Round-Preserving Parallel Composition of Probabilistic-Termination Cryptographic Protocols*

Ran Cohen[†1], Sandro Coretti[‡2], Juan Garay[3], and Vassilis Zikas[4]

1 **School of Computer Science, Tel Aviv University, Tel Aviv, Israel**
cohenran@tauex.tau.ac.il
2 **Courant Institute of Mathematical Sciences, New York University, New York, NY, USA**
corettis@nyu.edu
3 **Yahoo Research, Sunnyvale, CA, USA**
garay@yahoo-inc.com
4 **Dept. of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, USA**
vzikas@cs.rpi.edu

Abstract

An important benchmark for secure multi-party computation (MPC) protocols is their *round complexity*. For several important MPC tasks, (tight) lower bounds on the round complexity are known. However, for some of these tasks, such as broadcast, the lower bounds can be circumvented when the termination round of every party is not *a priori* known, and simultaneous termination is not guaranteed. Protocols with this property are called *probabilistic-termination (PT)* protocols.

Running PT protocols in parallel affects the round complexity of the resulting protocol in somewhat unexpected ways. For instance, an execution of m protocols with constant expected round complexity might take $O(\log m)$ rounds to complete. In a seminal work, Ben-Or and El-Yaniv (Distributed Computing '03) developed a technique for parallel execution of arbitrarily many broadcast protocols, while preserving expected round complexity. More recently, Cohen *et al.* (CRYPTO '16) devised a framework for universal composition of PT protocols, and provided the first composable parallel-broadcast protocol with a simulation-based proof. These constructions crucially rely on the fact that broadcast is "privacy free," and do not generalize to arbitrary protocols in a straightforward way. This raises the question of whether it is possible to execute *arbitrary* PT protocols in parallel, without increasing the round complexity.

In this paper we tackle this question and provide both feasibility and infeasibility results. We construct a round-preserving protocol compiler, secure against a minority of actively corrupted parties, that compiles arbitrary protocols into a protocol realizing their parallel composition, while having a black-box access to the underlying *protocols*. Furthermore, we prove that the same cannot be achieved, using known techniques, given only black-box access to the *functionalities* realized by the protocols, unless merely security against semi-honest corruptions is required, for which case we provide a protocol.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases Cryptographic protocols, secure multi-party computation, broadcast

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.37

* The full version of this paper can be found at the *IACR Cryptology ePrint Archive* [15], http://eprint.iacr.org/2017/364.

† Research supported by ERC starting grant 638121.

‡ Author supported by NSF grants 1314568 and 1319051.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 37; pp. 37:1–37:15

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Secure multi-party computation (MPC) [52, 30] allows a set of parties to jointly perform a computation on their inputs, in such a way that no coalition of cheating parties can learn any information beyond what is revealed by their outputs (privacy) or affect the outputs of the computation in any way other than by choosing their own inputs (correctness). Since the first seminal works on MPC [52, 30, 6, 12, 50], it has been studied in a variety of different settings and for numerous security notions: there exist protocols secure against passively corrupted (aka semi-honest) parties and against actively corrupted (aka malicious) parties; the underlying network can be synchronous or asynchronous; and the required security guarantees can be information-theoretic or computational – -to name but a few of the axes along which the MPC task can be evaluated.

The prevalent model for the design of MPC protocols is the synchronous model, where the protocol proceeds in rounds. In this setting, the round complexity, i.e., the number of rounds it takes for a protocol to deliver outputs, is arguably the most important efficiency metric. Tight lower bounds are known on the round complexity of several MPC tasks. For example, for the well-known problems of Byzantine agreement (BA) and broadcast [48, 44], it is known that any protocol against an active attacker corrupting a linear fraction of the parties has linear round complexity [25, 23]. This result has quite far-reaching consequences as, starting with the seminal MPC works mentioned above, a common assumption in the design of secure protocols has been that the parties have access to a broadcast channel, which they potentially invoke in every round. In reality, such a broadcast channel might not be available and would have to be implemented by a broadcast protocol designed for a point-to-point network. It follows that even though the round complexity of many MPC protocols is linear in the multiplicative depth of the circuit being computed, their actual running time depends on the number of parties, when executed over point-to-point channels.

The above lower bound on the number rounds for BA holds when all honest parties are required to complete the protocol together, at the same round [22]. Indeed, *randomized* BA protocols that circumvent this lower bound and run in *expected constant* number of rounds (cf. [4, 49, 24, 26, 41, 45]) do not provide simultaneous termination, i.e., once a party completes the protocol's execution it cannot know whether all honest parties have also terminated or if some honest parties are still running the protocol; in particular, the termination round of each party is not a priori known. A protocol with this property is said to have *probabilistic termination (PT)*.

As pointed out by Ben-Or and El-Yaniv [5], when several such PT protocols are executed in parallel, the expected round complexity of the combined execution might no longer be constant (specifically, might not be equal to the maximum of the expected running times of the individual protocols). Indeed, when m protocols, whose termination round is geometrically distributed (and so, have constant expected round complexity), are run in parallel, the expected number of rounds that elapse before all of them terminate is $\Theta(\log m)$ [14]. While an elegant mechanism was proposed in [5] for implementing parallel calls to broadcast such that the total expected number of rounds remains constant, it did not provide any guarantees to remain secure under composition, raising questions about its usability in a higher-level protocol (such as the MPC setting described above). Such a shortcoming was recently addressed by Cohen *et al.* [14] who provided a framework for universal composition of PT protocols (building upon the universal-composition framework of [8]). An application of their result was the first composable protocol for parallel broadcast (with a simulation-based proof) that can be used for securely replacing broadcast channels in arbitrary protocols, and whose round complexity is constant in expectation.

Indeed, an immediate application of the composable parallel-broadcast protocol from [14] is plugging it into broadcast-model MPC protocols in order to obtain point-to-point protocols with a round complexity that is independent of the number of parties. In the information-theoretic setting, this approach yields protocols whose round complexity depends on the depth of the circuit computing the function [6, 12, 50, 18], whereas in the computational setting, assuming standard cryptographic assumptions, this approach yields expected-constant-round protocols [43, 3, 20, 40, 1, 29, 31, 46]. However, the resulting point-to-point protocols have probabilistic-termination on their own. The techniques used for composing PT broadcast protocols in parallel crucially rely on the fact that broadcast is a privacy-free functionality, and a naïve generalization of this approach to arbitrary PT protocols fails to be secure. This raises the question of whether it is possible to execute *arbitrary* PT protocols in parallel, without increasing the round complexity.

We remark that circumventing lower bounds on round complexity is just one of the areas where such PT protocols have been successfully used. Indeed, randomizing the termination round has been proven to be a very useful technique in circumventing impossibilities and improving efficiency for many cryptographic protocols. Notable examples include *non-committing encryption* [21], cryptographic protocols designed for rational parties [34, 27, 47, 2, 32, 28], concurrent zero-knowledge protocols [9, 13] and parallel repetition of interactive arguments [33, 35]. The rich literature on such protocols motivates a thorough investigation of their security and composability. As mentioned above, in [14] the initial foundations were laid out for such an investigation, but what was proven for arbitrary PT protocols was a round-preserving *sequential* composition theorem, leaving parallel composition as an open question.

Our contributions. In this work, we investigate the issue of parallel composition for *arbitrary* protocols with probabilistic termination. In particular, we develop a compiler such that given functionalities $\mathcal{F}_1, \ldots, \mathcal{F}_{\mathtt{M}}$ and protocols $\pi_1, \ldots, \pi_{\mathtt{M}}$, where for every $i \in [\mathtt{M}]$, protocol π_i realizes $\mathcal{F}_i$ (possibly using correlated randomness as setup[1]), then the compiled protocol realizes the parallel composition of the functionalities, denoted $(\mathcal{F}_1 \| \cdots \| \mathcal{F}_{\mathtt{M}})$.

Our compiler uses the underlying protocols in a black-box manner,[2] is robust (i.e., secure without abort) and resilient against a computationally unbounded active adversary, adaptively corrupting up to $t < n/2$ parties (which is optimal [50]). Moreover, our compiler is round-preserving, meaning that if the maximal (expected) round complexity of each protocol is μ, then the expected round complexity of the compiled protocol is $O(\mu)$. For example, if each protocol π_i has constant expected round complexity, then so does the compiled protocol. Recall that this task is quite complicated even for the simple case of BA (cf. [5, 14]). For arbitrary functionalities it is even more involved, since as we show, the approach from [5] cannot be applied in a functionally black-box way in this case. Thus, effectively, our result is the first round-preserving parallel composition result for arbitrary multi-party protocols/tasks with probabilistic termination.

We now describe the ideas underlying our compiler. In [5] (see also [14]), a round-preserving parallel-broadcast protocol was constructed by iteratively running, for a constant number of rounds, *multiple* instances of BA protocols (each instance is executed multiple

[1] A trusted setup phase is needed for implementing broadcast in the honest-majority setting. As shown in [17, 16] some interesting functions can be computed without such a setup phase.

[2] Following [37], by a black-box access to a protocol we mean a black-box usage of a semi-honest MPC protocol computing its next-message function.

times in parallel, in a batch), hoping that at least one execution of every BA instance will complete. By choosing the multiplicity suitably, this would occur with constant probability, and the process need therefore be repeated a constant expected number of times only.

At first sight it might seem that this idea can be applied to arbitrary tasks, but this is not the case. Intuitively, the reason is that if the tasks that we want to compose in parallel have privacy requirements, then making the parties run them in (parallel) "batches" with the same input might compromise privacy, since the adversary will be able to use different inputs and learn multiple outputs of the function(s). This issue is not relevant for broadcast, because it is a "privacy-free" functionality; the adversary may learn the result of multiple computations using the same inputs for honest parties, without compromising security.

To cope with the above issue, our parallel-composition compiler generalizes the approach of [5] in a privacy-preserving manner. At a high level, it wraps the batching technique by an MPC protocol which restricts the parties to use the same input in all protocols for the same function. In particular, the compiler is defined in the *Setup-Commit-then-Prove* hybrid model [10, 39], which allows each party to commit to its input values and later execute multiple instances of every protocol, each time proving that the same input value is used in all executions.

The constructions in [10, 39] for realizing the Setup-Commit-then-Prove functionality are designed for the dishonest-majority setting and therefore allow for a premature abort. Since we assume an honest majority, we require security without abort. A possible way around would be, as is common in the MPC literature, to restart the protocol upon discovering some cheating or add for each abort a recovery round; this, however, would induce a linear overhead (in the number of parties) on the round complexity of the protocol.

Instead, in order to recover from a misbehavior by corrupted parties, we modify the Setup-Commit-then-Prove functionality and secret-share every committed random string between all the parties, using an error-correcting secret-sharing scheme (aka robust secret sharing [50, 19, 11]). In case a party is identified as cheating, every party broadcasts the share of the committed randomness corresponding to that party, reconstructs the correlated randomness for that party, and locally computes the messages corresponding to this party in every instance of every protocol. We also prove that the modified Setup-Commit-then-Prove functionality can be realized in a constant number of rounds, thus yielding no (asymptotic) overhead on the round complexity of the compiler.

Next, given that using only black-box access to the protocols π_i, it is possible to compile them into a protocol that implements the parallel composition $(\mathcal{F}_1 \parallel \cdots \parallel \mathcal{F}_{\mathtt{M}})$ of the functionalities $\mathcal{F}_1, \ldots, \mathcal{F}_{\mathtt{M}}$ realized by protocols $\pi_1, \ldots, \pi_{\mathtt{M}}$, we investigate the question of whether there exists a protocol that securely realizes $(\mathcal{F}_1 \parallel \cdots \parallel \mathcal{F}_{\mathtt{M}})$ given only black-box access to the *functionalities* $\mathcal{F}_1, \ldots, \mathcal{F}_{\mathtt{M}}$,[3] but not to protocols realizing them. This question is only sensible if asked for an entire *class* of functionalities (cf. [51]), since otherwise a protocol may always ignore the functionalities $\mathcal{F}_1, \ldots, \mathcal{F}_{\mathtt{M}}$ and implement $(\mathcal{F}_1 \parallel \cdots \parallel \mathcal{F}_{\mathtt{M}})$ from scratch.

On the one hand, we prove that against semi-honest corruptions, there indeed exists a protocol for parallel composition of arbitrary functionalities $\mathcal{F}_i$ in a functionally black-box manner. On the other hand, in the case of active corruptions, we devise a class of functionalities for which, when using a generalization of the "batching" technique from [5],

[3] Loosely speaking, a functionally black-box protocol, as defined in [51], is a protocol that can compute a function f without knowing the code of f, i.e., given an oracle access to the function f. Note that in this model, each ideal functionality $\mathcal{F}_i$ has an oracle access to the function f_i it computes.

such a black-box transformation is not possible in the presence of a single active corrupted party. More precisely, (1) a naïve call to each of the ideal functionalities $\mathcal{F}_i$ until termination will not be round-preserving, (2) it is impossible to compute the parallel composition without calling every ideal functionality, and (3) using the same input value in more than one call to any of the ideal functionalities will break privacy.[4] This negative result validates our choice of a *protocol* compiler, and is evidence that such a task, if at all possible, would require entirely new techniques.

We phrase our results using the framework for composition of protocols with probabilistic termination [14], extending it (a side result of independent interest) to include parallel composition, reactive functionalities (to capture the Setup-Commit-then-Prove functionality), and the higher corruption threshold of $t < n/2$.

Organization of the paper. The rest of the paper is organized as follows. In Section 2 we describe the network model, the basics of the probabilistic-termination framework by Cohen *et al.* [14], and other tools that are used throughout the paper. We start Section 3 with the extensions to the PT framework, followed by the protocol that achieves round-preserving parallel composition for arbitrary functionalities in a functionally black-box manner against semi-honest adversaries. Section 4 is dedicated to active corruptions; first, the round-preserving protocol-black-box construction, followed by the negative result on round-preserving functionally black-box composition in the case of active corruptions. Due to space limitations, finer details of the model and PT framework, our extension of the PT framework to the honest-majority setting, and proofs, will only appear in the full version of the paper.

2 Model and Preliminaries

2.1 Synchronous Protocols in UC

We consider synchronous protocols in the model of Katz *et al.* [42], which is designed on top of the universal composability framework of Canetti [8]. More specifically, we consider n parties $P_1, \ldots, P_n$ and a computationally unbounded, adaptive *t-adversary* that can dynamically corrupt up to t parties during the protocol execution. Synchronous protocols in [42] are protocols that run in a hybrid model where parties have access to a simple "clock" functionality. This functionality keeps an indicator bit, which is switched once *all honest parties* request the functionality to do so, i.e., once all honest parties have completed their operations for the current round. In addition, all communication is done over bounded-delay secure channels, where each party requests the channel to fetch messages that are sent to him, such that the adversary is allowed to delay the message delivery by a bounded and a priori known number of fetch requests. Stated differently, once the sender has sent some message, it is guaranteed that the message will be delivered within a known number of activations of the receiver. For simplicity, we assume that every message is delivered within a single fetch request. A more detailed overview of [42] can be found in the full version.

[4] We note that our negative result does not contradict Ishai et al. [38, 36] who constructed two-round protocols with guaranteed output delivery for $n \geq 4$ and $t = 1$ without broadcast. Indeed, the protocols in [38, 36] are non-black-box with respect to the function to be computed.

2.2 The Probabilistic-Termination Framework

Cohen *et al.* [14] extended the UC framework to capture protocols with probabilistic termination, i.e., protocols without a fixed output round and without simultaneous termination. This section outlines their techniques; additional details can be found in the full version.

Canonical synchronous functionalities. The main idea behind modeling probabilistic termination is to separate the functionality to be computed from the round complexity that is required for the computation. The atomic building block in [14] is a functionality template called a *canonical synchronous functionality (CSF)*, which is a simple two-round functionality with explicit (one-round) input and (one-round) output phases. The functionality $\mathcal{F}_{\text{CSF}}$ has two parameters: (1) a (possibly) randomized function f that receives $n+1$ inputs (n inputs from the parties and one additional input from the adversary) and (2) a leakage function l that determines what information about the input values is leaked to the adversary.

$\mathcal{F}_{\text{CSF}}$ proceeds in two rounds: in the first (input) round, all the parties hand $\mathcal{F}_{\text{CSF}}$ their input values, and in the second (output) round, each party receives its output. Whenever some input is submitted to $\mathcal{F}_{\text{CSF}}$, the adversary is handed some leakage function of this input; the adversary can use this leakage when deciding the inputs of corrupted parties. Additionally, he is allowed to input an extra message, which – depending on the function f – might affect the output(s).

Wrappers and traces. Computation with probabilistic termination is captured by defining *output-round randomizing wrappers*. Such wrappers address the issue that while an ideal functionality abstractly describes a protocol's task, it does not describe its round complexity. Each wrapper is parametrized by a distribution (more precisely, an efficient probabilistic sampling algorithm) D that may depend on a specific protocol implementing the functionality. The wrapper samples a round $\rho_{\mathsf{term}} \leftarrow D$, by which all parties are guaranteed to receive their outputs. Two wrappers are considered: the first, denoted $\mathcal{W}_{\text{strict}}$, ensures in a strict manner that all (honest) parties terminate together in round ρ_{term}; the second, denoted $\mathcal{W}_{\text{flex}}$, is more flexible and allows the adversary to deliver outputs to individual parties at any time before round ρ_{term}.

As pointed out in [14], it is not sufficient to inform the simulator $\mathcal{S}$ about the round ρ_{term}. In many cases, the wrapper should explain to $\mathcal{S}$ *how* this round was sampled; concretely, the wrapper provides $\mathcal{S}$ with the random coins that are used to sample ρ_{term}. In particular, $\mathcal{S}$ learns the entire *trace* of calls to ideal functionalities that are made by the protocol in order to complete by round ρ_{term}. A trace basically records which hybrids were called by a protocol's execution, and in a recursive way, for each hybrid, which hybrids would have been called by a protocol realizing that hybrid.The recursion ends when the base case is reached, i.e., when the protocol is defined using the *atomic* functionalities that are "assumed" by the model.[5] Formally, a trace is defined as follows:

▶ **Definition 1** (Traces). A *trace* is a rooted tree of depth at least 1, in which all nodes are labeled by functionalities and where every node's children are *ordered*. The root and all internal nodes are labeled by wrapped CSFs (by either of the two wrappers), and the leaves are labeled by unwrapped CSFs. The *trace complexity* of a trace T, denoted $c_{\mathsf{tr}}(T)$, is the number of leaves in T. Moreover, denote by $\mathsf{flex}_{\mathsf{tr}}(T)$ the number nodes labeled by flexibly wrapped CSFs in T.

[5] The atomic functionalities considered in this work are the CSFs for the point-to-point communication functionality $\mathcal{F}_{\text{SMT}}$ and the correlated-randomness functionality for broadcast $\mathcal{F}_{\text{CORR-BC}}$.

Sequential composition of probabilistic-termination protocols. When a set of parties execute a probabilistic-termination protocol, or equivalently, invoke a flexibly wrapped CSF, they might get out-of-sync and start the next protocol in different rounds. The approach in [14] for dealing with sequential composition is to start by designing simpler protocols, that are in a so-called *synchronous normal form*, where the parties remain in-sync throughout the execution, and next, compile these protocols into slack-tolerant protocols.

▶ **Definition 2** (Synchronous normal form)**.** Let $\mathcal{F}_1, \ldots, \mathcal{F}_m$ be CSFs. A synchronous protocol π in the $(\mathcal{F}_1, \ldots, \mathcal{F}_m)$-hybrid model is in *synchronous normal form (SNF)* if in every round exactly one ideal functionality $\mathcal{F}_i$ is invoked by all honest parties, and in addition, no honest party hands inputs to other CSFs before this instance halts.

SNF protocols are designed as an intermediate step only, since the hybrid functionalities $\mathcal{F}_1, \ldots, \mathcal{F}_m$ are two-round CSFs, and, in general, cannot be realized by real-world protocols. In order to obtain protocols that can be realized in the real world, [14] introduced *slack-tolerant* variants of both the strict and the flexible wrappers, denoted $\mathcal{W}_{\text{sl-strict}}$ and $\mathcal{W}_{\text{sl-flex}}$. These wrappers are parametrized by a slack parameter $c \geq 0$ and can be used even if parties provide inputs within $c+1$ consecutive rounds (i.e., they tolerate input slack of c rounds); furthermore, the wrappers ensure that all honest parties obtain output within two consecutive rounds (i.e., they reduce the slack to $c = 1$). Next, [14] constructed compilers to convert any SNF protocol realizing a wrapped CSF $\mathcal{W}^{D}_{\text{strict}}(\mathcal{F})$ (resp., $\mathcal{W}^{D}_{\text{flex}}(\mathcal{F})$) into a (non SNF) protocol realizing $\mathcal{W}^{D',c}_{\text{sl-strict}}(\mathcal{F})$ (resp., $\mathcal{W}^{D',c}_{\text{sl-flex}}(\mathcal{F})$), using wrapped CSFs as hybrids. The compilers maintain the security and the asymptotic (expected) round complexity of the original SNF protocols. At the same time, the compilers take care of any potential slack that is introduced by the protocol and ensure that the resulting protocol can be safely executed even if the parties do not start the protocol simultaneously.

Finally, in [14], the authors also provided protocols for realizing wrapped variants of the atomic CSF functionality for secure point-to-point communication. This suggested the following design paradigm for realizing a wrapped functionality $\mathcal{W}_{\text{sl-strict}}(\mathcal{F})$ (resp., $\mathcal{W}_{\text{sl-flex}}(\mathcal{F})$): First, construct an SNF protocol for realizing $\mathcal{W}_{\text{strict}}(\mathcal{F})$ (resp., $\mathcal{W}_{\text{flex}}(\mathcal{F})$) using CSF hybrids $\mathcal{F}_1, \ldots, \mathcal{F}_m$. Next, for each of the non-atomic hybrids $\mathcal{F}_i$, show how to realize $\mathcal{W}_{\text{strict}}(\mathcal{F}_i)$ (resp., $\mathcal{W}_{\text{flex}}(\mathcal{F}_i)$) using CSF hybrids $\mathcal{F}'_1, \ldots, \mathcal{F}'_{m'}$. Proceed in this manner until all CSF hybrids are atomic functionalities. Finally, repeated applications of the composition theorems above yield a protocol for $\mathcal{W}_{\text{sl-strict}}(\mathcal{F})$ (resp., $\mathcal{W}_{\text{sl-flex}}(\mathcal{F})$) using only atomic functionalities as hybrids.

2.3 A Lemma on Termination Probabilities

The following lemma, which will be used in our positive results, provides a constant lower bound on the probability that when running simultaneously (i.e., in parallel) $\mathtt{N}$ copies of $\mathtt{M}$ probabilistic-termination protocols $\pi_1, \ldots, \pi_{\mathtt{M}}$, at least one copy of each π_i will complete after $\mathtt{R}$ rounds, for suitable choices of $\mathtt{N}$ and $\mathtt{R}$. The proof of the lemma appears in the full version.

▶ **Lemma 3.** *Let $M, N, R \in \mathbb{N}$. For $i \in [M]$ and $j \in [N]$, let X_{ij} be independent random variables over the natural numbers, such that $X_{i1}, \ldots, X_{iN}$ are identically distributed with expectation μ_i, for every $i \in [M]$. Denote $Y_i = \min\{X_{i1}, \ldots, X_{iN}\}$ and $\mu = \max\{\mu_1, \ldots, \mu_M\}$.*

Then, for any constant $0 < \epsilon < 1$, if $R > \mu$ and $N > \frac{\log(M/\epsilon)}{\log(R/\mu)}$, then $\Pr[\forall i : Y_i < R] \geq 1 - \epsilon$.

3 Round-Preserving Parallel Composition: Passive Security

In this section, we show that round-preserving parallel composition is feasible, in a functionally black-box manner, facing semi-honest adversaries. To that end we first extend, in Section 3.1, the probabilistic-termination framework [14] to capture the notions of functionally black-box protocols [51] and of parallel composition of canonical synchronous functionalities. The passively secure protocol is presented in Section 3.2.

3.1 Functionally Black-Box Protocols and Parallel Composition

Functionally black-box protocols. We formalize the notion of functionally black-box protocols of Rosulek [51] in the language of canonical synchronous functionalities. As in [51], we focus on secure function evaluation. The SFE functionality $\mathcal{F}_{\text{SFE}}^{g}$, parametrized by an n-party function g, is defined as the CSF $\mathcal{F}_{\text{CSF}}^{f_{\text{SFE}},l_{\text{SFE}}}$, where $f_{\text{SFE}}(x_1,\ldots,x_n,a) = g(x_1,\ldots,x_n)$ (i.e., computes the function g while ignoring the adversary's input a) and the leakage function is $l_{\text{SFE}}(x_1,\ldots,x_n) = (|x_1|,\ldots,|x_n|)$. The following definition explains what we mean by a protocol that realizes the secure function evaluation functionality in a black-box way with respect to the function g.

▶ **Definition 4.** Let $\mathcal{C} = \{g\colon (\{0,1\}^*)^n \to (\{0,1\}^*)^n\}$ be a class of n-party functions. Denote by $\mathcal{F}_{\text{SFE}}^{\mathcal{C}}$ the CSF, implemented as an (uninstantiated) oracle machine that in order to compute $f_{\text{SFE}}^{\mathcal{C}}(x_1,\ldots,x_n,a)$, queries the oracle with $(x_1,\ldots,x_n)$ and stores the response $(y_1,\ldots,y_n)$. The leakage function $l_{\text{SFE}}(x_1,\ldots,x_n) = (|x_1|,\ldots,|x_n|)$ is unchanged.

Then, a protocol $\pi = (\pi_1,\ldots,\pi_n)$ is a functionally black-box (FBB) protocol for (a wrapped version of) $\mathcal{F}_{\text{SFE}}^{\mathcal{C}}$, if for every $f \in \mathcal{C}$, the protocol $\pi^f = (\pi_1^f,\ldots,\pi_n^f)$ UC-realizes $\mathcal{F}_{\text{SFE}}^{f}$.

Parallel Composition of CSFs. The parallel composition of CSFs is defined in a natural way as the CSF that evaluates the corresponding functions in parallel.

▶ **Definition 5.** Let $f_1,\ldots,f_{\mathtt{M}}$ be n-input functions. We define the $(n\cdot\mathtt{M})$-input function $(f_1 \parallel \cdots \parallel f_{\mathtt{M}})$ as follows. Upon input $(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n)$, where each $\boldsymbol{x}_i$ is an $\mathtt{M}$-tuple $(x_i^1,\ldots,x_i^{\mathtt{M}})$, the output is the $\mathtt{M}$-tuple defined as

$$(f_1 \parallel \cdots \parallel f_{\mathtt{M}})(\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n) = \big((y_1^1,\ldots,y_1^{\mathtt{M}}),\ldots,(y_n^1,\ldots,y_n^{\mathtt{M}})\big)\,,$$

where $(y_1^j,\ldots,y_n^j) = f_j(x_1^j,\ldots,x_n^j)$.

Let $\mathcal{F}_{\text{CSF}}^{f_1,l_1},\ldots,\mathcal{F}_{\text{CSF}}^{f_{\mathtt{M}},l_{\mathtt{M}}}$ be CSFs, denote $\mathcal{F}_i = \mathcal{F}_{\text{CSF}}^{f_i,l_i}$. The parallel composition of $\mathcal{F}_1,\ldots,\mathcal{F}_{\mathtt{M}}$, denoted as $(\mathcal{F}_1 \parallel \cdots \parallel \mathcal{F}_{\mathtt{M}})$, is the CSF defined by the function $(f_1 \parallel \cdots \parallel f_{\mathtt{M}})$ and the leakage function $(l_1 \parallel \cdots \parallel l_{\mathtt{M}})$.

3.2 Passively Secure FBB Parallel-Composition Protocol

The underlying idea of our protocol is based on a simplified form of the parallel-broadcast protocol of Ben-Or and El-Yaniv [5]. The protocol proceeds in iterations, where in each iteration, the parties invoke, in parallel and using the same input values, sufficiently many instances of each (oracle-aided) ideal functionality, but only for a constant number of rounds. If some party received an output in at least one invocation of every ideal functionality, it distributes all output values and the protocol completes; otherwise, the protocol resumes with another iteration. This protocol retains privacy for deterministic functions,[6] since the

[6] Although the result holds for deterministic functionalities, we note that using standard techniques every functionality can be transformed to an equivalent deterministic functionality in a black-box way.

adversary is semi-honest, and so corrupted parties will provide the same input values to all instances of each ideal functionality.

Intuitively, during the simulation of the protocol, the simulator should imitate every call for every ideal functionality towards the adversary. A subtle issue is that in order to do so, the simulator must know the exact trace that is sampled by each instance of each ideal functionality during the execution of the real protocol. Therefore, it is indeed essential for the simulator to receive the *random coins* used to sample the trace for the entire protocol, by the ideal functionality computing the parallel composition (cf. Section 2.2). By defining the trace-distribution sampler in a way that consists of all (potential) sub-traces for every instance of every ideal functionality, the simulator can induce the exact random coins used to sample the correct sub-trace for every ideal functionality that is invoked.

▶ **Theorem 6.** *Let $\mathcal{C}_1, \ldots, \mathcal{C}_M$ be (deterministic-)function classes, let $\mathcal{F}_{\text{SFE}}^{\mathcal{C}_1}, \ldots, \mathcal{F}_{\text{SFE}}^{\mathcal{C}_M}$ be oracle-aided secure function evaluation functionalities, and let $t < n/2$. Let $D_1, \ldots, D_M$ be distributions, such that for every $j \in [M]$, the round complexity of $\mathcal{W}_{\text{flex}}^{D_j}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_j})$ has expectation μ_j. Denote $\mu = \max\{\mu_1, \ldots, \mu_M\}$.*

Then, $\mathcal{W}_{\text{flex}}^{D}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_1} \| \cdots \| \mathcal{F}_{\text{SFE}}^{\mathcal{C}_M})$, for some distribution D with expectation $\mu' = O(\mu)$, can be UC-realized by an FBB protocol in the $(\mathcal{F}_{\text{SMT}}, \mathcal{W}_{\text{flex}}^{D_1}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_1}), \ldots, \mathcal{W}_{\text{flex}}^{D_M}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_M}))$-hybrid model, with information-theoretic security, in the presence of an adaptive, semi-honest t-adversary, assuming that all honest parties receive their inputs at the same round.

In particular, if for every $j \in [M]$, the expectation μ_j is constant, then μ' is constant.

The proof of Theorem 6 can be found in the full version.

4 Round-Preserving Parallel Composition: Active Security

In this section, we consider security against active adversaries. First, in Section 4.1, we show how to compute the parallel composition of probabilistic-termination functionalities, in a round-preserving manner, using a black-box access to *protocols* realizing the individual functionalities. In Section 4.2, we investigate the question of whether there exists a *functionally* black-box round-preserving malicious protocol for the parallel composition of probabilistic functionalities, and show that for a natural extension of protocols, following the techniques from [5], this is not the case – i.e., there exist functions such that no such protocol with black-box access to them can compute their parallel composition, in a round-preserving manner, tolerating even a single adversarial party.

4.1 Feasibility of Round-Preserving Parallel Composition

In this section, we show how to compile multiple protocols, realizing probabilistic-termination functionalities, into a single protocol that realizes the parallel composition of the functionalities, in a round-preserving manner, and while only using black-box access to the underlying protocols. We start by providing a high-level description of the compiler.

The compiler receives as input protocols $\pi_1, \ldots, \pi_M$, where each protocol π_j is defined in the point-to-point model, in which the parties are given correlated randomness in a secure setup phase, i.e., in the $(\mathcal{F}_{\text{SMT}}, \mathcal{F}_{\text{CORR}}^{D_j^{\text{corr}}})$-hybrid model.[7] It follows that the next-message function for each party in each protocol is a deterministic function that receives the input

[7] This captures, for example, broadcast-model protocols, where the broadcast channel is realized using an expected-constant-round protocol.

value, correlated randomness, private randomness and history of incoming messages, and outputs a vector of n messages to be sent in the following round (one message for each party). In particular, we note that the entire transcript of the protocol is fixed once the input value, correlated randomness and private randomness of each party are determined.

The underlying ideas of the compiler are inspired by the constructions in [5, 14], where a round-preserving parallel-broadcast protocol was constructed by iteratively running, for a constant number of rounds, multiple instances of BA protocols (each instance is executed multiple times in parallel), until at least one execution of every BA instance is completed. This approach is suitable for computing "privacy-free" functionalities, where the adversary may learn the results of multiple computations using the same inputs for honest parties, without compromising security. However, when considering the parallel composition of arbitrary functions, running two instances of a protocol *using the same inputs* will violate privacy, since the adversary can use different inputs to learn multiple outputs of the function.

The parallel-composition compiler generalizes the above approach in a privacy-preserving manner. The compiler follows the GMW paradigm [30] and is defined in the Setup-Commit-then-Prove hybrid model [10, 39], which generates committed correlated randomness for the parties and ensures that all parties follow the protocol specification. This mechanism allows each party to commit to its input values and later execute multiple instances of each protocol, while proving that the same input value is used in all executions. For simplicity and without loss of generality, we assume that each function is deterministic and has a public output. In this case it is ensured that if two parties receive output values in two executions of π_j, then they receive the same output value. The private random coins that are used in each execution only affect the termination round, but not the output value. Using this simplification, we can remove the leader-election phase from the output-agreement technique in [5, 14] and directly use the termination technique from Bracha [7].

Another obstacle is to recover from corruptions without increasing the round complexity. Indeed, in case some party misbehaves, e.g., by using different input values in different instances of the same protocol π_j, then the Setup-Commit-then-Prove functionality ensures that all honest parties will identify the cheating party. In this case, the parties cannot recover by, for example, backtracking and simulating the cheating party, as this will yield a round complexity that is linear in the number of parties. Furthermore, the protocol must resume in a way such that all instances of a specific protocol π_j will use the same input value that the identified corrupted party used throughout the protocol's execution until it misbehaved (since the cheating party might have learned an output value in one of the executed protocols).

To this end, we slightly adjust the Setup-Commit-then-Prove functionality and secret-share every committed random string r_i (the correlated randomness for party P_i) among all the parties, using an error-correcting secret-sharing scheme. Note that this can be done information theoretically as we assume an honest majority [50, 19, 11]. In case a cheating party is identified, every party broadcasts the share of the committed randomness corresponding to that party, reconstructs this party's correlated randomness and from that point onwards, locally computes the messages corresponding to this party in every instance of every protocol. Using this approach, every round in the original protocols $\pi_1, \ldots, \pi_{\mathtt{M}}$ is expanded by a constant number of rounds, and the overall round complexity is preserved.

▶ **Theorem 7.** *Let $\mathcal{F}_1, \ldots, \mathcal{F}_M$ be CSFs, let $t < n/2$, and let $c \geq 1$. Let $\pi_1, \ldots, \pi_M$ be SNF protocols such that for every $j \in [M]$, protocol π_j UC-realizes $\mathcal{W}_{\text{flex}}^{D_j}(\mathcal{F}_j)$ with expected round complexity μ_j and information-theoretic security, in the $(\mathcal{F}_{\text{SMT}}, \mathcal{F}_{\text{CORR}}^{D_j^{\mathsf{corr}}})$-hybrid model (for a distribution D_j and a distribution D_j^{corr} in NC^0), in the presence of an adaptive, malicious t-adversary, assuming that all honest parties receive their inputs at the same round. Denote $\mu = \max\{\mu_1, \ldots, \mu_M\}$.*

Then, $\mathcal{W}^{D,c}_{\text{sl-flex}}(\mathcal{F}_1 \parallel \cdots \parallel \mathcal{F}_M)$*, for some distribution* D *with expectation* $\mu' = O(\mu)$*, can be UC-realized with information-theoretic security by a protocol* π *in the* $(\mathcal{F}_{\text{SMT}}, \mathcal{F}_{\text{CORR-BC}})$*-hybrid model, in the same adversarial setting, assuming that all honest parties receive their inputs within* $c+1$ *consecutive rounds. In addition, protocol* π *requires only black-box access to the protocols* π_j*.*

In particular, if for every $j \in [M]$*,* μ_j *is constant, then* D *has constant expectation.*

The proof of Theorem 7 can be found in the full version.

4.2 An Impossibility of FBB Round-Preserving Parallel Composition

In this section, we prove that for a natural class of protocols, following and/or extending in various ways the techniques from Ben-Or and El-Yaniv [5],[8] there exist functions such that no protocol can compute their parallel composition in a round-preserving manner, while accessing the functions in a black-box way, tolerating even a single adversarial party. Although this is not a general impossibility result, it indicates that the batching approach of [5] is limited to semi-honest security (cf. Section 3) and/or functionally white-box transformations.

We observe that this impossibility serves as an additional justification for the optimality of our protocol-black-box parallel composition (cf. Section 4.1). Indeed, on the one hand, it formally confirms the generic observation that the natural parallel composition of a set of PT functionalities does not preserve their round complexity. On the other hand, and most importantly, it proves that all existing techniques for composing PT functionalities in parallel in the natural (FBB) manner fail in preserving the round complexity. Hence, the only known existing round-preserving composition for such functionalities are the protocol-black-box compiler presented in Section 4.1 or more inefficient non-black-box techniques. The wideness of the class of excluded protocols by our impossibility result justifies our conjecture that there exists no round-preserving FBB protocol for parallel composition of PT functionalities. Proving this conjecture is in our opinion a very interesting research direction.

We first argue informally why the approach of [5], cannot be directly extended to privacy-sensitive functions. The idea in [5] for allowing each of the n parties to broadcast its value is to have each of the n parties participate in $m = O(\log n)$ parallel invocations (hereafter called *batches*, to avoid confusion with the goal of parallel broadcast for different messages) of broadcast as sender with the same input. Each of those batches is executed in parallel for a fixed (constant) number of rounds (for the same broadcast message); this increases the probability that sufficiently many parties receive output from each batch. At the end of each batch execution, the parties check whether they jointly hold the output, and if not, they repeat the computation of the batches. It might seem that this idea can be applied to arbitrary tasks, but this is not the case. The reason is that this idea fails if the functionality has any privacy requirements, is that the adversary can input different values on different calls of the functionality within a batch and learn more information on the input.

Batched parallel composition. The above issue with privacy appears whenever a function is invoked twice in the same round on the same inputs from honest parties. Indeed, in this case the adversary can use different inputs to each invocation and learn information as sketched above. The same attack can be extended to composition-protocols which invoke the function in two different rounds ρ and ρ'; as long as the adversary knows these rounds

[8] To our knowledge, the only known techniques for round-preserving parallel composition are those of Ben-Or and El-Yaniv [5] and are only for the specific case of Byzantine agreement.

he can still launch the above attack on privacy. Generalizing the claim even further, for specific classes of functions, it suffices that there are two (possibly different) functions which are evaluated on the same inputs in rounds ρ and ρ'. This excludes protocols that might attempt to avoid using some functionality $\mathcal{W}_{\text{flex}}^{D_j}(\mathcal{F}_j)$ by invoking some other $\mathcal{W}_{\text{flex}}^{D_{j'}}(\mathcal{F}_{j'})$.

To capture the above generalization, we define the class of *batched-parallel composition* protocols: A protocol π implementing the PT parallel composition $\mathcal{W}_{\text{flex}}^{D}(\mathcal{F}_1 \parallel \cdots \parallel \mathcal{F}_{\mathtt{M}})$ in the $(\mathcal{W}_{\text{flex}}^{D_1}(\mathcal{F}_1), \ldots, \mathcal{W}_{\text{flex}}^{D_{\mathtt{M}}}(\mathcal{F}_{\mathtt{M}}))$-hybrid model (for some distributions $D, D_1, \ldots, D_{\mathtt{M}}$) is a *batched-parallel composition* protocol if it has the following structure: It proceeds in rounds, where in each round the protocol might initiate (possibly multiple) calls to any number of the hybrid functionalities $\mathcal{W}_{\text{flex}}^{D_j}(\mathcal{F}_j)$ and/or continue calls that were initiated in previous rounds. Furthermore, there exist two publicly known protocol rounds ρ and ρ', and indices $j, j', \ell \in [\mathtt{M}]$, such that for the input vector $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ that π gives to $\mathcal{W}_{\text{flex}}^{D}(\mathcal{F}_1 \parallel \cdots \parallel \mathcal{F}_{\mathtt{M}})$ (where $\boldsymbol{x}_i = (x_i^1, \ldots, x_i^{\mathtt{M}})$) the following properties are satisfied:

1. In round ρ the functionality $\mathcal{W}_{\text{flex}}^{D_j}(\mathcal{F}_j)$ is called on input $\boldsymbol{x}^\ell = (x_1^\ell, \ldots, x_n^\ell)$ and at least two of its rounds are executed.
2. In round ρ' the functionality $\mathcal{W}_{\text{flex}}^{D_{j'}}(\mathcal{F}_{j'})$ is also called on input $\boldsymbol{x}^\ell$ and at least two of its rounds are executed.

We next show that the there are classes of functions $\mathcal{C}_1, \ldots, \mathcal{C}_{\mathtt{M}}$ such and for any protocol π that securely computes the parallel composition $\mathcal{W}_{\text{flex}}^{D}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_1} \parallel \cdots \parallel \mathcal{F}_{\text{SFE}}^{\mathcal{C}_{\mathtt{M}}})$ while given hybrid access to PT functionalities $\mathcal{W}_{\text{flex}}^{D_i}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_i})$ the following properties hold *simultaneously*:

1. π has to call each of the hybrids $\mathcal{W}_{\text{flex}}^{D_i}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_i})$ (for at least 2 rounds each).[9]
2. The naïve solution of π calling each of the $\mathcal{W}_{\text{flex}}^{D_i}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_i})$'s in parallel until they terminate is not round-preserving (for an appropriate choice of D_i's.)
3. π cannot be a batched-parallel composition protocol.

The above shows that the classes $\mathcal{C}_1, \ldots, \mathcal{C}_{\mathtt{M}}$ not only exclude the existence of a batched-parallel composition protocol, but they also exclude all other known solutions. This implies that for this classes of functions, every known approach – and generalizations thereof – fail to compute the parallel composition of the corresponding functionality in an FBB and round-preserving manner. In the full version we prove the following theorem.

▶ **Theorem 8.** *Let $\mathtt{M} = O(\kappa)$. There exist n-party function classes $\mathcal{C}_1, \ldots, \mathcal{C}_{\mathtt{M}}$ and distributions $D_1, \ldots, D_{\mathtt{M}}$, such that the following properties hold in the presence of a malicious adversary corrupting any one of the parties:*

1. *The protocol that calls each $\mathcal{W}_{\text{flex}}^{D_i}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_i})$ in parallel (once) until termination is not round-preserving (its round complexity is asymptotically higher than of the distributions D_i).*
2. *Any $(\mathcal{F}_{\text{SMT}}, \mathcal{W}_{\text{flex}}^{D_1}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_1}), \ldots, \mathcal{W}_{\text{flex}}^{D_{\mathtt{M}}}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_{\mathtt{M}}}))$-hybrid protocol for computing $\mathcal{W}_{\text{flex}}^{D}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_1} \parallel \cdots \parallel \mathcal{F}_{\text{SFE}}^{\mathcal{C}_{\mathtt{M}}})$ has to make a meaningful call (i.e., a call that executes at least two rounds) to each PT hybrid $\mathcal{W}_{\text{flex}}^{D_i}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_i})$.*
3. *There exists no functionally black-box batched-parallel composition protocol for computing $\mathcal{W}_{\text{flex}}^{D}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_1} \parallel \cdots \parallel \mathcal{F}_{\text{SFE}}^{\mathcal{C}_{\mathtt{M}}})$ in the $(\mathcal{F}_{\text{SMT}}, \mathcal{W}_{\text{flex}}^{D_1}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_1}), \ldots, \mathcal{W}_{\text{flex}}^{D_{\mathtt{M}}}(\mathcal{F}_{\text{SFE}}^{\mathcal{C}_{\mathtt{M}}}))$-hybrid model, where D has (asymptotically) the same expectation as $D_1, \ldots, D_{\mathtt{M}}$.*

References

1 Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and

[9] Note that this does not mean that π is not round preserving as the calls might be in parallel.

interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 483–501. Springer, April 2012.

2 Gilad Asharov and Yehuda Lindell. Utility dependence in correct and fair rational secret sharing. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 559–576. Springer, August 2009.

3 Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *22nd ACM STOC*, pages 503–513. ACM Press, May 1990.

4 Michael Ben-Or. Another advantage of free choice: Completely asynchronous agreement protocols (extended abstract). In Robert L. Probert, Nancy A. Lynch, and Nicola Santoro, editors, *2nd ACM PODC*, pages 27–30. ACM Press, August 1983.

5 Michael Ben-Or and Ran El-Yaniv. Resilient-optimal interactive consistency in constant time. *Distributed Computing*, 16(4):249–262, 2003.

6 Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, May 1988.

7 Gabriel Bracha. An asynchronou [(n-1)/3]-resilient consensus protocol. In Robert L. Probert, Nancy A. Lynch, and Nicola Santoro, editors, *3rd ACM PODC*, pages 154–162. ACM Press, August 1984.

8 Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

9 Ran Canetti, Joe Kilian, Erez Petrank, and Alon Rosen. Black-box concurrent zero-knowledge requires omega (log n) rounds. In *33rd ACM STOC*, pages 570–579. ACM Press, July 2001.

10 Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, May 2002.

11 Alfonso Cevallos, Serge Fehr, Rafail Ostrovsky, and Yuval Rabani. Unconditionally-secure robust secret sharing with compact shares. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 195–208. Springer, April 2012.

12 David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *20th ACM STOC*, pages 11–19. ACM Press, May 1988.

13 Kai-Min Chung, Rafael Pass, and Wei-Lung Dustin Tseng. The knowledge tightness of parallel zero-knowledge. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 512–529. Springer, March 2012.

14 Ran Cohen, Sandro Coretti, Juan A. Garay, and Vassilis Zikas. Probabilistic termination and composability of cryptographic protocols. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 240–269. Springer, August 2016.

15 Ran Cohen, Sandro Coretti, Juan A. Garay, and Vassilis Zikas. Round-preserving parallel composition of probabilistic-termination cryptographic protocols. Cryptology ePrint Archive, Report 2017/364, 2017. URL: `http://eprint.iacr.org/2017/364`.

16 Ran Cohen, Iftach Haitner, Eran Omri, and Lior Rotem. Characterization of secure multiparty computation without broadcast. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 596–616. Springer, January 2016.

17 Ran Cohen and Yehuda Lindell. Fairness versus guaranteed output delivery in secure multiparty computation. In *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 466–485. Springer, December 2014.

18 Ronald Cramer, Ivan Damgård, Stefan Dziembowski, Martin Hirt, and Tal Rabin. Efficient multiparty computations secure against an adaptive adversary. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 311–326. Springer, May 1999.

19 Ronald Cramer, Ivan Damgård, and Serge Fehr. On the cost of reconstructing a secret, or VSS with optimal reconstruction phase. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 503–523. Springer, August 2001.

20 Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 378–394. Springer, August 2005.

21 Ivan Damgård and Jesper Buus Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 432–450. Springer, August 2000.

22 Danny Dolev, Rüdiger Reischuk, and H. Raymond Strong. Early stopping in byzantine agreement. *Journal of the ACM*, 37(4):720–741, 1990.

23 Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.

24 Pesech Feldman and Silvio Micali. An optimal probabilistic protocol for synchronous byzantine agreement. *SIAM Journal on Computing*, 26(4):873–933, 1997.

25 Michael J. Fischer and Nancy A. Lynch. A lower bound for the time to assure interactive consistency. *Information Processing Letters*, 14(4):183–186, 1982.

26 Matthias Fitzi and Juan A. Garay. Efficient player-optimal protocols for strong and differential consensus. In Elizabeth Borowsky and Sergio Rajsbaum, editors, *22nd ACM PODC*, pages 211–220. ACM Press, July 2003.

27 Georg Fuchsbauer, Jonathan Katz, and David Naccache. Efficient rational secret sharing in standard communication networks. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 419–436. Springer, February 2010.

28 Juan A. Garay, Jonathan Katz, Ueli Maurer, Björn Tackmann, and Vassilis Zikas. Rational protocol design: Cryptography against incentive-driven adversaries. In *54th FOCS*, pages 648–657. IEEE Computer Society Press, October 2013.

29 Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure MPC from indistinguishability obfuscation. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 74–94. Springer, February 2014.

30 Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

31 S. Dov Gordon, Feng-Hao Liu, and Elaine Shi. Constant-round MPC with fairness and guarantee of output delivery. In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 63–82. Springer, August 2015.

32 Adam Groce and Jonathan Katz. Fair computation with rational players. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 81–98. Springer, April 2012.

33 Iftach Haitner. A parallel repetition theorem for any interactive argument. In *50th FOCS*, pages 241–250. IEEE Computer Society Press, October 2009.

34 Joseph Y. Halpern and Vanessa Teague. Rational secret sharing and multiparty computation: Extended abstract. In László Babai, editor, *36th ACM STOC*, pages 623–632. ACM Press, June 2004.

35 Johan Håstad, Rafael Pass, Douglas Wikström, and Krzysztof Pietrzak. An efficient parallel repetition theorem. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 1–18. Springer, February 2010.

36 Yuval Ishai, Ranjit Kumaresan, Eyal Kushilevitz, and Anat Paskin-Cherniavsky. Secure computation with minimal interaction, revisited. In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 359–378. Springer, August 2015.

37 Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 21–30. ACM Press, June 2007.

38 Yuval Ishai, Eyal Kushilevitz, and Anat Paskin. Secure multiparty computation with minimal interaction. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 577–594. Springer, August 2010.

39 Yuval Ishai, Rafail Ostrovsky, and Vassilis Zikas. Secure multi-party computation with identifiable abort. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 369–386. Springer, August 2014.

40 Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer – efficiently. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 572–591. Springer, August 2008.

41 Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 445–462. Springer, August 2006.

42 Jonathan Katz, Ueli Maurer, Björn Tackmann, and Vassilis Zikas. Universally composable synchronous computation. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 477–498. Springer, March 2013.

43 Joe Kilian. Founding cryptography on oblivious transfer. In *20th ACM STOC*, pages 20–31. ACM Press, May 1988.

44 Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

45 Silvio Micali. Fast and furious byzantine agreement. In *ITCS 2017*, January 2017.

46 Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016*, volume 9666 of *LNCS*, pages 735–763. Springer, May 2016.

47 Shien Jin Ong, David C. Parkes, Alon Rosen, and Salil P. Vadhan. Fairness with an honest minority and a rational majority. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 36–53. Springer, March 2009.

48 Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, 1980.

49 Michael O. Rabin. Randomized byzantine generals. In *24th FOCS*, pages 403–409, November 1983.

50 Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *21st ACM STOC*, pages 73–85. ACM Press, May 1989.

51 Mike Rosulek. Must you know the code of f to securely compute f? In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 87–104. Springer, August 2012.

52 Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, November 1982.