}_{i \in [r]} \leftarrow \mathbb{Z}_n^r$, which determine the shares of the sender’s input $R$ obtains. $R$ inputs $x_R$ into $I^n_{SFE}[f]$.

On output (output, $y_R, \Delta$) from $I^n_{SFE}[f]$ to $R$, $R$ outputs (receipt transfer, $m_c$).

On output (output, $y_P, \Delta$) from $I^n_{SFE}[f]$ to $P$, where $P \in P \setminus R$, party $P$ outputs (receipt transfer, $\perp$).

On input (unveil message, $b$) from $Z$ to $S$, if (messages, $m_0, m_1$) has been received, $S$ inputs the previously generated $u^b_S$ into $I^n_{SFE}[g_S^b]$.

On output $u^b_S$ from $I^n_{SFE}[g_S^b]$ to $P$, where $P \in P \setminus S$, $P$ checks the consistency with all previously obtained shares. If all shares match and are consistent, i.e. across all $i \in [r]$ the sharings encode the same bit, $P$ outputs (unveil message, $b, m_b$). Otherwise, $P$ outputs (abort, $S$).

On input (unveil choice) from $Z$ to $R$, if (choice, $c$) has been received, $R$ inputs $u_R$ into $I^n_{SFE}[g_R^b]$.

On output $u_R$ from $I^n_{SFE}[g_R]$ to $P$, where $P \in P \setminus R$, $P$ checks the consistency with their previously obtained shares. If all shares match and are consistent across $i \in [r]$, $P$ outputs (unveil choice, $c$). Otherwise, $P$ outputs (abort, $R$).

We now give a description of the simulator. At the onset of the simulation, the simulator follows the program of all uncorrupted witnesses on input (local start) and computes their input $x_i$ according to the protocol and inputs it into the simulated $I^n_{SFE}$ in the name of $W_l$.

On input (receipt messages, $\perp$) from $I^n_{PCOT}$, the simulator gives local input to all uncorrupted simulated parties as follows: The simulator gives the simulated $S$ local input (messages, $0, 0$). Thus, $S$ computes $x_S$ according to the protocol and sends it to $I^n_{SFE}[f]$.

The simulator gives the simulated $R$ local input (choice, $0$). Thus, $R$ computes $x_R$ according to the protocol and sends it to $I^n_{SFE}[f]$.

On output (receipt output) from $I^n_{SFE}[f]$, the simulator sends output (receipt output) to $Z$. 

32
If the common output of \( T^\mu_{\text{SFE}}[f] \) contains \( \Delta_S = 0 \) or \( \Delta_R = 0 \), the simulator aborts \( T^\mu_{\text{FCOT}} \) with output \((\text{abort}, C')\) where \( S \in C' \) or \( R \in C' \).

**On output (unveil message, \( b, m_b \)) from \( T^\mu_{\text{FCOT}} \), the simulator fabricates input \( u^b_S \) for the uncorrupted simulated \( S \) to send as input into \( T^\mu_{\text{SFE}}[g^b_S] \). The simulator knows the indices of all shares of \( \mu^0 \) and \( \mu^1 \) that have been chosen by all other parties in the OT-phase. Either \( S \) learned the choice indices from a corrupted party as local input or it generated the choice indices itself for the uncorrupted simulated parties. Because each sharing has \( n \) additive shares, there is at least one index \( \nu^i \in \mathbb{Z}_n \) for each \( i \in \{1, \ldots, r\} \) whose share is known by no party (recall that only \((n - 1)\) shares per sharing have been distributed). Hence, the simulator flips the bits at the correct index for each \( i \in [r] \) if necessary, that is, if \( m_b = 1 \); recall that \( \mu^0 \) and \( \mu^1 \) encode 0. Denote the manipulated sharings by \( \mu^0 \) and \( \mu^1 \); they encode \( m_0 \) resp. \( m_1 \). Finally, the simulator lets \( S \) input \( u^b_S \) (computed from \( \mu^b \)) into \( T^\mu_{\text{SFE}}[g^b_S] \).

Because the manipulations of the sharings are chosen specifically such that no party yields a share that is manipulated, no party will notice the equivocation and accept the unveiling by outputting \((\text{unveil message}, b, m_b)\).

**On output (unveil choice, \( c \)) from \( T^\mu_{\text{FCOT}} \), the simulator fabricates input \( u_R \) for the uncorrupted simulated \( R \) to input into \( T^\mu_{\text{SFE}}[g_R] \). Here, the simulator proceeds completely analogous to the previous case of the unveiling of the sender’s messages. The same argumentation regarding the acceptance of the fabricated sharing applies.

**On input (\( \text{abort}, C' \)) for \( T^\mu_{\text{SFE}}[\cdot] \), the simulator aborts \( T^\mu_{\text{SFE}}[\cdot] \) as well as \( T^\mu_{\text{FCOT}} \) with \((\text{abort}, C')\).

Finally, we describe the simulator’s behavior when handling messages from and to malicious parties. In general, messages between hybrid functionalities and the environment are forwarded by the simulator. For simplicity, we assume \( r \) to be an uneven number.

**On input (input, \( x_S \)) from corrupted \( S \) to \( T^\mu_{\text{SFE}}[f] \), the simulator lets \( S \) input \((\text{messages}, m_0, m_1)\) into \( T^\mu_{\text{FCOT}} \), where \( m_0 \) and \( m_1 \) are the respective majority of the \( r \) encoded bits \( m^{b,i} = \bigoplus_{j=1}^r \mu^b_{j,i} \). The input \((\text{input}, x_S)\) is naturally passed on to \( T^\mu_{\text{SFE}}[f] \).

**On input (input, \( x_R \)) from corrupted \( R \) to \( T^\mu_{\text{SFE}}[f] \), the simulator lets \( R \) input \((\text{choice}, c)\) into \( T^\mu_{\text{FCOT}} \), where \( c \) is the majority of the encoded bits \( c' = \bigoplus_{j=1}^r \gamma_{j,i} \). The input \((\text{input}, x_R)\) is naturally passed on to \( T^\mu_{\text{SFE}}[f] \).

**On input (input, \( u^b_S \)) from corrupted \( S \) to \( T^\mu_{\text{SFE}}[g^b_S] \), the simulator checks consistency of the sharings in \( u^b_S \) with the sharings in \( x_S \). If they are consistent, then the simulator lets \( S \) input \((\text{unveil message}, b, m_b)\) into \( T^\mu_{\text{FCOT}} \), otherwise the simulator aborts \( T^\mu_{\text{FCOT}} \) with output \((\text{abort}, S)\).

**On input (input, \( u_R \)) from corrupted \( R \) to \( T^\mu_{\text{SFE}}[g_R] \), the simulator checks consistency of the sharings in \( u_R \) with the sharings in \( x_R \). If they are consistent, then the simulator lets \( R \) input \((\text{unveil choice})\) into \( T^\mu_{\text{FCOT}} \), otherwise the simulator aborts \( T^\mu_{\text{FCOT}} \) with output \((\text{abort}, R)\).

It remains to be shown that the simulator does provide an indistinguishable view for any input of \( Z \). If inconsistent sharings for \( m_b \) are fed into \( T^\mu_{\text{SFE}}[f] \),
then, upon termination of $SFE[f]$, all honest parties certainly abort with output $(\text{abort}, S)$ due to the common output $\Delta$. Also, if inconsistent sharings for $m_0$ are unveiled in $SFE[g]$, then all honest parties certainly abort with output $(\text{abort}, S)$, as all parties can check their consistency themselves. Consequently, a discrimination between the hybrid and ideal execution can only occur when the input sharings and the unveiled sharings are (internally) consistent but unequal. Fortunately, in this case all parties notice (with overwhelming probability) that the original input $m'_0$ and the unveiled value $m_0$ are not equal. This can be shown as follows: First note that in each of the $r$ sharings at least one share must have been altered, otherwise the sharings are inconsistent. We denote this fact as $Z \geq 1$ to indicate at least one alteration in each sharing. Now, let $H^i_P$ be a boolean random variable and $H^i_P = 1$ be the event that the $i$-th share index of (honest) party $P$ matches the index of the maliciously altered share, else $H^i_P = 0$. Using our Notation 7, we get the probability $\Pr[H^i_P \mid Z \geq 1] \geq 1/n$. Let $N_P := \bigvee_{i=1}^r H^i_P = \neg \bigwedge_{i=1}^r \neg H^i_P$ be the boolean random variable that describes whether $P$ notices an alteration in any (of the $r$) sharings. We can bound the probability for $N_P = 1$ by

$$
\Pr[N_P \mid Z \geq 1] = 1 - \Pr[\neg N_P \mid Z \geq 1] \\
= 1 - \Pr\left[\bigwedge_{i=1}^r \neg H^i_P \mid Z \geq 1\right] \\
= 1 - \prod_{i=1}^r \Pr[\neg H^i_P \mid Z \geq 1] \\
\geq 1 - \prod_{i=1}^r (1 - 1/n) \\
= 1 - (1 - 1/n)^r
$$

(5)

because the different $H^i_P$ are independent of each other. Let the number of honest parties be $h$. The final probability $p$ that all honest parties notice any fault is then greater than $(1 - (1 - 1/n)^r)^h$ which can be bounded by $p > (1 - (1 - 1/n)^r)^h$. We apply some transformations to the desired condition $p \in \text{owhl}(\lambda)$ and get the
sufficient condition

\[ 1 - p \leq 1 - (1 - (1 - 1/n)^r)^n \]

\[ = 1 - \sum_{i=0}^{n} \binom{n}{i} (-1)^i (1 - 1/n)^{ri} \]

\[ = \sum_{i=1}^{n} \binom{n}{i} (-1)^{i+1} (1 - 1/n)^{ri} \]

\[ \leq \left| \sum_{i=1}^{n} \binom{n}{i} (-1)^{i+1} (1 - 1/n)^{ri} \right| \]

\[ \leq \sum_{i=1}^{n} \binom{n}{i} \left| (-1)^{i+1} (1 - 1/n)^{ri} \right| \]

\[ = \sum_{i=1}^{n} \binom{n}{i} (1 - 1/n)^{ri} \]

\[ \leq \left( \frac{n}{n/2} \right) \sum_{i=1}^{n} (1 - 1/n)^{ri} \]

\[ = \left( \frac{n}{n/2} \right) (1 - 1/n)^r \cdot \frac{(1 - 1/n)^n - 1}{(1 - 1/n)^r - 1} \]

\[ \leq \left( \frac{n}{n/2} \right) (1 - 1/n)^r \]

\[ \in \negl(\lambda) \]  

(6)

where the last equality is the partial geometric sum with \( q = (1 - 1/n)^r \). Since \( \binom{n}{n/2} \in \Theta(2^n/\sqrt{n}) \) the condition \( \binom{n}{n/2}(1 - 1/n)^r \in \negl(\lambda) \) is in particular fulfilled by \( r \in \omega(n^2 + n \ln \lambda) \). We write \( r = n(n + \ln \lambda)w \) with \( w \in \omega(1) \) with respect to \( \lambda \to \infty \), thus the above claim follows from

\[ 2^n/\sqrt{n} \cdot (1 - 1/n)^{n(n + \ln \lambda)w} \leq 2^n \cdot (1 - 1/n)^{n(n + \ln \lambda)w} \]

\[ = 2^n \cdot ((1 - 1/n)^n)^{(n + \ln \lambda)w} \]

\[ \leq 2^n \cdot (1/e)^{(n + \ln \lambda)w} \]

\[ = 2^n \cdot e^{-nw \lambda^{-w}} \]

\[ \leq \lambda^{-w} \]

\[ \in \negl(\lambda) \]  

(7)

Although our bound for \( r \) is probably not tight, we already see that the protocol is at most quadratic in \( n \) and logarithmic in \( \lambda \).

\[ \square \]

**Lemma 8** (FCOT \( \leadsto \) SFE). There is a protocol \( \pi^n_{\text{FCOT}} \) in the \( \{ T^n_{\text{SFE}} \} \)-hybrid model that securely realizes \( T^n_{\text{FCOT}} \).

\[ 35 \]
Proof. Here, we prove that there exists a \( \{I_{FCOT}\} \)-hybrid protocol \( \Phi \) that securely UC-realizes \( I_{SFE} \). Again, for arbitrary \( n \), denote the set of parties by \( P = \{S,R,W_1, \ldots, W_{n-2}\} \).

We use the IPS-compiler from Ishai, Prabhakaran, and Sahai [IPS08], which compiles two protocols \( \Pi \) and \( \rho \) into an \( \{F_{2,OT}\} \)-hybrid protocol \( \Phi \). There, the outer protocol \( \Pi \) can be formulated in the client-server model and must be secure against a constant fraction of malicious parties, say \( t \leq n/4 \). The inner protocol \( \rho \) needs to be secure against arbitrarily many, semi-honest (passive) corruptions; it may be in the \( \{F_{2,OT}\} \)-hybrid model. Both protocols depend on the actual function \( f \) that is to be evaluated. The combined protocol \( \Phi \) then securely realizes \( I_{SFE} \) with Anonymous Abort, iff the outer protocol \( \Pi \) securely realizes \( I_{SFE} \). This holds both in the computational and the statistical case.

Their result cannot be directly transferred into the setting of IA. However, if we replace \( F_{2,OT} \)-calls with to \( I_{FCOT} \), we claim that their result still holds. When a party \( P \) notices misbehavior in server \( i \), it can publicly demand the unveiling of all communication corresponding to server \( i \). If any party refuses to unveil, it must be malicious and all honest parties can abort with said party. If all inputs into server \( i \) have been unveiled, then either all parties can retrace any deviation from the correct protocol transcript, or no actual misbehavior has occurred, then the initial party \( P \) demanded the unveiling unjustifiedly. Either way, at least one party can be identified.

Additionally, we must ensure that unveiling all inputs of a single server does not compromise the privacy to the inputs of the outer protocol \( \Pi \). In the following, we formalize this idea: Let \( n \) be the number of parties (clients) of the outer protocol. In the original paper [IPS08] there are \( m \in \Theta(n^2 \lambda) \) servers. Each party gets to select \( \lambda \) watchlists from each party, such that each party can see all in- and outcoming communication of \( \lambda \) servers. In total, at most a fraction of \( n\lambda/n^2\lambda = 1/n \) of all servers state is known by any set of parties. Because the used secret sharing requires a constant fraction of shares to reconstruct the original input, no coalition of parties can learn the input of another party. Now, if misbehavior occurs and the state of an additional server is unveiled any coalition of parties knows at most \( n\lambda/n^2\lambda \leq \lambda \), which is still less than a constant fraction.

To make this more formal, we consider the function \( f \) that presents a boolean circuit in \( NC^1 \). Then, using the BGW-protocol [BGW88] for \( \Pi \) and the GMW-protocol [GMW87] for \( \rho \), we obtain a protocol \( \Phi \) that securely realizes \( F_{SFE}^2[f] \) against arbitrarily many corruptions with Anonymous Abort. Now, we replace all calls to \( F_{2,OT}^2 \) with calls to \( I_{FCOT}^2 \). OT-calls are made in the distribution phase of the watchlist mechanism and in the inner protocol, in particular the outer protocol stays unchanged. Call the new protocols \( \Phi' \) and \( \rho' \). We require that all communication in the new inner protocol is processed via FCOTs. This is not an additional restriction because a secure-channel can be trivially realized by OT and thus by FCOT. However, it enables us to unveil all communication of the inner protocol upon abort.
If not aborted, the original protocol $\Phi$ and the new protocol $\Phi'$ yield exactly the same results. Note that the original simulator $S$ and the new simulator $S'$ learn exactly the same information, if no abort occurs.

In the original protocol, if any party aborts, then the original simulator $S$ also abort the ideal functionality $F_{\text{SFE}}[f]$. The new simulator $S'$, however, must provide a set of corrupted parties to abort the ideal functionality $I_{\text{SFE}}[f]$. Hence, all simulated parties in the simulated new protocol $\Phi'$ must provide output $(\text{abort}, C')$. The new protocol ensures this in the following way. Whenever a party $P$ aborts in the original protocol $\Phi$ due to a malicious message from the $i$-th server, it, instead, broadcasts $(\text{challenge}, i)$. Then, all parties unveil the FCOTs used to distribute their watchlist one-time pads and all FCOTs in the $i$-th instance of the inner protocol ($i$-th server). More precisely, we actually assume a $\binom{n^2}{\lambda}$-FCOT for each party which can be canonically constructed from $\binom{n}{1}$-FCOTs. Then each choice index in the $\binom{n^2}{\lambda}$-FCOT corresponds to multiple choice bits in the $\binom{n}{1}$-FCOTs in a priori known manner, hence all $\binom{n}{1}$-messages $m_c$ associated with the $i$-th watchlist can be unveiled.

Consequently, all parties learn the complete in- and outcoming messages of the $i$-th server but no additional communication of any other server. Thus each party can retrace the complete computation of the $i$-th server and register any deviation from the protocol. If the aborting party $P$ indeed received a malicious message on the $i$-th server, then all party notice this misbehavior and identify the disruptor party $P$. They then abort with $(\text{abort}, P')$. If the aborting party lied about receiving a malicious message on the $i$-th server, then the other parties can retrace that all message that $P$ received were indeed correct, and they will abort with $(\text{abort}, P)$. Either way, the simulator will abort the ideal functionality $I_{\text{SFE}}$ with the corresponding abort output.

Note that because the abort happens at exactly the same time as in the original protocol, the new protocol leaks exactly as much information as the original one which is secure. Also, by unveiling the state of the corrupted server $i$, the adversary does not learn anything that it did not know beforehand. \qed

C Security proof

In this section, we provide the simulators for our protocols from Section 5.

C.1 COM-Expansion

Because in every case the behavior of honest receivers in unambiguous, we can formulate a coherent simulator. In particular, the dummy adversary’s and the corrupted parties’ local in- and output is forwarded from and to the environment.

**On output** (receipt commit) **from** $I_{\text{COM}}^{n}$, the simulator gives local input to all uncorrupted simulated parties as follows: The simulator gives the uncorrupted $C$ local input $(\text{commit}, 0)$, hence $C$ inputs $(\text{commit}, 0)$ into all $I_{R_i}^{-1}$. 37
On output (unveil, m) from $T^n_{\text{COM}}$, the simulator lets the simulated functionalities $T^{n-1}_{R_i}$ output (unveil, m). This is possible because $T^{n-1}_{R_i}$ is a simulated functionality and thus fully under the simulator’s control.

Once all broadcasts (receipt, R) from $T^n_{\text{BC}}$ have been received, the simulator lets C input (unveil) into the ideal functionality $T^n_{\text{COM}}$.

On input (abort, C') from $Z$ for $T^{n-1}_{R_i}$, the simulator sends (abort, C') to $T^{n-1}_{R_i}$.

If the (simulated) conflict graph becomes t-semisettled, the simulator aborts $T^n_{\text{COM}}$ with the malicious partition (abort, C').

Finally, we describe the simulator’s behavior when handling messages from/to malicious parties.

On input (commit, m) from corrupted C to $T^{n-1}_{R_i}$, the simulator lets C pass (commit, m) on to $T^{n-1}_{R_i}$. After receiving local input for all (non-aborted) subfunctionalities, the simulator lets C input (commit, m) into $T^n_{\text{COM}}$ where m is the majority of all $m_j$. If there is no majority then C inputs a random bit $m$. The inputs (commit, $m_j$) are naturally passed to the simulated $T^{n-1}_{R_i}$.

On input (unveil) from corrupted C to $T^{n-1}_{R_i}$, the simulator lets C forward (unveil) to $T^{n-1}_{R_i}$.

On input (receipt commit) from corrupted R to $T^{n-1}_{R_i}$, the simulator forwards (receipt commit) in the name of R. The key to a coherent simulation is that the protocol ensures that the outputs of all (honest) receivers are consistent, this can be leveraged by the simulator to abort the functionality on invalid inputs of the (corrupted) committer.

C.2 FCOT-Expansion

Now, we give a description of the simulator.

On input (messages, $a^0_{j,i}, a^1_{j,i}$) from corrupted S to the i-th session of $T^n_{j}$, the simulator lets S forward the input $a^0_{j,i}$ into $T^n_{FCOT}$.

On input (messages, $a^0_{j,i}, a^1_{j,i}$) from corrupted S to $T^n_{\text{COM}}$, the simulator computes $m_0$ and $m_1$ from the obtained shares. Then the simulator lets S input (messages, $m_0, m_1$) into $T^n_{FCOT}$.

On input (commit, $a^b_{j,i}$) from corrupted S to $T^n_{\text{COM}}$, the simulator lets S input into the simulated (commit, $(a^b_{j,i}, b, i, j)$) to $T^n_{BC}$.

On input (choice, $c_{j,i}$) from corrupted R for the i-th session of $T^n_{j}$, the simulator lets S forward the input to the simulated $T^n_{j}$.

On input (choice, $c_{j,i}$) from corrupted R to all i-th sessions of all remaining FCOT-functionalities and (commit, c) for $T^n_{\text{COM}}$, the simulator lets R input (choice, c) into $T^n_{FCOT}$.

On output (receipt transfer, ⊥) from $T^n_{FCOT}$, the simulator gives local input to simulated parties as follows:

If the receiver is malicious, then S learns $m_c$ from the ideal $T^n_{FCOT}$ in the name of R through (receipt transfer, $m_c$). Otherwise, the simulator gives
the uncorrupted $R$ local input $(\text{choice}, 0)$. If the sender is uncorrupted, the simulator gives $S$ local input $(\text{messages}, m_0, m_1)$ with $m_{1-c} = 0$. If the receiver is uncorrupted, the simulator also uses $m_c = 0$. Then, the simulator simulates the protocol program with the respective inputs. If any party is malicious, its inputs are directly forwarded to the simulated hybrid functionalities. Consequently, the simulated dummy adversary receives $(\text{receipt transfer}, \perp)$ from each simulated FCOT-subfunctionality which is forwarded to the environment.

**On input** $(\text{unveil messages}, b)$ **from corrupted** $S$ **to all remaining** FCOT-functionalities, the simulator lets $S$ input $(\text{unveil messages}, b)$ into $I^s_{FCOT}$.

**On output** $(\text{unveil message}, b, m_b)$ **from** $I^s_{FCOT}$, the simulator unveils all shares of $m_b$ from all FCOT-subfunctionalities and all their commitments. If the local input $(\text{messages}, m_0, m_1)$ of the simulated sender matches with the unveiled $m_b$, then the simulation is valid.

However, if the simulated sender’s input is not equal to the ideal uncorrupted sender’s input, the simulator must equivoke some of the FCOTs and COMs. Therefore, the simulator fabricates additive shares $\mu^b_j$ and sub-shares $\alpha^b_{j,i}$ that encode $m_b$ but still are consistent with the shares that the environment learned so far. The simulator must be able to equivoke more than $r/3$ (out of $r$) shares of a single FCOT-subfunctionality. This is possible because the consistency check in the OT-phase only leaks less than $r/10$ per subfunctionality to the environment, leaving more than $9r/10$ for the simulator to equivoke.

If the receiver is malicious, then the environment also learns the shares that the receiver obtains during the OT-phase. However, if the receiver is malicious, then the simulator already learned $m_c$ prior to giving the simulated sender its input, hence the simulated sender’s input $m_c$ is consistent with $m_b$ if $b = c$. Otherwise the environment only has less than $r/2$ shares per FCOT, leaving $r - (r/2 + r/10) > r/3$ for the simulator to equivoke.

Furthermore, the simulator knows exactly which shares can be equivoked, since every share that the environment obtains comes from the simulator.

**On input** $(\text{unveil choice})$ **from corrupted** $R$ **to all remaining** FCOT-functionalities, the simulator lets $R$ input $(\text{unveil choice})$ into $I^s_{FCOT}$.

**On output** $(\text{unveil choice}, c)$ **from** $I^s_{FCOT}$, the simulator lets all sessions of all simulated FCOT-subfunctionalities $I^s_w$ output $(\text{unveil choice}, c)$ to all parties and the dummy adversary.

**On input** $(\text{abort}, C')$ **for** $I^{n-1}_W$, the simulator aborts $I^{n-1}_W$ with $(\text{abort}, C')$.

**If the (simulated) conflict graph becomes** $t$-semisettled, **the simulator aborts** $I^n_{COM}$ **with the malicious partition** $(\text{abort}, C')$.

### D Conflict Graph from Broadcast

In this section, we present a protocol $\pi^n_{CG}$ that realizes $T^n_{CG}$ in the $\{T^n_{BC}\}$-hybrid model, which proves the following lemma:
Lemma 9. There is a protocol $\pi_{CG}^n$ in the $\{T_{BC}^n\}$-hybrid model that securely realizes $T_{CG}^n$:

$$T_{BC}^n \rightsquigarrow T_{CG}^n$$

Proof. We proof our statement by providing a protocol description for $\pi_{CG}^n$ and prove it secure by providing a simulator. We have $n$ parties $P = \{P_1, \ldots, P_n\}$. The protocol is given as follows:

Initialize. All parties start with a graph $G = (P, E)$ with $E = \emptyset$.
On input $(\text{conflict}, P_i)$ from $Z$ to $P_j$, $P_j$ inputs $(\text{input}, (\text{conflict}, P_j, P_i))$ to $T_{BC}^n$.
On input $(\text{output}, (\text{conflict}, P_j, P_i))$ from $T_{BC}^n$, all parties $P$ add $\{P_j, P_i\}$ to $E$.
On input $(\text{query})$ from $Z$ to $P_i$, $P_i$ locally computes $G^* := \text{DeduceCG}(P, E, t)$ and outputs $G^*$.

A simulator for this protocol is straightforward:
1. If $P$ is corrupted:
   On input $(\text{input}, (\text{conflict}, P_j, P_i))$ from $P$ to $T_{BC}^n$, if $P_j = P$, $S$ calls $T_{CG}^n$ with input $(\text{conflict}, P_i)$ in the name of $P$ and sends $(\text{output}, (\text{conflict}, P_j, P_i))$ to all other parties in the name of $T_{BC}^n$.
2. If $P$ is honest:
   On input $(\text{conflict}, P, P_i)$ from $T_{CG}^n$ to $S$, $S$ sends $(\text{output}, (\text{conflict}, P, P_i))$ in the name of $T_{BC}^n$ to all other parties.

The simulator provides an indistinguishable view for $Z$:
• Inputs $(\text{query})$ do not have to be handled at all. For honest parties, the parties merely forward the request and obtain the correct conflict graph $G$. Corrupted parties neither send messages for $(\text{query})$, nor change the state of the functionality $T_{CG}^n$ in any way.
• For corrupted parties, $S$ obtains the input via the simulated $T_{BC}^n$. If the broadcasted message was valid, $S$ inputs this into $T_{CG}^n$, thus causing the same behavior as if an honest party had called $T_{CG}^n$.
• For honest parties, $S$ only has to simulate the behavior of $T_{BC}^n$.

E Global Commitment from Fully Committed Oblivious Transfer

We present a protocol, which realizes $T_{COM}^n$ in a $T_{FCOT}^n$-hybrid model, and thus prove the following lemma:

Lemma 10. There is a protocol $\pi_{COM}^n$ that securely realizes $T_{COM}^n$ in the $\{T_{FCOT}^n\}$-hybrid model:

$$T_{FCOT}^n \rightsquigarrow T_{COM}^n$$

Proof. We assume our $n$ parties for $T_{COM}^n$ to be $P := (C, R_1, \ldots, R_{n-1})$, our $n$ parties for $T_{FCOT}^n$ are $P' := (S, R, W_1, \ldots, W_{n-2})$. We start by sketching the protocol:
On input \((\text{commit}, m)\) for \(m \in \{0, 1\}\) from \(Z\) to \(C\), \(C\) acts as \(R\) in \(T^n_{FCOT}\) and inputs \((\text{choice}, m)\).

On input \((\text{receipt commit})\) from \(T^n_{COM}\) to \(R_i\) for \(i \in [n-1]\), all receiver for \(i \neq 1\) ignore the message. \(R_1\) acts as the sender in \(T^n_{FCOT}\): it draws one random message \(m \xleftarrow{\$} \{0, 1\}\) and sends \((\text{messages}, m, m)\) to \(T^n_{FCOT}\).

On input \((\text{unveil})\) from \(Z\) to \(C\) and \((\text{receipt transfer}, m_c)\) from \(T^n_{FCOT}\) to \(C\), \(C\) sends \((\text{unveil choice})\) to \(T^n_{FCOT}\).

On input \((\text{unveil choice}, c)\) from \(T^n_{FCOT}\) to any receiver \(R_i\) for \(i \in [n-1]\), \(R_i\) outputs \((\text{output}, c)\).

A simulator for this case is straightforward, since all the secrets are sent to the hybrid functionality \(T^n_{FCOT}\):

1. If \(C\) is corrupted:
   On input \((\text{choice}, c)\) from \(C\) to \(T^n_{FCOT}\), \(S\) sends \((\text{commit}, c)\) to \(T^n_{COM}\) in the name of \(C\).

2. If \(C\) is honest:
   On input \((\text{receipt commit})\) from \(T^n_{COM}\), \(S\) simulates \(T^n_{FCOT}\) according to the code of \(S_{FCOT}\) with arbitrary input.

3. If \(R_i\) for \(i \in [n]\) is corrupted:
   On input \((\text{messages}, m_0, m_1)\), if \(m_c\) or \(m_c /\in \{0, 1\}\), \(S\) aborts with output \(R_1\).
   Else, \(S\) reports \((\text{receipt transfer})\) to \(Z\).

4. If \(R_i\) for \(i \in [n]\) is honest:
   \(S\) acts according to the protocol of \(R_i\).

5. If \(C\) is corrupted:
   On input \((\text{unveil choice})\) from \(R\) to \(T^n_{FCOT}\), \(S\) sends \(c\) to all \(R_i\) for \(i \in [n-1]\).

6. If \(C\) is honest:
   On input \((\text{unveil}, c)\) from \(T^n_{COM}\), \(S\) reports message \((\text{unveil choice}, c)\) to all \(R_i\).

The simulator trivially provides an indistinguishable view:

- Simulation of \(T^n_{FCOT}\) follows from simulation-based security.
- The only secret is the to-be-committed bit \(m\), as the receivers \(R_i\) obtain no secret input, meaning that \(S\) can execute their protocol.
- Against an honest committer, \(S\) just has to send messages from \(T^n_{FCOT}\) accordingly and pretend that \(C\) used the correct choice bit – which does not have to be known in advance, as \((\text{unveil choice}, c)\) is only required after \(T^n_{COM}\) unveiled \(c\).
- Against a corrupted committer, \(S\) learns \(c\) via simulation of \(T^n_{FCOT}\).

Thus, the claim follows. \(\square\)

References


