Hello, Goodbye: Concurrent General Composition From Super-Rushing Security *

Anirudh Chandramouli
Bar-Ilan University, Israel

Anirudh.Chandramouli@biu.ac.il

October 29, 2026

Abstract

Kushilevitz, Lindell, and Rabin (STOC '06, SIAM Journal on Computing '10), in their seminal result, showed that any standalone-secure synchronous multiparty computation (MPC) protocol can be compiled into one secure under concurrent general composition, by adding a synchronisation step after every round; thereby, at least doubling the round complexity. We show that for protocols that are secure against *super-rushing* adversaries, introduced by Asharov, Chandramouli, Cohen, and Ishai (EuroCrypt '25), two synchronisation rounds suffice. Specifically, for an R-round protocol that is secure against super-rushing adversaries with a black-box, straight-line simulator, adding a round of *start synchronisation* (saying "Hello") and a round of *end synchronisation* (saying "Goodbye") yields security under concurrent general composition with *selective* abort in $R+2$ rounds. If the end synchronisation is over a broadcast channel, the security is upgraded to *unanimous* abort. Both compilers incur no loss in the security reduction and preserve the flavour of security (perfect, statistical, or computational) of the underlying synchronous protocol. Moreover, combining with the sufficient conditions of Asharov et al. for perfect security, and the round-preserving super-rushing compilers of Ciampi, Ravi, and Zou (Cryptology ePrint Archive 2026/1745) for statistical and computational security, yields compilers from *rushing* security to security under concurrent general composition, with an additive overhead of at most 2 rounds.

We further extend our compilers to protocols with a trusted setup, such as a common reference string or a public-key infrastructure, which is necessary in the dishonest majority setting. We require two conditions. First, every execution of the protocol uses its own *stateless* copy of the setup, whose responses to the adversary do not depend on the interactions of the honest parties. In particular, this rules out setups that are shared across executions, such as a global common reference string or a single public-key infrastructure used by every execution. Second, the underlying protocol is secure against super-rushing adversaries with *adaptive inputs*, that is, even when the inputs are chosen after interacting with the setup. Under these conditions, the same compilers yield security under concurrent general composition. We also show that adaptive inputs cannot be relaxed to the usual notion of security in the setup-hybrid model. Specifically, we give a protocol that is statistically secure against super-rushing adversaries with guaranteed output delivery under the standard hybrid model, but is insecure under concurrent general composition after adding start and end synchronisation, even with selective abort.

1 Introduction

Secure multiparty computation (MPC) enables n mutually distrusting parties to jointly compute a function of their private inputs, such that nothing is revealed beyond the prescribed output, in the presence of an adversary that may corrupt some of the parties. Classical feasibility results [Yao86, GMW87, BGW88, CCD88, RB89] show that any n-ary functionality that can be efficiently computed can also be *securely*

*A preliminary draft of this work, was prepared when the author was a PhD student at Bar-Ilan University. Research supported by the European Union (ERC, FTRC, 101043243). Views and opinions expressed are however those of the author only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

computed. These feasibility results and the bulk of the literature on secure MPC typically assume that the parties are connected pair-wise by authenticated and *synchronised* channels and that the parties have access to an *atomic* broadcast channel. Furthermore, the security of these protocols is typically established for the case when the protocols are executed in a *standalone* manner, i.e., in an isolated environment and independent of any external computations.

Real-world environments in which secure computation protocols may be deployed, rarely offer the comforts of synchrony and isolation. A protocol may be executed over networks with arbitrary message delays and in conjunction with a variety of other protocols that may or may not be secure themselves. Moreover, in such real-world settings, the delays and external computations may even be malicious. Hence, a desirable property of secure protocols is that of *concurrent general composition*, where protocols are executed over networks with arbitrary (and potentially malicious) delays and in conjunction with arbitrary (and potentially malicious) computations. The seminal universal composability (UC) framework, introduced by Canetti [Can01], considers exactly such a robust notion of security. In the UC framework, the primary adversarial entity is the *environment* that represents any external influence on the execution of the protocol. A UC-secure protocol remains secure under composition with *any* other protocol, run any number of times, and over a network under complete control by an adversary. Moreover, Lindell [Lin03b] showed that the definition of security in the UC framework is in fact equivalent to a direct notion of security under *concurrent general composition*.

The cost of concurrent general composition. Perhaps unsurprisingly, UC security (correspondingly, security under concurrent general composition[1]) does not come for free. In the *plain* model (without any trusted setup), it is impossible to UC-securely compute broad classes of two-party functionalities [KL11]. Additionally, different trust assumptions [Can07, GGJS11] have been studied, to identify the minimal trust assumptions required to achieve UC security. A natural question is thus

"*Under what conditions does standalone synchronous security imply security under concurrent general composition?*"

The seminal work of Kushilevitz, Lindell, and Rabin [KLR06, KLR10] characterised security of standalone-secure synchronous protocols in the setting of concurrent general composition. The key insight from [KLR10] was that perfect security with straight-line black-box simulation is sufficient for concurrent general composition. However, contrary to folklore belief, [KLR10, Section 2] modifies a synchronous protocol to make it secure under concurrent execution: after each round r of a synchronous protocol, say ρ, the parties are required to additionally perform a "synchronisation step". In the synchronisation step, the parties confirm the receipt of their prescribed round r messages and proceed to round $r+1$ only upon receiving the confirmation receipts from all parties. The synchronisation step ensures that when the (modified) synchronous protocol, say $\widehat{\rho}$, is executed concurrently with the adversary controlling message delivery, once a party proceeds to a round r, all parties (and particularly all non-faulty parties) must have already completed *all* the previous rounds of the protocol. Although, the synchronisation steps yield a very strong security guarantee (security under concurrent general composition), the round complexity of the underlying protocol is doubled. Motivated by this, we revisit the question of sufficient conditions for security under concurrent general composition.

1.1 Our Results

A closely related notion to security under concurrent execution of a synchronous protocol is that of *super-rushing* introduced by [ACCI25]. Super-rushing security considers *optimistic* executions of synchronous protocols, that is, executions where the implicit instruction is the following: proceed to round $r+1$ upon receiving all prescribed messages corresponding to round r. Typically, implementations of synchronous protocols avoid waiting for the round-trip timeout (and moreover, avoid designing such a timeout under arbitrary network conditions) by adding the optimistic instruction. Our key observation is the following:

"*Super-rushing is well-behaved concurrency.*"

[1] We use the terms UC security and security under concurrent general composition, interchangeably. Note that [Lin03b] establishes their equivalence.

Specifically, the modelling of super-rushing security allows adversaries to "peek into the future", but restricts the adversaries to always allow honest-to-honest communication and guarantee corrupt-to-honest communication. Removing these restrictions on the adversary immediately induces a concurrent execution of a synchronous protocol: parties proceed to the round r only after receiving their messages corresponding to all rounds $\leq r-1$; but if the adversary blocks any messages, the parties just do not proceed.

On security with abort. We consider the setting of concurrent general composition, where the adversary has complete control over message delivery, including the ability to indefinitely block messages between honest parties. Naturally, such a strong adversary, may mount denial-of-service attacks that prevents the honest parties from obtaining outputs. Moreover, we consider the strongest setting of a *dishonest majority*, that is, the adversary may corrupt up to $t < n$ parties. Cleve's classical impossibility result [Cle86] rules out fairness of output delivery in this setting and the best guarantee that one can hope for is some flavour of *abort*: honest parties may asymmetrically be left without output (*selective* abort), or the parties are instead guaranteed to agree on whether the computation aborted at all (*unanimous* abort) [GL05, CGZ20].

1.1.1 Composing Protocols in the Plain Model

Inspired by our observation, we show that security in the standalone setting against *super-rushing* adversaries, with straight-line black-box simulation, suffices to obtain security under concurrent general composition, at the cost of only *two* additional rounds: a round of *start synchronisation*, where every party announces that it has begun the protocol and waits for the announcements of all parties, and a round of *end synchronisation*, where every party announces that it has completed the protocol and outputs only upon receiving the announcements of all parties.

Theorem 1.1 (informal, main theorem)**.** *Let f be an n-party functionality and let ρ be an R-round synchronous protocol such that*

1. *ρ computes f against **super-rushing** adversaries with perfect security (resp. statistical, computational); and*
2. *The underlying simulator for ρ's security is **black-box** and **straight-line**.*

Let $\widehat{\rho}$ be the $(R+2)$-round protocol obtained by adding start and end synchronisation to ρ. Then, $\widehat{\rho}$ computes f under concurrent general composition with perfect (resp. statistical, computational) security and selective abort. If the end synchronisation is over an atomic broadcast channel, then $\widehat{\rho}$ computes f under concurrent general composition with unanimous abort.

The underlying protocol ρ need not guarantee output delivery: security against super-rushing adversaries with selective abort suffices for the first part of Theorem 1.1, and with unanimous abort for the second part. However, the converse is not true: a protocol that offers guaranteed output delivery against super-rushing adversaries, is only secure with selective/unanimous abort under concurrent general composition. As noted before, this is the best guarantee achievable against an adversary that has complete control over the network, and may indefinitely block messages. In essence, the distinguishing conditions of Theorem 1.1, in comparison with [KLR10] are the following: super-rushing security + two synchronisation rounds ($R+2$ rounds) vs rushing security + a synchronisation step after every round ($2R$ rounds). We refer the reader to Section 4, and specifically to Theorem 4.2 and Theorem 4.4, for the formal statements.

Super-rushing breaks sequential composition! The standard setting of multi-party computation with security against rushing adversaries [Can00] in the standalone setting implies *sequential composition* [Can00, Gol04], that is, any rushing secure protocol may be sequentially composed with any other protocol without losing security. Perhaps surprisingly, [ACCI25] showed that super-rushing security is not preserved under sequential composition: the sequential composition of two single-input protocols, each secure against super-rushing adversaries, need not be secure against super-rushing adversaries! In essence, super-rushing security captures security under a *standalone* concurrent execution. In fact, the proof of Theorem 1.1 is obtained by first showing that security against super-rushing adversaries yields security of the underlying protocol in a standalone concurrent execution with selective abort (see Section 4.1). Security under concurrent general

composition is given by appending start and end synchronisation; which is philosophically similar to the compiler of [KLR10], which performs the synchronisation for each round.

1.1.2 Composing Protocols with Setup

Our compiler, Theorem 1.1, is stated in the plain model, i.e, in the absence of any trusted setup and suffices for protocols in the setting of honest majority. However, in the setting of $t < n$, black-box straight-line simulation in the plain model is impossible for broad classes of functionalities [CF01, CKL06]. As a consequence, protocols in the dishonest majority setting typically rely on a *setup*, such as a common reference string (CRS), a random oracle (RO), or a public-key infrastructure (PKI). In other words, while Theorem 1.1 yields a compiler for computationally-secure protocols, it does not resolve the question of obtaining security under concurrent general composition in the dishonest majority setting. Therefore, it is natural to consider identical pre-conditions as in Theorem 1.1 (super-rushing security with black-box straight-line simulation), but instead for protocols ρ in the setup hybrid model.

We show that the same compiler (saying HELLO and GOODBYE) yields security under composition for protocols with a setup, with two additional requirements: (i) the setup is local to each execution of ρ; and (ii) the super-rushing security of ρ holds with *adaptive inputs*, i.e., even when the environment chooses the inputs of the parties after interacting with the setup.

Theorem 1.2 (informal, compiler with setup)**.** *Let f be an n-party functionality, let $\mathcal{G}$ be an ideal functionality, and let ρ be an R-round synchronous protocol in the $\mathcal{G}$-hybrid model such that*

1. *ρ computes f against **super-rushing** adversaries with statistical security (resp. computational) and **adaptive inputs**;*
2. *The underlying simulator for ρ's security is **black-box** and **straight-line**;*
3. *Each execution of ρ invokes a **localised** copy of $\mathcal{G}$; and*
4. *$\mathcal{G}$ is stateless.*

Let $\widehat{\rho}$ be the $(R+2)$-round protocol obtained by adding start and end synchronisation to ρ, where every execution of $\widehat{\rho}$ uses its own localised and stateless copy of $\mathcal{G}$. Then, $\widehat{\rho}$ computes f under concurrent general composition with statistical (resp. computational) security and selective abort. If the end synchronisation is over an atomic broadcast channel, then $\widehat{\rho}$ computes f under concurrent general composition with unanimous abort.

On localised setups. We assume a fresh copy of the setup for every execution of ρ, which is the standard (non-global) setting of the UC framework [Can01]. While such an assumption makes the analysis easier, we note that some restriction of this form is necessary in general; for instance, [LLR06] show that authenticated Byzantine agreement does not compose once several instances share the same signature setup. Such setups exclude *global* notions of setup that are shared across executions and with other arbitrary protocols [CDPW07], such as a single common reference string used by every execution. We leave the question of extending our compilers to protocols with global setups open.

On stateless setups. We say that the setup, $\mathcal{G}$, is *stateless* if the interactions of the honest parties with $\mathcal{G}$ do not affect the interactions of the adversary (or corrupted parties) with $\mathcal{G}$. Intuitively, Statelessness stems from the gap between the concurrent and the super-rushing adversary: a concurrent adversary may stall some honest parties indefinitely, whereas our reduction to super-rushing security completes the execution of ρ on their behalf, and the resulting interactions with the setup must remain invisible to the adversary and the environment, who may continue to interact with the setup after the execution.

We refer the reader to Section 5, and specifically to Theorem 5.2 and Theorem 5.5, for the formal statements. The relevant definitions of localised and stateless setups can be found in Section 3.6.

A separation: adaptive inputs. The usual notion of security in the setup-hybrid model fixes the inputs of the parties independently of the setup. One may hope that this suffices for Theorem 1.2, since start synchronisation fixes the inputs of the honest parties to ρ before any message of ρ is sent. However, under general composition, the adversary may interact with the setup before the start synchronisation completes, and influence the inputs of the honest parties to ρ, via the calling protocol, as a function of the responses of the setup. We show that this is not an artefact of our proof.

Theorem 1.3 (informal, separation)**.** *Let $\mathcal{G}$ be any **localised** and **stateless** setup that returns a fixed value of high min-entropy. There exist a two-party functionality f and a single-round protocol ρ_{na} in the $\mathcal{G}$-hybrid model such that*

1. *ρ_{na} computes f against super-rushing adversaries with **guaranteed output delivery**, statistical security and a **black-box** and **straight-line** simulator, under the usual notion of security in the $\mathcal{G}$-hybrid model; and*
2. *$\widehat{\rho}_{\mathsf{na}}$ (with* HELLO *and* GOODBYE *messages) does not compute f under 1-bounded concurrent general composition, even with selective abort.*

The protocol ρ_{na} satisfies all the required preconditions of Theorem 1.2, except for security with *adaptive inputs*. Therefore, the notion of security with adaptive inputs is necessary for obtaining security with concurrent general composition. We refer the reader to Theorem 5.7 for the formal details.

1.1.3 Compiling from Rushing Security

Theorem 1.1 and Theorem 1.2 establish composition from protocols that are secure against super-rushing adversaries. Hence, contrasting them against the [KLR10] compiler would be inaccurate as [KLR10] showed composition from the weaker, yet more standard notion of security against rushing adversaries. We note that our compilers may be also applied to protocols that are secure against rushing adversaries to obtain composition directly.

Perfect security. The sufficient conditions of [ACCI25] for obtaining perfect super-rushing security from perfect rushing security for free, yields compilers upgrading *rushing* security to security under concurrent general composition.

Theorem 1.4 (informal, perfect security from rushing)**.** *Let f be an n-party functionality, and let ρ be an R-round synchronous protocol such that*

1. *ρ perfectly securely computes f against **rushing** adversaries, with a compatible simulator [ACCI25, Definition 6.1];*
2. *ρ has all-to-all communication;*
3. *ρ admits a committal round* CR *and an output revealing round* ORR*; and*
4. *$\mathsf{ORR} > \mathsf{CR} + 1$.*

Then, there is an $(R+2)$-round protocol that t-securely computes f under concurrent general composition with perfect security and selective abort. If the end synchronisation is over an atomic broadcast channel, then the protocol t-securely computes f under concurrent general composition with perfect security and unanimous abort.

Theorem 1.4 follows by composing Theorem 4.2 (resp. Theorem 4.4) with the sufficient conditions of [ACCI25]. The above conditions may be contrasted against the sufficient conditions of Dodis and Micali [DM00], who showed that a committal round and straight-line black-box simulation security against rushing adversaries, yields security under parallel composition (without adding any communication). Our result shows that adding two rounds yields concurrent general composition.

Statistical and computational security. For statistical and computational security, [CRZ26] give an explicit, round-preserving compiler from rushing to super-rushing security. Composing their compiler with Theorem 1.1 yields compilers from statistical rushing security with the same overhead of rounds.

Theorem 1.5 (informal, statistical security [CRZ26])**.** *Let f be an n-party functionality, and let ρ be an R-round synchronous protocol in the plain model that t-securely computes f with statistical (resp. computational) security against **rushing** adversaries, with a black-box straight-line simulator in the plain model. Then there is an $(R+2)$-round protocol that t-securely computes f under concurrent general composition with statistical security and selective abort. If the end synchronisation is over an atomic broadcast channel, then the protocol t-securely computes f under concurrent general composition with statistical (resp. computational) security and unanimous abort.*

Note that the hypothesis of Theorem 1.5 is not satisfied for every $t < n$: black-box straight-line simulation in the plain model is impossible for broad classes of functionalities once $t \geq n/2$ [CF01, CKL06], and in this regime, the underlying protocol ρ relies on a trusted setup. For such protocols, the analogous statement follows from Theorem 1.2 instead. We note that one of the compilers of [CRZ26], the compiler for security with *identifiable abort*, itself makes use of setup. Since, we do not capture identifiable abort for concurrent general composition, and furthermore, since the best guarantee achievable in the setting of concurrent composition is security with unanimous abort, this compiler is not relevant for our use.

1.2 Related Work

Composition of secure protocols. The general study of secure composition originates with [MR91, Can00], which formalised *sequential* composition. [PW00] gave the first treatment of *general* composition, with a composition theorem but no feasibility result. Universal composability (UC) [Can01] then extended this to the unbounded, concurrent case. [Lin03b] showed that UC security is equivalent to security under concurrent general composition. [LLR06] shows that composability can already be delicate for simpler primitives, such as authenticated Byzantine agreement, once several instances share a signature setup. [DM00] showed that perfectly secure protocols with a straight-line black-box simulator and no fixed-input assumption compose when executed in parallel (a restriction on the notion of concurrent general composition). [CCGZ21] achieves round-preserving parallel composition without a committal round, using the composed protocols only as black boxes. Interestingly, [BMU07] showed that black-box simulation for perfectly secure protocols may require rewinding in general, breaking the folklore belief that perfect security encapsulated the sufficient conditions for concurrent composition. A similar conclusion is reached, via a different route, by the *simulatable-security* framework [HU06]: statistical or perfect stand-alone simulatability already implies bounded concurrent *self*-composition, though this fails at the computational level. Unlike our compilers, this is a transfer theorem between abstract security notions rather than a construction, and it does not address concurrent *general* composition.

Other notions of secure composition. [CF01, CKL06] give broad impossibility results for UC security in the plain model, which, combined with [Lin03b]'s equivalence, extend to concurrent general composition; [KL11] extends this further to public-key models and fixed-input settings. Relaxing the security guarantee via super-polynomial-time simulation [Pas03, PS04, BS05, MMY06] showed that these impossibility results may be circumvented. These works are complementary to our approach, as we keep standard polynomial-time simulation and instead pay in rounds and security with abort. Other work studies how much trusted setup UC composability actually requires [Can07, BCNP04, GGJS11]. At the level of specific primitives, [DvdGMQ13, DMQN20] show that statistically secure bit commitments and oblivious transfer are automatically universally composable. Composition has also been studied for interactive oracle proofs [VW26], for information-theoretic t-certified protocols [CsHL+14], and for composing PKIs with the protocols that run over them [Her24]. Finally, concurrency is studied in the context of *self*-composition, where one protocol design controls every session: this includes concurrent zero-knowledge [GS14] and bounded-concurrent self-composition of general MPC without setup [Lin03a, PR03, Lin04, Pas04, MPR06].

Synchrony and universal composability. Another approach to concurrently composing synchronous protocols is to build synchrony directly into the composability framework [KMTZ13], which formalises syn-

chronous computation as a UC hybrid functionality. [LZM20] gives a synchronous composable framework within Constructive Cryptography, and [LZLM+20] studies obtaining *responsiveness* on top of it.

Timing assumptions instead of synchrony. [KLP05] achieves composability via loosely synchronised clocks rather than synchronisation rounds: full concurrent *self*-composition, and concurrent *general* composition when concurrently executing protocols have a bounded message delay. Moreover, they show that this restriction is necessary. Our compilers make the opposite trade-off: we do not assume anything about timing, and instead show secure concurrent composition of any protocol, in exchange for ≤ 2 extra rounds.

Asynchronous MPC with eventual message delivery. A long line of work, initiated by Ben-Or, Canetti, and Goldreich [BCG93], studies MPC over asynchronous networks where the adversary may arbitrarily delay messages, but every message is eventually delivered [BCG93, BKR94]. Since an honest party cannot distinguish a slow honest party from a silent corrupted one, the parties may only wait for messages from $n - t$ parties in each round. As a consequence, only the inputs of some core set of $n - t$ parties are taken into account in the computation, and the corruption threshold is limited to $t < n/3$ (resp. $t < n/4$ for perfect security) [BCG93, BKR94]. In contrast, we consider synchronous protocols tolerating any $t < n$ corruptions, executed over a network where messages may be blocked indefinitely, and settle for security with abort.

Asynchronous composability. [CFG+25] study whether asynchronous MPC protocols compose in *parallel*, which is a weaker guarantee than concurrent general composition. They show that black-box parallel composition is impossible in general, even against a single semi-honest corruption, in sharp contrast to the synchronous case, and obtain feasibility only for protocols with a specific three-phase structure and a compiler with some knowledge of it.

Round-optimal composable MPC. For computational security in the CRS model, two-round MPC protocols are known [GS18, BL18], which are UC-secure when instantiated with a two-round UC-secure oblivious transfer. Our compilers are decidedly worse with regards to round complexity, yielding a UC-secure protocol with > 3 rounds at least. Moreover, these constructions are analysed in a synchronous UC model, whereas our compilers assume an arbitrary network, at the price of security with abort. Our compiler on the other hand, is generic, and yields composition directly from security analysis in the standalone setting. Crucially, for information-theoretic protocols, our compiler yields a protocol that runs in $R + 2$ rounds, compared to $2R + 1$[2] rounds due to [KLR10].

Asynchronous and network-agnostic MPC. Protocols designed directly for asynchronous networks [BCG93, BKR94, CGHZ16], and network-agnostic protocols that are secure under synchrony and retain security (for a smaller corruption threshold) under asynchrony [BLZL20, LZLM+20], are tailored to the network model by design. In contrast, our compilers take an arbitrary synchronous protocol, without modifying its rounds, and run it over an adversarially scheduled network, at the cost of two additional rounds and settling for security with abort.

Use of AI

All technical ideas, results, and most of the written content in this work were produced by the author. Claude Code, was used in the capacity of a reviewer to identify editorial mishaps, to validate the correctness of the proofs, and to make certain edits of the manuscript such as rewriting sentences, fixing typos, refactoring and restructuring. The author takes full responsibility for the final content, especially for the errors and oversights.

[2]The KLR compiler requires start synchronisation and synchronisation after each round.

2 Technical Overview

In this section, we give a high level overview and sketch of our results. The complete details, including all formal statements, are presented in Section 4 for the plain model, and in Section 5 for protocols with setup. In both settings, our compiler is proven to be secure via two steps:

- **Security under concurrent execution.** We show that any super-rushing secure protocol is immediately secure under concurrent execution with *selective* abort, where the adversary has the additional ability to block messages indefinitely. Note that we are still in the standalone setting, that is, the protocol is executed in an isolated environment with a stronger adversary.
- **Security under concurrent composition.** We then show that adding start and end synchronisation lifts super-rushing security to full security under concurrent general composition.

2.1 Composing Protocols in the Plain Model

Reducing a concurrent execution to super-rushing. To isolate the effect of message scheduling from the effect of composing with an arbitrary external protocol, we first consider a *dummy composition* π_0: a protocol that does nothing but forward every party's input to a single ideal call to f, and return the result verbatim. Consider an adversary $\mathcal{A}$ for π_0^ρ that, beyond corrupting parties, may also completely control the scheduling of every message of ρ, including delivering messages out of order or withholding them forever. We show that $\mathcal{A}$ can be reduced to a super-rushing adversary $\mathcal{A}_{\mathsf{SR}}$ for a single *standalone* execution of ρ, which emulates $\mathcal{A}$ as is, with no modifications. Since $\mathcal{A}$ may not deliver all the messages, the honest parties may be in an intermediate stage of the execution of ρ when $\mathcal{A}$ halts. $\mathcal{A}_{\mathsf{SR}}$ just completes the execution in a fixed manner in its standalone execution of ρ, and the honest parties that are stalled in the concurrent execution simply keep waiting for their output. Crucially, $\mathcal{A}_{\mathsf{SR}}$ is super-rushing, and makes only black-box use of $\mathcal{A}$.

Now, the super-rushing security of ρ with a black-box straight-line simulator, say $\mathcal{S}_{\mathsf{SR}}$, allows the black-box composition of the above reduction with $\mathcal{S}_{\mathsf{SR}}$. This yields a black-box straight-line simulator for the dummy composition, i.e., security of π_0^ρ in the presence of $\mathcal{A}$ with selective abort. We show the same via a standard hybrid argument. We refer the reader to Lemma 4.1 for the formal details.

From selective to unanimous abort: end synchronisation. Selective abort is the best one can hope for from a concurrent execution over point-to-point channels: an honest party has no way of knowing whether another honest party has completed the protocol, and different honest parties may unavoidably reach different conclusions about whether to abort. We obtain *unanimous* abort by additionally assuming an atomic broadcast channel and appending an *end synchronisation* round to ρ, giving an $(R+1)$-round protocol ρ_+: every party broadcasts $(\textsc{goodbye}, \rho)$ upon completing the R rounds of ρ, and outputs only upon receiving $(\textsc{goodbye}, \rho)$ from *all* n parties. Atomicity is what makes unanimity possible at all: a broadcast that has been delivered to some honest party is eventually delivered, identically, to every honest party. Hence, one of two outcomes occurs:

- Either the broadcast of every party is delivered to some honest party, in which case every honest party eventually receives all n broadcasts and outputs; or
- The broadcast of some party is never delivered, in which case no honest party ever receives all n broadcasts, and no honest party outputs.

We extend the reduction $\mathcal{A}_{\mathsf{SR}}$ to additionally track the deliveries of the round-$(R+1)$ broadcasts, and use an almost identical hybrid argument to show security under concurrent execution with unanimous abort. We refer the reader to Lemma 4.3 for the formal details.

From dummy to general composition: HELLO's and GOODBYE's. Both the results above only consider the dummy composition π_0, where nothing happens beside the single call to ρ. Under *general* composition, an arbitrary calling protocol π may run arbitrary other computations concurrently with ρ. As a consequence, the adversary may:

- Influence the inputs of some honest parties to ρ (via π) as a function of the messages of ρ that other honest parties, which have already begun executing ρ, have sent. In the hybrid world, no information about the execution of f is available before all the inputs are fixed.
- Observe the messages of π that honest parties send after completing ρ, which may depend on their outputs of ρ, while ρ has not yet completed for the other honest parties, and choose its remaining messages of ρ as a function of them. In the hybrid world, f releases outputs only after the inputs of the corrupted parties are fixed.

The former is mitigated by *start synchronisation* [KLR10]: before sending a single message of ρ, every party sends (HELLO, ρ) and waits to receive (HELLO, ρ) from all n parties. The latter is mitigated by *end synchronisation*: every party sends $(\text{GOODBYE}, \rho)$ upon completing ρ, and outputs only upon receiving such a message from all n parties.

The instants T_0 and T. In the proof, we identify two instants in the execution of $\pi^{\widehat{\rho}}$:

- The first instant T_0 at which an honest party completes the start synchronisation, at which point the inputs of all the honest parties to ρ are fixed; and
- The first instant T at which an honest party completes the end synchronisation, at which point every honest party has completed ρ, and all the messages of ρ between the honest and the corrupted parties are determined.

Between T_0 and T, the honest parties' messages corresponding to the invocation of ρ by π are effectively a standalone execution of ρ: their inputs have been fixed, and no honest party has obtained its output. Moreover, the adversary requests a message of ρ from an honest party only once the party could compute it, and the reduction emulates π internally, so this execution between T_0 and T corresponds to a partial execution of ρ against a super-rushing adversary. As in the case of the dummy composition, we again complete this execution in a fixed manner if the adversary halts in between T_0 and T. Moreover, the inputs to this standalone execution are fixed by the execution of π until T_0, and after T, the execution of $\pi^{\widehat{\rho}}$ is a deterministic function of the outputs of the standalone execution. The reduction emulates the execution of π until T_0 and after T, and the simulator does the same in the hybrid world. With broadcast end synchronisation, the same argument yields unanimous abort, where T is redefined as the first instant at which the broadcast of every party has been delivered to some honest party. We refer the reader to Theorem 4.2 and Theorem 4.4 for the formal details.

2.2 Composing Protocols with Setup

For protocols ρ in the $\mathcal{G}$-hybrid model, we follow the same two steps. In a composed execution, every execution of ρ uses its own copy of $\mathcal{G}$, which the honest parties interact with only as part of ρ, whereas the adversary may interact with it at any point in time, and in particular, before the start synchronisation completes. Moreover, the environment may interact with $\mathcal{G}$ both before choosing the inputs and after the execution. This raises two difficulties, which correspond to the two conditions of Theorem 1.2: adaptive inputs and stateless setup.

Inputs that depend on the setup. Even though start synchronisation fixes the inputs of the honest parties to ρ before any message of ρ is sent, these inputs may depend on the setup: the adversary may obtain some value of the setup prior to T_0, and influence the inputs of the honest parties as a function of that value, via π. Therefore, the reduction to a standalone execution of ρ must account for inputs that are chosen after interacting with the setup. To that end, the environment for the standalone execution of ρ runs the execution of $\pi^{\widehat{\rho}}$ prior to T_0 as part of its input selection, forwarding the interactions of the adversary with $\mathcal{G}$ to its own interface, and hands the record of this execution to the reduction as its auxiliary input. The reduction then resumes the execution from T_0, exactly as in the plain model. Hence, we require that ρ is secure with *adaptive inputs*, i.e., even when the environment chooses the inputs after interacting with the setup. This is required already for the dummy composition, since the environment may itself interact with the setup prior to choosing the inputs.

Simulating the setup. The simulator for ρ plays the role of $\mathcal{G}$ in two ways: towards the environment, before and after the execution, and towards its oracle, during the execution. Our simulator for π uses the former to respond to the interactions of the adversary with $\mathcal{G}$ prior to T_0 and after T, and the latter between T_0 and T. Since our simulator can rewind neither the environment nor the adversary, we require that the simulator for ρ responds to the environment in a straight-line manner, in addition to being black-box and straight-line in its oracle.

Completing the execution. In the plain model, the completion of the standalone execution after the adversary halts is invisible: no entity observes the execution afterwards. With a setup, however, the honest parties that are stalled in the concurrent execution do interact with $\mathcal{G}$ in the completion, whereas the environment, and in the composed execution, the adversary, may continue to interact with $\mathcal{G}$ after the execution. Hence, we require the setup to be *stateless*: the interactions of the honest parties with $\mathcal{G}$ do not affect its responses to the adversary and the corrupted parties. We refer the reader to Lemma 5.1, Theorem 5.2 and Theorem 5.5 for the formal details.

On multiple calls. Theorem 4.2 and Theorem 4.4, as well as Theorem 5.2 and Theorem 5.5, consider a protocol π that makes a single ideal call to f. For protocols that make multiple ideal calls, we sketch the extension via a sequence of hybrids following [Lin03b], where in each hybrid a single execution of $\widehat{\rho}$ is replaced by an ideal call to f, and the above argument is applied to the replaced execution, treating the remaining executions as part of the calling protocol. With setup, every execution of $\widehat{\rho}$ uses its own localised copy of $\mathcal{G}$, and the remaining executions are treated as part of the calling protocol together with their own copies of $\mathcal{G}$. Since the copies are localised, the random coins of the copy of the replaced execution are independent of those of every other copy, and only the replaced execution and the adversary interact with it. Hence, the above argument applies to the replaced execution as is. Note that while replacing each invocation to $\widehat{\rho}$, the hybrid model execution of π contains other independent localised setup instances, i.e., π is defined in the $(f, \mathcal{G})$-hybrid.

A separation: adaptive inputs. We show that security with adaptive inputs cannot be replaced by the usual notion of security in the $\mathcal{G}$-hybrid model, where the inputs are fixed independently of the setup. Consider a setup $\mathcal{G}$ that returns a fixed value u of high min-entropy, such as a common reference string, and the following two-party protocol ρ_{na}:

1. P_1 holds no input and P_2 holds an input (v, s).
2. P_2 reveals the secret s to P_1, if and only if $v = u$ (as obtained from the setup).
3. P_1 outputs nothing and P_2 outputs its secret s.

We consider the simple two party functionality $f(\lambda, (v, s)) = (\lambda, s)$ and note that ρ_{na} securely computes f in the presence of a single corruption: the inputs are independent of the setup, $v = u$ occurs with negligible probability, and w.h.p the secret s is not revealed to a (potentially) corrupted party P_1. Now, consider the following calling protocol π:

1. P_1 holds no input and P_2 holds an input s.
2. P_1 sends a value v to P_2.
3. P_2 invokes f with (v, s) and the parties output their output from f.

In the above composed execution, an adversary corrupting P_1 can receive u from the setup before the start synchronisation, and sends $v \stackrel{def}{=} u$ to P_2, effectively, choosing the input v of P_2. Then, P_2 reveals s in $\widehat{\rho}_{\mathsf{na}}$, whereas in the hybrid world, s is independent of the view of the adversary. Hence, ρ_{na} is secure in the standalone setting with non-adaptive inputs, but is not composable! Therefore, security with adaptive inputs is necessary for composing protocols with setup. We refer the reader to Theorem 5.7 for the formal details.

2.3 Natural Protocols with Setup and Adaptive Inputs

Theorem 1.2 requires the two conditions of adaptive inputs and stateless setup. We now argue that both conditions are satisfied by natural protocols and setups, and discuss where they fail. We stress that the following concerns only these two conditions; the super-rushing security of the protocols below has to be established separately.

Achieving security with adaptive inputs. The protocol ρ_{na} of our separation is secure under the usual notion only because its security argument relies on the inputs being statistically independent of the setup. Typical proofs of security in a setup model do not argue in this manner. Rather, they proceed in two steps:

- **Simulating the setup.** The real setup is replaced by a simulated one, e.g., a CRS with a trapdoor, or public keys for which the simulator knows the secret keys, which is indistinguishable even to a distinguisher that observes the setup.
- **Using the trapdoor.** The simulator uses the trapdoor to extract the inputs of the corrupted parties, or to equivocate the messages of the honest parties, in a straight-line manner.

Neither step uses the independence of the inputs from the setup, and an environment that chooses the inputs after interacting with the setup is subsumed by the distinguisher of the first step. Hence, such proofs typically establish security with adaptive inputs as is. Classical examples include non-interactive zero-knowledge proofs in the CRS model, with soundness against statements chosen after the CRS [FLS99]; simulation-sound non-interactive zero knowledge [Sah99]; commitments with straight-line extraction and equivocation, and the protocols built from them [CF01, CLOS02]; and authenticated broadcast (the Byzantine generals problem) [DS83] with a public-key infrastructure, tolerating any number of corruptions, where the simulator generates the keys of the honest parties, and inputs that depend on the public keys are handled by the unforgeability of the signature scheme under chosen-message attacks.

Programmable random oracles. For protocols in the random oracle model [BR93], the picture is less clear. If the simulator programs the random oracle only at fresh, high-entropy points, e.g., points that include randomness sampled by the simulator, then the queries of the environment prior to choosing the inputs hit these points only with negligible probability, and the above argument applies. However, if the simulator must program the random oracle at points that are determined by the inputs of the honest parties, then the environment, which knows these inputs, may have queried these points prior to choosing the inputs, and the simulator can no longer program them. Whether natural protocols in the random oracle model are secure with adaptive inputs against super-rushing adversaries is left open.

Stateless setups. A stateless setup may still maintain an internal state, e.g., the CRS, the table of a random oracle, or the keys of a public-key infrastructure. What matters is that this state, and hence its responses to the adversary, is sampled independently of whether, when, and how the honest parties interact with it. This holds for a common reference string [CF01], a (non-observable) random oracle [BR93], and a public-key infrastructure in which the keys are generated by the setup itself, since the honest parties only read from such setups.

Stateful setups. Statelessness excludes setups whose interface to the adversary reflects the interactions of the honest parties. Examples include a public-key infrastructure in which the honest parties register their keys during the execution of ρ, and the adversary may retrieve the registered keys [BCNP04, CDPW07]; an observable random oracle, in which the adversary learns the queries of the honest parties [CJS14, CDG+18]; and setups that release a value to the adversary once some set of honest parties has requested it, or that otherwise maintain a counter of the requests of the honest parties. For such setups, the completion of the execution by our reduction may be detected by the environment, even perfectly, and our proofs do not apply. We leave open whether security under concurrent general composition can be obtained for such stateful setups, e.g., via a reduction that does not need to complete the execution of ρ.

3 Definitions

Notations Let $\lambda \in \mathbb{N}$ denote the security parameter. For $n \in \mathbb{N}$, let $[n] \stackrel{def}{=} \{1, \ldots, n\}$. Let poly denote the set of all positive polynomials and let PPT denote a probabilistic interactive Turing machine that runs in polynomial time in λ. A function $\mu : \mathbb{N} \to [0,1]$ is *negligible*, denoted as $\mu(\lambda) = \mathsf{negl}(\lambda)$, if $\mu(\lambda) < \frac{1}{p(\lambda)}$ for every $p \in \mathsf{poly}$ and sufficiently large λ. Two distribution ensembles $\{X(\lambda, a)\}_{\lambda \in \mathbb{N}, a \in \{0,1\}^*}$ and $\{Y(\lambda, a)\}_{\lambda \in \mathbb{N}, a \in \{0,1\}^*}$ are *computationally indistinguishable*, if for every non-uniform polynomial-time distinguisher D, there exists a negligible function μ such that for every $\lambda \in \mathbb{N}$ and $a \in \{0,1\}^*$, it holds that

$$|\mathsf{Pr}\left[D(X(\lambda, a)) = 1\right] - \mathsf{Pr}\left[D(Y(\lambda, a)) = 1\right]| \leq \mu(\lambda) \ .$$

Two distribution ensembles are *statistically close*, if the above holds for every non-uniform (not necessarily polynomial-time) distinguisher D. Computational and statistical indistinguishability are denoted by $\stackrel{c}{\equiv}$ and $\stackrel{s}{\equiv}$, respectively.

3.1 Synchronous Communication

The messages in an execution of a *synchronous protocol* ρ may be ordered by a special index, called a *round* such that in each round, all the parties are activated simultaneously. Each round consists of a *send phase*, denoting the sending of messages for the round, and a corresponding, *receive phase*, denoting the receipt of the messages sent by all the parties in the send phase of the round. All parties begin the protocol at round 1, and for every round $r \in \mathbb{N}$, all parties advance from round r to round $r+1$ only upon all non-faulty parties having completed their prescribed operations for the round r. When a party P_i sends a message m to party P_j in round r, we denote the message together with the metadata by (i, j, r, m). Note that this description implicitly assumes that the point-to-point channels are authenticated. In addition, we assume an atomic broadcast channel that allows each designated party to unequivocally send a message to all parties. For the case of information-theoretic security, we additionally assume that the channels are ideally private, that is, the adversary only obtains the metadata (j, k, r) for every message between a pair of honest parties j, k.

In every round, every party can send a message to any other party as a function of its input, its randomness, its history of incoming messages, and the identity of the recipient. We also assume that every corrupted party always sends a message when supposed to (this is a standard assumption for synchronous protocols, as the recipient can identify the event where a party is cheating by not sending a message when it is supposed to, and can instead pretend to have received a default message instead).

We note that in an execution of a synchronous protocol each honest party knows how many messages it should receive in a given round and from whom. As such, every honest party can generate its messages for round $r+1$ immediately after receiving all of its messages for round r. The synchrony assumption, however, ensures that those messages are delivered to their destinations only after all round-r messages have been delivered to all honest parties.

3.2 Super-Rushing Adversaries

We consider super-rushing adversaries, as introduced in [ACCI25], that can reorder messages within a given round (similar to standard rushing adversaries), and, in addition, can also reorder messages *between different rounds*. As in [ACCI25], we consider an interface that allows the adversary to perform such a re-ordering under certain restrictions.

The interface of the adversary. For concreteness, and to simplify notations in our proofs, we will use the following interface between the adversary and the honest parties:

- The adversary can send a message from a corrupted party to an honest party.
- The adversary can request a message from an honest party to a corrupted party.

We assume that the adversary is well behaved in the following sense. First, the adversary always sends a message from a corrupted party to an honest party when supposed to, and second, if the adversary requests

a message corresponding to (j^*, i^*, r^*) for an honest P_{j^*} and a corrupted P_{i^*}, then the honest party P_{j^*} must have received all its prescribed messages as per the protocol until round $r^* - 1$. In effect, the super-rushing adversary may request an honest party's future message only when the honest party could compute it using its next message function.

Remark 3.1 (Scheduling of honest-to-honest messages)**.** *In the interface of [ACCI25], messages between honest parties are delivered without the involvement of the adversary. In contrast, we let the adversary also schedule the delivery of messages between honest parties (while it eventually delivers all of them). This does not strengthen the adversary: delivering a message between honest parties earlier does not change the messages that the honest parties send, and only enlarges the set of messages that the adversary is allowed to request. Hence, any adversary in our interface can be emulated by an adversary in the interface of [ACCI25] with an identical view, and in particular, the sufficient conditions of [ACCI25] for super-rushing security apply to our interface.*

3.3 Standalone Security

We consider security in the standalone model as defined by [Can00]. Let $f : (\{0,1\}^*)^n \mapsto (\{0,1\}^*)^n$ be an n-party functionality and let ρ be a protocol over the parties $[n]$ where the parties are connected pair-wise by authenticated channels and the parties have access to an *atomic* broadcast channel. The adversary $\mathcal{A}$ has auxiliary input z, and let $I \subset [n]$.

The real execution. In this world, the parties run the protocol ρ where the adversary $\mathcal{A}$ maliciously corrupts the parties in I. The adversary is super-rushing and adheres to the restrictions of sending all corrupted parties' messages when it is supposed to and only requesting future messages that can be computed by the corresponding honest parties' as per the protocol ρ.

Definition 3.2 (Real-World Execution)**.** *Let $\rho = (P_1, \ldots, P_n)$ be an n-party protocol and let $I \subset [n]$ denote the set of indices of the parties corrupted by a real-world super-rushing adversary $\mathcal{A}$. Denote by $\mathsf{REAL}_{\rho,I,\mathcal{A}}(\lambda, \boldsymbol{x}, z)$ the random variable describing the output vector of $P_1, \ldots, P_n$ and $\mathcal{A}$ resulting from the protocol interaction on security parameter $\lambda \in \mathbb{N}$, input vector $\boldsymbol{x} = (x_1, \ldots, x_n)$, and auxiliary input z.*

Let $\mathsf{REAL}_{\rho,I,\mathcal{A}}$ denote the distribution ensemble $\{\mathsf{REAL}_{\rho,I,\mathcal{A}}(\lambda, \boldsymbol{x}, z)\}_{\lambda\in\mathbb{N},(\boldsymbol{x},z)\in(\{0,1\}^*)^{n+1}}$.

The ideal execution. The ideal model consists of honest parties, a trusted party or ideal functionality, and an ideal adversary $\mathcal{S}$ controlling the same subset I of parties. The honest parties send their inputs to the ideal functionality. $\mathcal{S}$ receives the auxiliary input z and the inputs of the corrupted parties. $\mathcal{S}$ may substitute the inputs of the corrupted parties (as long as the length of the inputs remains the same). The ideal functionality then receives all the inputs $x_1, \ldots, x_n$ where $\mathcal{S}$ may have replaced some inputs and computes the output $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$ and sends y_j to P_j for the honest parties and hands the outputs of the corrupted parties to the adversary.

Definition 3.3 (Ideal Computation)**.** *Let $f : (\{0,1\}^*)^n \to (\{0,1\}^*)^n$ be an n-party functionality and let $I \subset [n]$ denote the set of indices of the parties corrupted by the ideal-world adversary $\mathcal{S}$. Denote by $\mathsf{IDEAL}_{f,I,\mathcal{S}}(\lambda, \boldsymbol{x}, z)$ the random variable describing the output of $\mathcal{S}$ resulting from the above described ideal process on security parameter $\lambda \in \mathbb{N}$, input vector $\boldsymbol{x} = (x_1, \ldots, x_n)$, and auxiliary input z.*

Let $\mathsf{IDEAL}_{f,I,\mathcal{S}}$ denote the distribution ensemble $\{\mathsf{IDEAL}_{f,I,\mathcal{S}}(\lambda, \boldsymbol{x}, z)\}_{\lambda\in\mathbb{N},\,(\boldsymbol{x},z)\in(\{0,1\}^*)^{n+1}}$.

Security. Security is given by emulation of every real-world adversary $\mathcal{A}$ by the ideal-world adversary $\mathcal{S}$.

Definition 3.4 (Security)**.** *Let $f : (\{0,1\}^*)^n \to (\{0,1\}^*)^n$ be an n-party functionality. A protocol ρ t-securely computes f with computational security if for every PPT real-world adversary $\mathcal{A}$ and every $I \subset [n]$ with $|I| \le t$, there exists a PPT ideal-world adversary $\mathcal{S}$, such that*

$$\mathsf{REAL}_{\rho,I,\mathcal{A}} \stackrel{c}{\equiv} \mathsf{IDEAL}_{f,I,\mathcal{S}} .$$

A protocol ρ t-securely computes f with statistical security if the same holds for every (not necessarily PPT) real-world adversary $\mathcal{A}$, with an ideal-world adversary $\mathcal{S}$ that need not run in polynomial time, and with the two ensembles above being statistically close, i.e., $\stackrel{s}{\equiv}$ instead.

For the setting of perfect security, we omit the security parameter λ and instead include random coins $r_i \in \{0,1\}^*$ for each party $i \in [n]$ in the real-world execution and the random coins r_f and $r_{\mathcal{S}}$ in the ideal world execution. Note that perfect security requires that the real and ideal executions are identically distributed.

Remark 3.5 (Black-Box Straight-Line Simulation)**.** *We refer the reader to [Gol01] (Section 4.5) for the standard treatment of black-box simulation. In short, a black-box simulator $\mathcal{S}^{\mathcal{A}}$ is a single ideal-world adversary that works for every real-world adversary $\mathcal{A}$, given oracle access to $\mathcal{A}$ rather than to its code. It is straight-line if it never rewinds $\mathcal{A}$, i.e., $\mathcal{A}$ maintains state across oracle calls exactly as it would across rounds of a real protocol execution; equivalently, $\mathcal{S}^{\mathcal{A}}$ interacts with $\mathcal{A}$ as real parties would in a real execution, except that it may play the role of several parties at once. As usual, the oracle of $\mathcal{S}^{\mathcal{A}}$ is $\mathcal{A}$ with its auxiliary input z and its random coins fixed, and the output of $\mathcal{S}^{\mathcal{A}}$ is the output of its oracle, which $\mathcal{S}^{\mathcal{A}}$ does not otherwise use. We denote the resulting ideal-world execution by* $\mathsf{IDEAL}_{f,I,\mathcal{S}^{\mathcal{A}}}$.

3.4 Security Under General Composition

We consider security under *concurrent general composition*, as defined by [Lin03b]: a protocol ρ is secure if it remains secure when run concurrently with an arbitrary other protocol π that determines its inputs and makes use of its outputs, rather than proving a stand-alone notion of security and separately establishing a composition theorem.

Adversarial behavior. As before, we consider malicious adversaries $\mathcal{A}$ that corrupt a fixed subset $I \subset [n]$ of at most t parties, statically, before the protocol execution begins. Here, however, the adversary also has full control over the scheduling of the delivery of all messages, so that the network is asynchronous; it is precisely this control over scheduling that gives rise to concurrency. This adversary shares the super-rushing well-behavedness restriction: if it requests a message corresponding to (j^*, k^*, r^*) for an honest P_{j^*}, then P_{j^*} must have received all its prescribed messages as per the protocol until round $r^* - 1$. Its interface is otherwise strictly more permissive than that of a super-rushing adversary: a super-rushing adversary must eventually request the delivery of every message, so that the protocol always runs to completion, whereas this adversary may withhold any message of ρ permanently, leaving the corresponding party stalled indefinitely. It is precisely this ability to block messages forever, absent from the super-rushing interface, that forces us to settle for security with (selective) abort rather than guaranteed output delivery in what follows. The atomic broadcast channel, however, guarantees consistency: if a message sent over the broadcast channel is delivered to some honest party, then it is eventually delivered, identically, to every honest party, even if the adversary halts in the meantime, although the adversary controls the time at which it is delivered to each honest party. Equivalently, if some honest party never receives a broadcast message, then no honest party does. In particular, a broadcast message that has been delivered to no honest party by the time the adversary halts is never delivered.

On the atomic broadcast channel. We treat the atomic broadcast channel as an ideal resource, and the guarantees of our compilers with unanimous abort are only as strong as this resource. Its consistency property, namely that a message delivered to some honest party is eventually delivered, identically, to every honest party, even if the adversary halts in the meantime, is a strong assumption in a network in which the adversary schedules the delivery of every message. In particular, it is not implied by the point-to-point channels in this setting: realising such a channel over an adversarially scheduled point-to-point network is a question of its own, and protocols for this task typically tolerate only a bounded fraction of corruptions, and in the setting of a dishonest majority, the channel should be viewed as an additional trust assumption. Under composition, we assume a single broadcast channel that is shared by all the executions, where every broadcast message is tagged with the identifier of the instance (e.g., $(\text{GOODBYE}, \rho)$ for the instance ρ), so that the consistency of the channel holds for the messages of every instance separately. Our compilers with selective abort send the start and end synchronisation over the point-to-point channels.

The hybrid model. Let π be an arbitrary n-party protocol that, in addition to standard point-to-point and broadcast messages, may send *ideal messages* to and from a trusted party computing the functionality

f (recall that π models arbitrary network activity, and may in particular make subroutine calls to f). The resulting execution is a hybrid of the real and ideal executions of Definition 3.2 and Definition 3.3: the messages of π are sent directly between the parties, while the trusted party is used only to respond to the ideal calls to f. We refer the reader to [Lin03b] for the explicit description of the hybrid execution. In short, a hybrid-model adversary $\mathcal{S}$ controls the delivery of all messages, standard and ideal alike, and corrupts the same fixed subset I of parties as in the real-world execution; the channels to the trusted party, however, remain authenticated and private, so $\mathcal{S}$ may deliver, but not read, the ideal messages.

Definition 3.6 (Hybrid Execution)**.** *Let $\pi = (P_1, \ldots, P_n)$ be an n-party protocol in the f-hybrid model, and let $I \subset [n]$ denote the set of indices of the parties corrupted by a hybrid-model adversary $\mathcal{S}$. Denote by $\mathsf{HYBRID}^{f}_{\pi,I,\mathcal{S}}(\lambda, \boldsymbol{x}, z)$ the random variable describing the output vector of $P_1, \ldots, P_n$ and $\mathcal{S}$ resulting from the above hybrid execution, on security parameter $\lambda \in \mathbb{N}$, input vector $\boldsymbol{x} = (x_1, \ldots, x_n)$, and auxiliary input z.*

Let $\mathsf{HYBRID}^{f}_{\pi,I,\mathcal{S}}$ denote the distribution ensemble $\{\mathsf{HYBRID}^{f}_{\pi,I,\mathcal{S}}(\lambda, \boldsymbol{x}, z)\}_{\lambda\in\mathbb{N}, (\boldsymbol{x},z)\in(\{0,1\}^*)^{n+1}}$.

The real model: general composition. Let $\rho = (P_1, \ldots, P_n)$ be an n-party protocol computing f. The *composition* of π with ρ, denoted π^ρ, replaces every ideal call to f by an execution of ρ: each party runs a separate copy of ρ's next message algorithm, feeding it the ideal message as input and forwarding ρ's eventual output back into π in place of the trusted party's reply. The composed protocol π^ρ is then executed exactly as in Definition 3.2, with the real-world adversary $\mathcal{A}$ corrupting the fixed subset I and additionally controlling message scheduling. We denote the resulting real-model execution by $\mathsf{REAL}_{\pi^\rho,I,\mathcal{A}}$, exactly as in Definition 3.2.

Security definition. We are now ready to define security under concurrent general composition.

Definition 3.7 (Security Under Concurrent General Composition)**.** *Let ρ be an n-party protocol and f an n-party functionality. The protocol ρ t-securely computes f under concurrent general composition if for every PPT n-party protocol π in the f-hybrid model, every PPT real-model adversary $\mathcal{A}$ for π^ρ, and every $I \subset [n]$ with $|I| \leq t$, there exists a PPT hybrid-model adversary $\mathcal{S}$ such that*

$$\mathsf{HYBRID}^{f}_{\pi,I,\mathcal{S}} \stackrel{c}{\equiv} \mathsf{REAL}_{\pi^\rho,I,\mathcal{A}} \ .$$

The protocol ρ t-securely computes f under concurrent general composition with statistical security if the same holds for every (not necessarily PPT) real-model adversary $\mathcal{A}$, with a hybrid-model adversary $\mathcal{S}$ that need not run in polynomial time, and with the two ensembles above being statistically close, i.e., $\stackrel{s}{\equiv}$ instead. Finally, the protocol ρ t-securely computes f under concurrent general composition with perfect security if the same holds for every (not necessarily PPT) real-model adversary $\mathcal{A}$, with a hybrid-model adversary $\mathcal{S}$ that need not run in polynomial time, and with the two ensembles above being identically distributed, i.e., $\equiv$ instead, where, as in the standalone setting, the security parameter is omitted and the random coins of the parties, of the trusted party and of $\mathcal{S}$ are included instead.

Remark 3.8 (Bounded Concurrent General Composition)**.** *If π is restricted to protocols that make at most ℓ ideal calls to f, then ρ is said to securely compute f under ℓ-bounded concurrent general composition.*

3.5 Type of Abort

The ideal and hybrid executions of Definition 3.3 and Definition 3.6 implicitly guarantee that every party receives its output, i.e., *guaranteed output delivery*. Following [CGZ20], we instead consider two weaker guarantees, distinguished by the power given to $\mathcal{S}$ to make the trusted party withhold output from a subset of the honest parties: *selective abort* (SA) and *unanimous abort* (UA).

Ideal execution with abort. Let $I \subseteq [n]$ denote the fixed subset of parties corrupted by $\mathcal{S}$, as in the ideal computation of Definition 3.3. Once the trusted party computes the outputs of f, denoted as $(y_1, \ldots, y_n)$, it sends $\{y_i\}_{i\in I}$ to $\mathcal{S}$ (if $I \neq \emptyset$), who may then respond with a message $(\texttt{abort}, J)$. Depending on $\mathsf{type} \in \{\mathsf{SA}, \mathsf{UA}\}$, the following occurs:

SA. $\mathcal{S}$ selects an arbitrary $J \subseteq [n] \setminus I$.

UA. $\mathcal{S}$ selects either $J = [n] \setminus I$ or $J = \emptyset$.

Every party P_j with $j \in J$ receives $\perp$ in place of y_j; every other party receives y_j as usual. We denote the ideal execution with $\mathsf{type} \in \{\mathsf{SA}, \mathsf{UA}\}$, as $\mathsf{IDEAL}^{\mathsf{type}}_{f,I,\mathcal{S}}$, exactly as in Definition 3.3.

Standalone security with abort. We say that a protocol ρ t-securely computes f (against super-rushing adversaries) with selective (resp. unanimous) abort, if Definition 3.4 holds with $\mathsf{IDEAL}^{\mathsf{SA}}_{f,I,\mathcal{S}}$ (resp. $\mathsf{IDEAL}^{\mathsf{UA}}_{f,I,\mathcal{S}}$) in place of $\mathsf{IDEAL}_{f,I,\mathcal{S}}$. Security with guaranteed output delivery implies security with unanimous abort (the simulator always responds with $(\mathtt{abort}, \emptyset)$), which in turn implies security with selective abort. Without loss of generality, the simulator sends its message $(\mathtt{abort}, J)$ before it halts.

Aborting versus waiting. In a synchronous execution, every party knows when each of its messages is due, and hence the absence of a message is detectable (see the convention on default messages above). In particular, a party that does not obtain its output by the end of the protocol outputs $\perp$, and there is no difference between not receiving an output and receiving $\perp$. Under concurrent composition, however, the network is asynchronous: a party cannot distinguish a message that is delayed from a message that is never delivered, and so it simply waits. Hence, not receiving an output is *not* the same as receiving $\perp$. The output $\perp$ is an event upon which the calling protocol π may act, e.g., by notifying the other parties or by proceeding differently, whereas a party that waits for its output takes no further action in that call. Accordingly, in the hybrid execution below, the parties in J receive $\perp$, whereas a party whose output is withheld by the hybrid-model adversary, which controls the delivery of the ideal messages, keeps waiting. The execution ends when the adversary halts, and the output of every honest party is whatever it has output by then.

Hybrid execution with abort. The hybrid execution of Definition 3.6 is defined identically, except that the trusted party now responds to every ideal call π makes to f as per type. Once again, we denote the hybrid execution with $\mathsf{type} \in \{\mathsf{SA}, \mathsf{UA}\}$, as $\mathsf{HYBRID}^{f,\mathsf{type}}_{\pi,I,\mathcal{S}}$, exactly as in Definition 3.6.

3.6 Security with Setup

We also consider protocols that make use of a trusted *setup*, such as a common reference string, a random oracle, or a public-key infrastructure, modelled as an ideal functionality $\mathcal{G}$. A protocol ρ in the $\mathcal{G}$-hybrid model may, in addition to standard point-to-point and broadcast messages, interact with $\mathcal{G}$. We treat the communication between a party and $\mathcal{G}$ as messages of ρ, whose delivery is scheduled by the adversary, as for every other message. The adversary may interact with $\mathcal{G}$, on behalf of the corrupted parties and via the interface that $\mathcal{G}$ provides to the adversary, at any point in time. In the ideal execution, the simulator plays the role of $\mathcal{G}$ towards the adversary.

Adaptive inputs. In the standalone setting of Definition 3.4, the inputs $\boldsymbol{x}$ and the auxiliary input z are fixed independently of the setup. We consider a stronger notion of security, where the inputs of the honest parties may be chosen adaptively after the setup has been generated. To that end, we consider an additional adversarial entity, a non-uniform PPT machine $\mathcal{Z}$, called the *environment*. $\mathcal{Z}$ chooses the inputs after interacting with the setup, and may continue to interact with the setup after the execution. The output of the environment is a single bit.

The real execution. In this world, the parties run the protocol ρ where the adversary $\mathcal{A}$ maliciously corrupts the parties in I. The adversary may be super-rushing or concurrent (may block messages), and there is an additional entity, the environment $\mathcal{Z}$. The execution proceeds as follows:

1. The environment $\mathcal{Z}$ interacts with $\mathcal{G}$ via the interface of the adversary and of the corrupted parties, and outputs the parties' inputs $\boldsymbol{x}$ and an auxiliary input z.

2. The parties run the protocol ρ on inputs $\boldsymbol{x}$, in the $\mathcal{G}$-hybrid model with the same instance of $\mathcal{G}$, where the adversary $\mathcal{A}$ with auxiliary input z maliciously corrupts the parties in I.

3. Finally, $\mathcal{Z}$ receives the output vector of the execution, i.e., the outputs of the honest parties and the output of $\mathcal{A}$, may further interact with $\mathcal{G}$ via the same interface, and outputs a bit.

Definition 3.9 (Real-World Execution with Setup)**.** *Let $\rho = (P_1, \ldots, P_n)$ be an n-party protocol in the $\mathcal{G}$-hybrid model, let $I \subset [n]$ denote the set of indices of the parties corrupted by a real-world adversary $\mathcal{A}$, and let $\mathcal{Z}$ be an environment. Denote by $\mathsf{REAL}^{\mathcal{G}}_{\rho,I,\mathcal{A},\mathcal{Z}}(\lambda)$ the random variable describing the output of $\mathcal{Z}$ resulting from the above real execution on security parameter $\lambda \in \mathbb{N}$.*

The ideal execution. The ideal execution is identical, except that the ideal-world adversary $\mathcal{S}$ responds to the interactions of $\mathcal{Z}$ with $\mathcal{G}$, both before and after the execution, and the execution of ρ is replaced by the ideal computation of f on inputs $\boldsymbol{x}$ (chosen by $\mathcal{Z}$), or its variant with $\mathsf{type} \in \{\mathsf{SA}, \mathsf{UA}\}$ (Section 3.5), where $\mathcal{S}$ receives the auxiliary input z.

Definition 3.10 (Ideal Computation with Setup)**.** *Let $f : (\{0,1\}^*)^n \to (\{0,1\}^*)^n$ be an n-party functionality, let $I \subset [n]$ denote the set of indices of the parties corrupted by the ideal-world adversary $\mathcal{S}$, and let $\mathcal{Z}$ be an environment. Denote by $\mathsf{IDEAL}_{f,I,\mathcal{S},\mathcal{Z}}(\lambda)$ (resp. $\mathsf{IDEAL}^{\mathsf{type}}_{f,I,\mathcal{S},\mathcal{Z}}(\lambda)$ for $\mathsf{type} \in \{\mathsf{SA}, \mathsf{UA}\}$) the random variable describing the output of $\mathcal{Z}$ resulting from the above ideal execution on security parameter $\lambda \in \mathbb{N}$.*

Security. Security is defined, as before, as the emulation of every real-world adversary $\mathcal{A}$ by the ideal-world adversary $\mathcal{S}$, now for every environment $\mathcal{Z}$.

Definition 3.11 (Security with Adaptive Inputs)**.** *Let $f : (\{0,1\}^*)^n \to (\{0,1\}^*)^n$ be an n-party functionality and let ρ be a protocol in the $\mathcal{G}$-hybrid model. The protocol ρ t-securely computes f against super-rushing (resp. concurrent) adversaries with adaptive inputs and computational security, if for every PPT real-world super-rushing (resp. concurrent) adversary $\mathcal{A}$ and every $I \subset [n]$ with $|I| \le t$, there exists a PPT ideal-world adversary $\mathcal{S}$, such that for every environment $\mathcal{Z}$, it holds that*

$$\left|\mathsf{Pr}\left[\mathsf{REAL}^{\mathcal{G}}_{\rho,I,\mathcal{A},\mathcal{Z}}(\lambda) = 1\right] - \mathsf{Pr}\left[\mathsf{IDEAL}_{f,I,\mathcal{S},\mathcal{Z}}(\lambda) = 1\right]\right| \le \mathsf{negl}(\lambda) \ .$$

The protocol ρ t-securely computes f against super-rushing (resp. concurrent) adversaries with adaptive inputs and statistical security if the same holds for every (not necessarily PPT) real-world super-rushing (resp. concurrent) adversary $\mathcal{A}$ and environment $\mathcal{Z}$, with an ideal-world adversary $\mathcal{S}$ that need not run in polynomial time. Security with selective (resp. unanimous) abort is defined analogously, with $\mathsf{IDEAL}^{\mathsf{SA}}_{f,I,\mathcal{S},\mathcal{Z}}$ (resp. $\mathsf{IDEAL}^{\mathsf{UA}}_{f,I,\mathcal{S},\mathcal{Z}}$) in place of $\mathsf{IDEAL}_{f,I,\mathcal{S},\mathcal{Z}}$.

The usual notion of security in the $\mathcal{G}$-hybrid model is the special case where $\mathcal{Z}$ does not interact with $\mathcal{G}$ prior to choosing the inputs. In the plain model, i.e., without a setup, the two notions coincide: a randomised environment induces a distribution over the inputs $(\boldsymbol{x}, z)$, and its post-execution behaviour is a fixed function of the output vector. Finally, security under general composition in the $\mathcal{G}$-hybrid model is defined identically to Definition 3.11.

Remark 3.12 (Black-Box Straight-Line Simulation with Setup)**.** *We say that ρ is secure with adaptive inputs and black-box straight-line simulation, if Definition 3.11 holds with a single simulator $\mathcal{S}^{\mathcal{A}}$ that is black-box and straight-line in $\mathcal{A}$ (Remark 3.5), and moreover responds to the interactions of the environment with $\mathcal{G}$ in a straight-line manner, i.e., without rewinding $\mathcal{Z}$. As usual, the oracle of $\mathcal{S}^{\mathcal{A}}$ is $\mathcal{A}$ with its auxiliary input z and its random coins fixed; in particular, $\mathcal{S}^{\mathcal{A}}$ accesses z only through its oracle, and the output of $\mathcal{S}^{\mathcal{A}}$ is the output of its oracle, which $\mathcal{S}^{\mathcal{A}}$ does not otherwise use. After the execution of f ends, $\mathcal{S}^{\mathcal{A}}$ keeps responding to the interactions of the environment with $\mathcal{G}$.*

Setup under composition. In a composed protocol π^{ρ}, where ρ is a protocol in the $\mathcal{G}$-hybrid model, every instance sid of ρ has its own copy $\mathcal{G}_{\mathsf{sid}}$ of $\mathcal{G}$. We restrict our attention to setups that are local to each instance.

Definition 3.13 (Localised Setup)**.** *The copies $\{\mathcal{G}_{\mathsf{sid}}\}_{\mathsf{sid}}$ are said to be localised if the following hold for every instance sid of ρ:*

1. *The random coins of* $\mathcal{G}_{\mathsf{sid}}$ *are sampled independently of everything else in the execution.*
2. *Only the code of* ρ *in the instance* sid *and the adversary interact with* $\mathcal{G}_{\mathsf{sid}}$*. In particular, the calling protocol* π *and the other instances of* ρ *do not.*

We further restrict our attention to setups whose interface to the adversary is not affected by the interactions of the honest parties.

Interfaces. A copy $\mathcal{G}_{\mathsf{sid}}$ has random coins r_{G} and two interfaces: the *honest interface*, used by the honest parties, and the *adversarial interface*, used by the adversary on behalf of the corrupted parties and via the interface that $\mathcal{G}_{\mathsf{sid}}$ provides to the adversary. A *sequence of past queries* Q to $\mathcal{G}_{\mathsf{sid}}$ consists of the queries made via either interface, in order, each together with its response. For such Q and a (possibly unbounded and adaptive) party B that interacts with $\mathcal{G}_{\mathsf{sid}}$ via the adversarial interface, let $\mathsf{View}_{\mathsf{B}}(Q)$ denote the view of B, where r_{G} is conditioned on Q. For some additional queries Q' of the honest parties via the honest interface, let $\mathsf{View}_{\mathsf{B}}(Q, Q')$ denote the same, except that $\mathcal{G}_{\mathsf{sid}}$ first responds to Q', and only then B interacts with it.

Definition 3.14 (Stateless Setup)**.** *A copy* $\mathcal{G}_{\mathsf{sid}}$ *is said to be* ***stateless*** *(with respect to the honest parties) if for every sequence of past queries* Q*, all additional queries* Q' *of the honest parties, and every party* B*, it holds that*

$$\mathsf{View}_{\mathsf{B}}(Q) \equiv \mathsf{View}_{\mathsf{B}}(Q, Q') \ .$$

That is, additional interactions of the honest parties with a stateless setup do not change the distribution of its responses via the adversarial interface. Applying the definition repeatedly, these responses are distributed independently of whether, when, and how the honest parties interact with $\mathcal{G}_{\mathsf{sid}}$. For example, a common reference string, a (non-observable) random oracle, and a public-key infrastructure in which the setup generates the keys itself are stateless, whereas an observable random oracle, which reveals the queries of the honest parties to the adversary, is not.

If the copies are localised (Definition 3.13), then r_{G} is independent of everything else in the execution, and hence, conditioning on the sequence of past queries to $\mathcal{G}_{\mathsf{sid}}$ is equivalent to conditioning on a transcript of the execution. In the proofs of Section 5, Q' are the queries of the honest parties in the completion of an execution of ρ, and B is the environment interacting with the setup after the execution.

4 The Compiler via Start and End Synchronisation

Let ρ be an R-round protocol that t-securely computes an n-ary function f against super-rushing adversaries with straight-line black-box simulation, with perfect, statistical, or computational security. We first show that ρ is secure with selective abort against an adversary that, in addition to super-rushing, can also block messages, when ρ is executed in isolation (Section 4.1). We then show that adding start synchronisation and point-to-point end synchronisation to ρ, to obtain the $(R+2)$-round protocol $\widehat{\rho}$, yields security under concurrent general composition with selective abort (Section 4.3). Finally, we show that if the end synchronisation is over the broadcast channel, $\widehat{\rho}$ is secure under concurrent general composition with unanimous abort (Section 4.4).

4.1 Concurrent Execution with Selective Abort

A dummy composition. Let π_0 denote the trivial n-party protocol in the f-hybrid model that immediately forwards every party's entire input to f in a single ideal call and outputs the response verbatim, without sending any other message. In the execution of π_0^{ρ}, the adversary corrupts a fixed subset $I \subset [n]$ of at most t parties and additionally controls the scheduling of the delivery of every message of ρ arbitrarily (Definition 3.7), including permanently withholding it. Intuitively, we use the dummy composition to reduce the security of ρ against an adversary $\mathcal{A}$ with complete control over message scheduling to the super-rushing security of ρ.

Lemma 4.1. *Let* f *be an* n*-party functionality and let* ρ *be an* R*-round synchronous protocol that* t*-securely computes* f *against super-rushing adversaries (Definition 3.4) with selective abort, computational security and*

black-box straight-line simulation (Remark 3.5), for some $t < n$. Then for every PPT *real-model adversary $\mathcal{A}$ for π_0^ρ and every $I \subset [n]$ with $|I| \leq t$, there exists a* PPT *hybrid-model adversary $\mathcal{S}_{0,\mathsf{SA}}$, that is black-box and straight-line in $\mathcal{A}$, such that*

$$\mathsf{HYBRID}^{f,\mathsf{SA}}_{\pi_0,I,\mathcal{S}_{0,\mathsf{SA}}} \stackrel{c}{\equiv} \mathsf{REAL}_{\pi_0^\rho,I,\mathcal{A}} \ .$$

Moreover, $\mathcal{S}_{0,\mathsf{SA}}$ forwards the message $(\mathtt{abort}, J_{\mathsf{SR}})$ of the simulator of ρ to the trusted party unmodified, and withholds the output from f of exactly the honest parties that fail to locally complete all R rounds of ρ in the real execution, which keep waiting for their output.

Proof. Fix a real-model adversary $\mathcal{A}$ for π_0^ρ and $I \subset [n]$ with $|I| \leq t$. Without loss of generality, $\mathcal{A}$ outputs its view, which determines the sequence of its queries. We reduce $\mathcal{A}$ to a super-rushing adversary $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$ for a standalone execution of ρ itself.

Reduction $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$

State: A set $J \leftarrow \emptyset$.

Execution: Invoke $\mathcal{A}$, handling its queries as follows.

1. **Corruptions.** Forward every corruption $\mathcal{A}$ issues directly to the standalone execution of ρ.
2. **Corrupted parties' messages.** Forward every message $\mathcal{A}$ sends on behalf of a corrupted party directly to the standalone execution.
3. **Delivery requests.** Forward every one of $\mathcal{A}$'s requests for the delivery of an honest party's message (of any round, to any recipient) directly to the standalone execution.
4. **Output.** When $\mathcal{A}$ halts:
   (a) Set J to be the set of parties whose R rounds of ρ are not yet complete at this point.
   (b) Request the delivery of the rest of the protocol in an arbitrary manner and send some arbitrary messages on behalf of the corrupted parties. Note that the execution in this step is fixed.
   (c) Output whatever $\mathcal{A}$ outputs.

Figure 1: The super-rushing adversary $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$ for ρ corrupting I, given oracle access to the real-model adversary $\mathcal{A}$ for π_0^ρ, and the interface to a single standalone execution of ρ.

$\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$ is super-rushing. Note that until $\mathcal{A}$ halts, the adversary $\mathcal{A}$ and $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$ both execute the same steps. Crucially, both $\mathcal{A}$ and $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$ only request messages from honest parties corresponding to a round r, only when these honest parties have received all their messages corresponding to all rounds $\leq r-1$. Finally, the execution of $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$ arbitrarily completes the partial execution of the protocol by performing certain fixed requests for honest parties' messages while providing the corrupted parties' messages. Note that $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$ requests the delivery of all messages in the protocol. Therefore, $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$ is a valid super-rushing adversary.

The simulator $\mathcal{S}_{0,\mathsf{SA}}$. Since ρ t-securely computes f against super-rushing adversaries with black-box straight-line simulation, there is a single PPT, black-box and straight-line simulator, that can simulate the view of $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$. Let this simulator that has black-box access to $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$, be called $\mathcal{S}_{\mathsf{SR}}$.

Simulator $\mathcal{S}_{0,\mathsf{SA}}$

Oracles: The super-rushing adversary $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$ (Figure 1) and the black-box, straight-line simulator $\mathcal{S}_{\mathsf{SR}}$ that gives ρ's super-rushing security.

Execution: Invoke $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$, handling its queries as follows.

1. **Inputs.** Once an honest party sends its ideal message (its input) to the trusted party, of which $\mathcal{S}_{0,\mathsf{SA}}$ observes the metadata, deliver it to the trusted party immediately.
2. **Emulating the reduction.** Route every corruption, corrupted-party send, and delivery request $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$ issues to $\mathcal{S}_{\mathsf{SR}}$ instead, forwarding $\mathcal{S}_{\mathsf{SR}}$'s response back to $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$.
3. **Trusted-party queries.** Forward every query $\mathcal{S}_{\mathsf{SR}}$ makes to the trusted party for f, and every response, unmodified, to and from $\mathcal{S}_{0,\mathsf{SA}}$'s own external trusted party, including the message $(\mathtt{abort}, J_{\mathsf{SR}})$ of $\mathcal{S}_{\mathsf{SR}}$.
4. **Output.** When $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$ halts:
   (a) Let J be the set computed by $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$ when $\mathcal{A}$ halts.
   (b) Deliver the output from the trusted party to every honest party P_j with $j \notin J$, and withhold the output of every honest party P_j with $j \in J$, which hence keeps waiting.
   (c) Output whatever $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$ outputs and halt.

Figure 2: The simulator $\mathcal{S}_{0,\mathsf{SA}}$ for π_0^ρ corrupting I, given oracle access to the super-rushing adversary $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$, the simulator $\mathcal{S}_{\mathsf{SR}}$ for ρ, and the trusted party for f.

Black-box, straight-line simulation. $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$ (Figure 1) is black-box and straight-line in $\mathcal{A}$ (it only ever forwards $\mathcal{A}$'s queries, in the order issued, without rewinding), and $\mathcal{S}_{\mathsf{SR}}$ is black-box and straight-line in $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$. Hence, $\mathcal{S}_{0,\mathsf{SA}}$ is black-box and straight-line in $\mathcal{A}$.

Security. We show security via the following sequence of hybrids:

Hybrid Hyb_0. The real execution $\mathsf{REAL}_{\pi_0^\rho, I, \mathcal{A}}$.

Hybrid Hyb_1. This hybrid is almost identical to Hyb_0, except that $\mathcal{A}$ interacts with the standalone execution of ρ through $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$ (Figure 1), and the output of every party in the resulting set J is withheld, i.e., the party obtains no output and keeps waiting, in place of whatever it would otherwise obtain.

Hybrid Hyb_2. This hybrid is almost identical to Hyb_1, except that $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$'s standalone execution of ρ is replaced by an interaction with $\mathcal{S}_{\mathsf{SR}}$ and the trusted party for f. I.e., $\mathsf{REAL}_{\rho, I, \mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}}$ is replaced by $\mathsf{IDEAL}^{\mathsf{SA}}_{f, I, \mathcal{S}_{\mathsf{SR}}}$, with the output of every party in J still withheld, in place of the output it would otherwise obtain from the trusted party.

Hybrid Hyb_3. The hybrid execution $\mathsf{HYBRID}^{f,\mathsf{SA}}_{\pi_0, I, \mathcal{S}_{0,\mathsf{SA}}}$.

$\mathsf{Hyb}_0 \equiv \mathsf{Hyb}_1$. Before $\mathcal{A}$ halts, $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$ only relays $\mathcal{A}$'s messages on behalf of the corrupted parties, and delivery requests, verbatim, to and from the standalone execution of ρ. In Hyb_0, the protocol π_0^ρ corresponds (with a syntactic change) to exactly a standalone execution of ρ. As a consequence, every message any party sends or receives, and hence every party's output, is identically distributed in both hybrids. A party P_j is included in J by $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$, only when it never receives some prescribed messages as per ρ. Therefore, P_j is included in J in Hyb_1 if and only if it never completes the protocol in Hyb_0, i.e., if and only if it obtains no output and keeps waiting in Hyb_0, exactly as in Hyb_1. Every other honest party obtains the same output in both hybrids, since its output is determined at the point at which $\mathcal{A}$ halts. Moreover, since π_0 sends no messages other than the ideal call, the view of $\mathcal{A}$ does not depend on when the honest parties obtain their outputs. Therefore, the two hybrids must be identically distributed.

$\mathsf{Hyb}_1 \overset{c}{\equiv} \mathsf{Hyb}_2$. Hyb_1 is identically distributed to the real execution of ρ with the super-rushing adversary $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$, where $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$ deterministically computes the set J and completes the rest of the execution (after $\mathcal{A}$ halts). Hyb_2 is syntactically identical to the ideal world execution of f with the outputs of the parties in the same set J withheld. The set J is a deterministic function of the queries of $\mathcal{A}$, and hence of the output of $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$, which is the view of $\mathcal{A}$. Hence, Hyb_1 and Hyb_2 are deterministic functions of the real and ideal world executions of ρ, respectively, in a super-rushing execution. From the computational security of ρ in the presence of the super-rushing adversary $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$ with the simulator $\mathcal{S}_{\mathsf{SR}}$, it holds that $\mathsf{REAL}_{\rho, I, \mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}} \overset{c}{\equiv} \mathsf{IDEAL}^{\mathsf{SA}}_{f, I, \mathcal{S}_{\mathsf{SR}}}$, or $\mathsf{Hyb}_1 \overset{c}{\equiv} \mathsf{Hyb}_2$.

$\mathsf{Hyb}_2 \equiv \mathsf{Hyb}_3$. By construction, $\mathcal{S}_{0,\mathsf{SA}}$ performs exactly the computation Hyb_2 describes: it runs $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$ with $\mathcal{S}_{\mathsf{SR}}$ for every query prior to $\mathcal{A}$ halting, forwards the same message $(\mathtt{abort}, J_{\mathsf{SR}})$ to the trusted party, so that the parties in J_{SR} obtain $\perp$ exactly as in Hyb_2, delivers y_j to every other honest party not in J, and withholds the outputs of the parties in the same set J. Since the trusted party receives the inputs of the honest parties (Step 1) and the inputs of the corrupted parties from $\mathcal{S}_{\mathsf{SR}}$, exactly as in Hyb_2, the outputs $(y_j)_{j \notin I}$ are identical in both hybrids. Hence, the two hybrids Hyb_2 and Hyb_3 are identically distributed.

From the above, $\mathsf{REAL}_{\pi_0^{\rho}, I, \mathcal{A}} \overset{c}{\equiv} \mathsf{HYBRID}^{f,\mathsf{SA}}_{\pi_0, I, \mathcal{S}_{0,\mathsf{SA}}}$, as required. □

On private channels and information-theoretic security. We note that the proof of Lemma 4.1 assumes authenticated channels to identify the identity of the senders and recipients of messages observed on the network. On the other hand, information-theoretically secure protocols typically additionally assume the existence of ideally private channels between the parties. Observe that $\mathcal{A}$ and as a consequence $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$ may observe the messages between honest parties' in clear. Since we consider adversaries that are strictly stronger (in terms of the network), adapting the above proof for when $\mathcal{A}$ (resp. $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$) do not observe the content of the messages sent on the network between honest parties, is straightforward.

Moreover, computational security under concurrent general composition is obtained only because ρ is only assumed to be computationally secure. Consequently, Lemma 4.1 holds verbatim with "computational" replaced by "statistical" throughout, and $\overset{c}{\equiv}$ by $\overset{s}{\equiv}$, whenever ρ (in standalone setting) is statistically super-rushing secure. Similarly, Lemma 4.1 holds verbatim for perfect security, with $\overset{c}{\equiv}$ replaced by $\equiv$, whenever ρ is perfectly super-rushing secure.

4.2 Start and End Synchronisation

Start synchronisation. A synchronous protocol ψ is said to have *start synchronisation* if ψ begins with the following instruction:

- Send $(\textsc{hello}, \psi)$ to all parties; and
- Proceed only upon receiving $(\textsc{hello}, \psi)$ from all n parties.

Naturally, if a protocol does not have start synchronisation, it can be easily compiled to have start synchronisation by adding the above instructions in a special round 0 before the execution of the protocol.

End synchronisation. A synchronous R-round protocol ρ is compiled to have *end synchronisation* by appending the following round $R+1$:

- Upon completing the R rounds of ρ, send $(\textsc{goodbye}, \rho)$ to all parties; and
- Output the output of ρ only upon receiving the round-$(R+1)$ message $(\textsc{goodbye}, \rho)$ from all n parties.

A corrupted party may influence the end synchronisation by not sending a $(\textsc{goodbye}, \rho)$ message. We consider two variants: *point-to-point* end synchronisation, where the round-$(R+1)$ messages are sent over the point-to-point channels, and *broadcast* end synchronisation, where they are sent over the atomic broadcast channel. We denote the resulting $(R+1)$-round protocol by ρ_+ and specify the variant whenever it matters. Finally, we denote by $\widehat{\rho}$ the $(R+2)$-round protocol with both start synchronisation (round 0) and end synchronisation (round $R+1$).

4.3 Selective Abort Under Concurrent General Composition

Theorem 4.2 (Selective Abort via Start and End Synchronisation)**.** *Let f be an n-party functionality and let ρ be an R-round synchronous protocol that t-securely computes f against super-rushing adversaries (Definition 3.4) with selective abort (Section 3.5), computational (resp. statistical, perfect) security and black-box straight-line simulation (Remark 3.5), for some $t < n$. Let $\widehat{\rho}$ be the $(R+2)$-round protocol obtained by adding start synchronisation and point-to-point end synchronisation to ρ. Then for every n-party protocol π in the f-hybrid model that makes a single ideal call to f, every* PPT *real-model adversary $\mathcal{A}$ for $\pi^{\widehat{\rho}}$, and*

every $I \subset [n]$ with $|I| \leq t$, there exists a PPT *hybrid-model adversary $\mathcal{S}_{\mathsf{SA}}$, black-box and straight-line in $\mathcal{A}$, such that*

$$\mathsf{HYBRID}^{f,\mathsf{SA}}_{\pi,I,\mathcal{S}_{\mathsf{SA}}} \stackrel{c}{\equiv} \mathsf{REAL}_{\pi^{\widehat{\rho}},I,\mathcal{A}} \ ,$$

and respectively, $\stackrel{s}{\equiv}$ for statistical security and $\equiv$ for perfect security.

Before proving the theorem, we identify two instants in the execution of $\pi^{\widehat{\rho}}$, determined by the scheduling of $\mathcal{A}$.

- T_0 is the first instant at which some honest party receives a round-0 message from all n parties. At T_0, every honest party has sent (HELLO, ρ), i.e., every honest party has invoked $\widehat{\rho}$ and its input to ρ is fixed. Moreover, prior to T_0, no honest party sends any message of ρ.
- T is the first instant at which some honest party P_j receives a round-$(R+1)$ message from all n parties, i.e., the first point at which some honest party obtains its output from $\widehat{\rho}$. Since every honest party P_k sends its round-$(R+1)$ message only upon completing the R rounds of ρ, at T, *every* honest party has completed the R rounds of ρ. Consequently, at T: (i) every message of the rounds $1, \ldots, R$ from a corrupted party to an honest party has been delivered; (ii) every message that an honest party sends in $\widehat{\rho}$ is already determined by its input, its random coins and its incoming messages; and (iii) the output of every honest party from ρ is determined. Moreover, prior to T, no honest party has an output from $\widehat{\rho}$, and hence, the messages of the honest parties in π prior to T do not depend on the execution of $\widehat{\rho}$ beyond the honest inputs to it.

Intuitively, start synchronisation guarantees that the honest inputs to ρ are fixed before any message of ρ is sent. End synchronisation guarantees that the adversary obtains no information that depends on the outputs of ρ (through the subsequent messages of π) before all its messages of ρ to the honest parties are fixed.

Proof. Fix a real-model adversary $\mathcal{A}$ for $\pi^{\widehat{\rho}}$ and $I \subset [n]$ with $|I| \leq t$. Let $\mathcal{S}_{\mathsf{SR}}$ be the black-box straight-line simulator for ρ against super-rushing adversaries. Only the R rounds of ρ are simulated by $\mathcal{S}_{\mathsf{SR}}$, while the HELLO (start synchronisation, round 0) and GOODBYE (end synchronisation, round $R+1$) messages are handled internally, by both the simulator and the reduction below. Since the adversary schedules the delivery of every message, the execution ends when the adversary halts. An honest party that has not obtained its output from $\widehat{\rho}$ (resp. from f) by then never obtains it, and in particular, does not obtain $\perp$, which it could act upon in π. In other words, the output of every honest party in π is whatever it has output by then, as determined by π. Accordingly, the simulator below never delivers $\perp$ in place of an output that is withheld in the real execution, but rather withholds the output from f.

Simulator $\mathcal{S}_{\mathsf{SA}}$

State: A buffer $M \leftarrow \emptyset$.

Execution: Invoke $\mathcal{A}$, handling its queries as follows.

1. **Messages of π.** Forward every message and delivery request of π, between $\mathcal{A}$ and the hybrid execution, unmodified.
2. **Start synchronisation.** Once an honest party P_j sends its ideal message to the trusted party (of which $\mathcal{S}_{\mathsf{SA}}$ observes the metadata), deliver it to the trusted party immediately, consider P_j to have sent (HELLO, ρ) to all parties, and deliver these messages to $\mathcal{A}$ upon request.
3. **Messages of ρ until T.**
   (a) At T_0, invoke $\mathcal{S}_{\mathsf{SR}}$, and forward to it every message of ρ that $\mathcal{A}$ has sent on behalf of a corrupted party prior to T_0, in the order in which they were sent.
   (b) Until T, forward every message and delivery request of ρ between $\mathcal{A}$ and $\mathcal{S}_{\mathsf{SR}}$, unmodified.
4. **End synchronisation.**

(a) At any point in time, respond to every request of $\mathcal{A}$ for the round-$(R+1)$ message of an honest party P_j with $(\text{GOODBYE}, \rho)$. Since $\mathcal{A}$ requests it only after P_j has completed the R rounds of ρ (and the round-0), which $\mathcal{S}_{\mathsf{SA}}$ can determine from the scheduling.

(b) Upon a round-$(R+1)$ message from a corrupted party, or the delivery of a round-$(R+1)$ message between honest parties, update the tracked state.

(c) The round-$(R+1)$ messages are never forwarded to $\mathcal{S}_{\mathsf{SR}}$.

5. **Trusted-party queries.**

(a) Forward the query of $\mathcal{S}_{\mathsf{SR}}$ to the trusted party for f.

(b) Upon receiving the outputs of the corrupted parties, forward them to $\mathcal{S}_{\mathsf{SR}}$.

(c) Forward the message $(\texttt{abort}, J_{\mathsf{SR}})$ of $\mathcal{S}_{\mathsf{SR}}$ to the trusted party, and send $(\texttt{abort}, \emptyset)$ if $\mathcal{S}_{\mathsf{SR}}$ halts without sending such a message.

6. **Completion.** At T, or when $\mathcal{A}$ halts before T, complete the execution of $\mathcal{S}_{\mathsf{SR}}$ in a fixed manner:

(a) Send fixed messages on behalf of the corrupted parties for every message of ρ that $\mathcal{A}$ has not yet sent, round by round.

(b) Request every message of ρ from an honest party that $\mathcal{A}$ has not yet requested.

(c) Store the responses of $\mathcal{S}_{\mathsf{SR}}$ in M.

7. **Messages of ρ after T.**

(a) Respond to every request of $\mathcal{A}$ for a message of ρ from an honest party to a corrupted party from M.

(b) Once an honest party P_j has received a round-$(R+1)$ message from all n parties, deliver the output of P_j from the trusted party to P_j.

8. **Output.** Output whatever $\mathcal{A}$ outputs and halt.

Figure 3: The simulator $\mathcal{S}_{\mathsf{SA}}$ for $\pi^{\widehat{\rho}}$ corrupting I, given oracle access to the real-model adversary $\mathcal{A}$, the simulator $\mathcal{S}_{\mathsf{SR}}$ for ρ, and the trusted party for f.

Black-box, straight-line simulation. $\mathcal{S}_{\mathsf{SA}}$ only relays messages between $\mathcal{A}$ and the hybrid execution or $\mathcal{S}_{\mathsf{SR}}$, responds to some requests from the buffer M, responds to the requests for the round-$(R+1)$ messages of the honest parties with $(\text{GOODBYE}, \rho)$, and performs deterministic computation. Since $\mathcal{S}_{\mathsf{SR}}$ is black-box and straight-line in its oracle, and $\mathcal{A}$ is never rewound, $\mathcal{S}_{\mathsf{SA}}$ is black-box and straight-line in $\mathcal{A}$. Moreover, since $\mathcal{S}_{\mathsf{SR}}$ is invoked only at T_0, when every honest party has already sent its input to the trusted party, the trusted party always responds to the query of $\mathcal{S}_{\mathsf{SR}}$.

Security. Suppose, towards a contradiction, that $\mathsf{HYBRID}^{f,\mathsf{SA}}_{\pi,I,\mathcal{S}_{\mathsf{SA}}} \overset{c}{\not\approx} \mathsf{REAL}_{\pi^{\widehat{\rho}},I,\mathcal{A}}$. That is, there exist a PPT distinguisher D and a non-negligible function ϵ, such that for infinitely many $\lambda \in \mathbb{N}$ there is $(\boldsymbol{x}, z) \in (\{0,1\}^*)^{n+1}$ with

$$\left|\Pr\left[D\big(\mathsf{HYBRID}^{f,\mathsf{SA}}_{\pi,I,\mathcal{S}_{\mathsf{SA}}}(\lambda,\boldsymbol{x},z)\big)=1\right] - \Pr\left[D\big(\mathsf{REAL}_{\pi^{\widehat{\rho}},I,\mathcal{A}}(\lambda,\boldsymbol{x},z)\big)=1\right]\right| \;>\; \epsilon(\lambda)\;.$$

Fix such $\lambda, \boldsymbol{x}, z$.

Averaging. Let $\mathcal{T}$ denote the random variable consisting of the random coins of $\mathcal{A}$, the random coins that the honest parties use in π (but not in $\widehat{\rho}$), and all the messages (including the metadata of the ideal messages) delivered prior to T_0. Since π and $\mathcal{A}$ are PPT, every value of $\mathcal{T}$ has length at most some fixed polynomial $p(\lambda)$. Prior to T_0, both executions run π untouched and differ only in how the round-0 messages are generated; these are deterministic and $\mathcal{S}_{\mathsf{SA}}$ generates them exactly as the honest parties do. Hence, $\mathcal{T}$ is identically distributed in both executions. If T_0 never occurs (i.e., $\mathcal{A}$ halts before any honest party completes the start synchronisation), then $\mathcal{T}$ consists of the entire execution and both executions are identical. Therefore, we may restrict our attention to values τ for which T_0 occurs. Writing $\Pr[\,\cdot \mid \tau\,]$ for probabilities

conditioned on $\mathcal{T} = \tau$, by the triangle inequality and since $\sum_\tau \Pr[\mathcal{T} = \tau] = 1$,

$$\epsilon(\lambda) \;<\; \sum_\tau \Pr[\mathcal{T} = \tau] \cdot \left| \Pr[D(\mathsf{HYBRID}^{f,\mathsf{SA}}_{\pi,I,\mathcal{S}_{\mathsf{SA}}}) = 1 \mid \tau] - \Pr[D(\mathsf{REAL}_{\pi^{\widehat{\rho}},I,\mathcal{A}}) = 1 \mid \tau] \right| .$$

Since the weights $\Pr[\mathcal{T} = \tau]$ sum to 1, there exists τ^* in the support of $\mathcal{T}$ such that

$$\left| \Pr[D(\mathsf{HYBRID}^{f,\mathsf{SA}}_{\pi,I,\mathcal{S}_{\mathsf{SA}}}) = 1 \mid \tau^*] - \Pr[D(\mathsf{REAL}_{\pi^{\widehat{\rho}},I,\mathcal{A}}) = 1 \mid \tau^*] \right| \;>\; \epsilon(\lambda) .$$

Fix such a τ^* (note that this does not require the maximum to be attained, and holds even if the support of $\mathcal{T}$ is infinite), and let $\boldsymbol{x}_\rho$ denote the honest parties' inputs to ρ, which are determined by τ^* (since every honest party has invoked $\widehat{\rho}$ by T_0).

The reduction. We construct a super-rushing adversary $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$ for a standalone execution of ρ that receives τ^* as its auxiliary input.

Reduction $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$

State: A buffer $M \leftarrow \emptyset$.

Execution:

1. **Replay.** Run $\mathcal{A}$ with the coins in τ^*, and internally emulate the honest parties of π with the coins in τ^*, replaying τ^* up to T_0, including the round-0 messages.
2. **Until T.**
   (a) Forward every message of ρ that $\mathcal{A}$ has sent on behalf of a corrupted party prior to T_0 (as recorded in τ^*) to the external execution of ρ, in the order in which they were sent.
   (b) Continue the internal emulation of $\mathcal{A}$ and of the honest parties in π (and of the round-0 messages). Forward every message and delivery request of $\mathcal{A}$ for ρ to the external execution of ρ, and every response back to $\mathcal{A}$.
   (c) Handle the round-$(R+1)$ messages internally, exactly as $\mathcal{S}_{\mathsf{SA}}$ does (Figure 3), without forwarding them to the external execution.
3. **Completion.** At T, or when $\mathcal{A}$ halts before T, complete the external execution of ρ exactly as $\mathcal{S}_{\mathsf{SA}}$ completes the execution of $\mathcal{S}_{\mathsf{SR}}$ (Figure 3), and store the responses in M.
4. **Output.** Output the internal state σ of the emulation (including τ^*, the state of $\mathcal{A}$, the state of the honest parties of π, and the scheduling so far) together with M.

Figure 4: The super-rushing adversary $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$ for ρ corrupting I, given oracle access to the real-model adversary $\mathcal{A}$ for $\pi^{\widehat{\rho}}$, the auxiliary input τ^*, and the interface to a single standalone execution of ρ.

$\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$ is super-rushing. Until T, $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$ requests a message of ρ from an honest party only when $\mathcal{A}$ does, i.e., only when the honest party has received all its prescribed messages of the previous rounds. The round-$(R+1)$ messages are handled internally, and do not involve the external execution of ρ. At T, every honest party has completed the R rounds of ρ, so every remaining message of an honest party may be requested. If $\mathcal{A}$ halts before T, $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$ completes the execution round by round. In both cases, $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$ eventually sends every message of the corrupted parties and requests every message of the honest parties. Hence, $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$ is a valid super-rushing adversary. Moreover, $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$ is PPT since $\mathcal{A}$ and π are.

Completing the execution of $\mathcal{A}$. Let Φ be the deterministic polynomial-time procedure that, given the output (σ, M) of $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$ and the outputs $(y_j)_{j \notin I}$ of the honest parties from ρ, resumes the emulation of $\mathcal{A}$ and of the honest parties of π from σ after T exactly as $\mathcal{S}_{\mathsf{SA}}$ does (Figure 3): it responds to the requests of $\mathcal{A}$ for messages of ρ from M, responds to the requests for the round-$(R+1)$ messages of the honest parties with $(\textsc{goodbye}, \rho)$, tracks the round-$(R+1)$ messages, hands y_j to the emulated honest party P_j once P_j has

received a round-$(R+1)$ message from all n parties, and finally outputs the output of $\mathcal{A}$ together with the outputs of the honest parties of π. If $\mathcal{A}$ halted before T, then Φ outputs the output of $\mathcal{A}$ and the outputs of the honest parties of π from σ.

The real execution. We claim that

$$\mathsf{REAL}_{\pi^{\widehat{\rho}},I,\mathcal{A}}(\lambda,\boldsymbol{x},z) \mid \tau^* \;\equiv\; \Phi\big(\mathsf{REAL}_{\rho,I,\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}}(\lambda,\boldsymbol{x}_\rho,\tau^*)\big)\ .$$

Until T, the internal emulation of $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$ together with the external execution of ρ is exactly the execution of $\pi^{\widehat{\rho}}$ conditioned on τ^*: the coins of $\mathcal{A}$ and the coins of the honest parties in π are those of τ^*, the honest parties use the inputs $\boldsymbol{x}_\rho$ and fresh coins in ρ, the round-$(R+1)$ message of an honest party is $(\textsc{goodbye},\rho)$ and is requested only once the party has completed the R rounds of ρ, and hence $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$ responds to its request exactly as the honest party would, and prior to T, the honest parties of π do not depend on the execution of ρ. At T, every message that an honest party sends in ρ after T is already determined, and the only messages of the corrupted parties to the honest parties that may still be sent are the round-$(R+1)$ GOODBYE messages, which are handled internally. Hence, the responses in M are exactly the messages that $\mathcal{A}$ obtains in $\pi^{\widehat{\rho}}$ after T, irrespective of the order in which $\mathcal{A}$ requests them, and the outputs $(y_j)_{j\notin I}$ of ρ are exactly the outputs that the honest parties obtain from $\widehat{\rho}$ in $\pi^{\widehat{\rho}}$, once they complete it. Finally, the completion of the external execution by $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$ does not affect the execution of $\mathcal{A}$, and Φ resumes the execution of $\pi^{\widehat{\rho}}$ exactly as it would continue. If $\mathcal{A}$ halts before T, no honest party obtains an output from $\widehat{\rho}$ in either execution.

The hybrid execution. Similarly, we claim that

$$\mathsf{HYBRID}^{f,\mathsf{SA}}_{\pi,I,\mathcal{S}_{\mathsf{SA}}}(\lambda,\boldsymbol{x},z) \mid \tau^* \;\equiv\; \Phi\big(\mathsf{IDEAL}^{\mathsf{SA}}_{f,I,\mathcal{S}^{\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}}_{\mathsf{SR}}}(\lambda,\boldsymbol{x}_\rho,\tau^*)\big)\ .$$

Conditioned on τ^*, the execution of $\mathcal{S}_{\mathsf{SA}}$ (Figure 3) from T_0 until T is exactly the interaction of $\mathcal{S}_{\mathsf{SR}}$ with $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$: $\mathcal{S}_{\mathsf{SA}}$ forwards the messages of ρ between $\mathcal{A}$ and $\mathcal{S}_{\mathsf{SR}}$, and handles the round-$(R+1)$ messages exactly as $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$ does, while the honest parties of π in the hybrid execution behave exactly as their emulation by $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$, since prior to T they obtain no output from f. The trusted party for f in the hybrid execution receives the honest inputs $\boldsymbol{x}_\rho$ and the inputs of the corrupted parties from $\mathcal{S}_{\mathsf{SR}}$, exactly as in the ideal execution; and since $\mathcal{S}_{\mathsf{SA}}$ forwards the message $(\mathtt{abort},J_{\mathsf{SR}})$ of $\mathcal{S}_{\mathsf{SR}}$, the output of every honest party P_j from f is the output y_j of P_j in the ideal execution (which is $\perp$ if $j\in J_{\mathsf{SR}}$). Note that $\mathcal{S}_{\mathsf{SR}}$ has sent this message by the completion of its execution at T, and prior to T, no honest party obtains an output from f. At T, $\mathcal{S}_{\mathsf{SA}}$ completes the execution of $\mathcal{S}_{\mathsf{SR}}$ exactly as $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$ does, and afterwards behaves exactly as Φ: it responds to the requests of $\mathcal{A}$ from M and delivers y_j to P_j exactly when P_j receives a round-$(R+1)$ GOODBYE message from all n parties.

Conclusion. From the above, the distinguisher that executes Φ and then the distinguisher D (that is assumed to break the security $\widehat{\rho}$ under concurrent general composition), distinguishes $\mathsf{REAL}_{\rho,I,\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}}(\lambda,\boldsymbol{x}_\rho,\tau^*)$ from $\mathsf{IDEAL}^{\mathsf{SA}}_{f,I,\mathcal{S}^{\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}}_{\mathsf{SR}}}(\lambda,\boldsymbol{x}_\rho,\tau^*)$ with advantage greater than $\epsilon(\lambda)$, for infinitely many λ. Since $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$ is a PPT super-rushing adversary for ρ and $\mathcal{S}_{\mathsf{SR}}$ is the black-box straight-line simulator for ρ, this contradicts the super-rushing security of ρ. Hence,

$$\mathsf{HYBRID}^{f,\mathsf{SA}}_{\pi,I,\mathcal{S}_{\mathsf{SA}}} \stackrel{c}{\equiv} \mathsf{REAL}_{\pi^{\widehat{\rho}},I,\mathcal{A}}\ .$$

Statistical and perfect security. For statistical security, the same argument holds verbatim for unbounded $\mathcal{A}$, $\mathcal{S}_{\mathsf{SA}}$ and D, with $\stackrel{s}{\equiv}$ in place of $\stackrel{c}{\equiv}$. Moreover, note that $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$ and Φ need not be PPT, but the above definition of $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$ and Φ suffice. In particular, the polynomial bound on the length of the values of $\mathcal{T}$ is used only to ensure that $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$ is PPT, and is not needed in this case. For perfect security, if the two executions are not identically distributed for some $(\boldsymbol{x},z)$, then they are not identically distributed conditioned on some τ^* in the support of $\mathcal{T}$, and the above argument yields that $\mathsf{REAL}_{\rho,I,\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}} \not\equiv \mathsf{IDEAL}^{\mathsf{SA}}_{f,I,\mathcal{S}^{\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}}_{\mathsf{SR}}}$, contradicting the perfect super-rushing security of ρ. □

On private channels. As in Lemma 4.1, the above proof is stated for adversaries that may observe the content of the messages between honest parties. Adapting the proof to ideally private channels, where $\mathcal{A}$ only observes the metadata of such messages, is straightforward.

4.4 Unanimous Abort Security

Since all parties are not guaranteed to complete the protocol ρ^+ when it is concurrently composed in a protocol π^{ρ^+}, the output decision of the honest parties cannot be *unanimous*. In the setting where the adversary may permanently block messages on the point-to-point network, it seems that unanimity is impossible to achieve. However, we note that under the assumption of an *atomic* broadcast channel, unanimity can be obtained. To that end, we consider *end synchronisation*.

We now show that *broadcast* end synchronisation (Section 4.2) suffices to yield security under concurrent general composition with unanimous abort. Recall that we assume that the broadcast channel is consistent (Section 3.4): a message sent over the broadcast channel that has been delivered to some honest party is eventually delivered, identically, to every honest party, even if the adversary halts in the meantime, whereas a message that has been delivered to no honest party by the time the adversary halts is never delivered. To that end, let ρ be an R-round protocol and let ρ_+ be the $(R+1)$-round protocol that is ρ with broadcast end synchronisation. We first show via the dummy composition, that ρ_+ achieves security under concurrent execution with unanimous abort.

Lemma 4.3. *Let f be an n-party functionality and let ρ be an R-round synchronous protocol that t-securely computes f against super-rushing adversaries (Definition 3.4) with unanimous abort (Section 3.5), computational security and black-box straight-line simulation (Remark 3.5), for some $t < n$. Let ρ_+ be the protocol obtained by appending broadcast end synchronisation to ρ. Then for every* PPT *real-model adversary $\mathcal{A}$ for $\pi_0^{\rho_+}$ and every $I \subset [n]$ with $|I| \leq t$, there exists a* PPT *hybrid-model adversary $\mathcal{S}_{0,\mathsf{UA}}$, that is black-box and straight-line in $\mathcal{A}$, such that*

$$\mathsf{HYBRID}^{f,\mathsf{UA}}_{\pi_0,I,\mathcal{S}_{0,\mathsf{UA}}} \overset{c}{\equiv} \mathsf{REAL}_{\pi_0^{\rho_+},I,\mathcal{A}} \ .$$

Proof. Fix a real-model adversary $\mathcal{A}$ for $\pi_0^{\rho_+}$ and $I \subset [n]$ with $|I| \leq t$. We build on the reduction $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$ and the simulator $\mathcal{S}_{\mathsf{SR}}$ from the proof of Lemma 4.1, extended to additionally track the round-$(R{+}1)$ end-synchronisation broadcasts of ρ_+.

Reduction $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{UA}}$

State: Sets $J \leftarrow \emptyset$ and $D \leftarrow \emptyset$.

Execution: Invoke $\mathcal{A}$, handling every corruption, every message $\mathcal{A}$ sends on behalf of a corrupted party for one of ρ's R rounds, and every delivery request for ρ, exactly as $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$ does (Figure 1), and additionally as follows.

1. **Honest round-$(R{+}1)$ broadcasts.** Whenever $\mathcal{A}$ requests the delivery of the round-$(R+1)$ broadcast of an honest party P_j, respond $(\textsc{goodbye}, \rho)$ on behalf of P_j, without forwarding the request to the standalone execution of ρ. Since $\mathcal{A}$ requests the round-$(R+1)$ message of P_j only after P_j has completed the R rounds of ρ, $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{UA}}$ can determine this message from the scheduling.
2. **Delivered round-$(R{+}1)$ broadcasts.** Whenever $\mathcal{A}$ delivers the round-$(R+1)$ broadcast of a party $k \in [n]$ (honest or corrupted) to an honest party, add k to D. A corrupted party's round-$(R+1)$ broadcast is sent by $\mathcal{A}$ on its behalf, and is never forwarded to the standalone execution of ρ.
3. **Output.** When $\mathcal{A}$ halts:
   (a) Set J to be the set of parties whose R rounds of ρ are not yet complete at this point, exactly as $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$ does.
   (b) Set $\mathsf{AllSent} \leftarrow 1$ if $D = [n]$, i.e., the round-$(R+1)$ broadcast of every party has been delivered to at least one honest party, and $\mathsf{AllSent} \leftarrow 0$ otherwise. Note that $\mathsf{AllSent} = 1$ implies $J = \emptyset$, since an honest party broadcasts its round-$(R+1)$ message only upon completing the R rounds of ρ.

(c) Request the delivery of the rest of the protocol in an arbitrary manner and send some arbitrary messages on behalf of the corrupted parties.

(d) Output whatever $\mathcal{A}$ outputs.

Figure 5: The super-rushing adversary $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{UA}}$ for ρ corrupting I, given oracle access to the real-model adversary $\mathcal{A}$ for $\pi_0^{\rho+}$, and the interface to a single standalone execution of ρ.

$\mathcal{A}^{\mathcal{A}}_{0,\mathsf{UA}}$ is super-rushing. As in the proof of Lemma 4.1, until $\mathcal{A}$ halts, $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{UA}}$ only relays $\mathcal{A}$'s corruptions, corrupted messages of ρ, and delivery requests, verbatim, to and from the standalone execution of ρ. Moreover, the round-$(R{+}1)$ broadcasts are fixed deterministically based on the scheduling. Hence $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{UA}}$ is a valid super-rushing adversary for ρ, exactly as $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$ is.

The simulator $\mathcal{S}_{0,\mathsf{UA}}$. Let $\mathcal{S}_{\mathsf{SR}}$ be the black-box, straight-line simulator from the proof of Lemma 4.1 that gives ρ's super-rushing security. Define $\mathcal{S}_{0,\mathsf{UA}}$ exactly as $\mathcal{S}_{0,\mathsf{SA}}$ in the proof of Lemma 4.1 (Figure 2), running $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{UA}}$ in place of $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$ against $\mathcal{S}_{\mathsf{SR}}$ and the trusted party for f, except that it does not forward the message $(\mathtt{abort}, J_{\mathsf{SR}})$ of $\mathcal{S}_{\mathsf{SR}}$ to the trusted party, but records it and withholds its response to the trusted party until $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{UA}}$ halts, and except at the final step: when $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{UA}}$ halts, with the set J and the bit $\mathsf{AllSent}$ that it has computed, $\mathcal{S}_{0,\mathsf{UA}}$ forwards $(\mathtt{abort}, \emptyset)$ to the external trusted party if $J = \emptyset$, $\mathsf{AllSent} = 1$, and the recorded message of $\mathcal{S}_{\mathsf{SR}}$ is $(\mathtt{abort}, \emptyset)$ (i.e., $J_{\mathsf{SR}} = \emptyset$), and forwards $(\mathtt{abort}, [n] \setminus I)$ otherwise. It delivers the output from the trusted party to every honest party if $J = \emptyset$ and $\mathsf{AllSent} = 1$ (these outputs are all $\perp$ if $J_{\mathsf{SR}} = [n] \setminus I$), and delivers no output otherwise, in which case every honest party keeps waiting. It then outputs whatever $\mathcal{A}$ outputs and halts.

Black-box, straight-line simulation. $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{UA}}$ is black-box and straight-line in $\mathcal{A}$, $\mathcal{S}_{\mathsf{SR}}$ is black-box and straight-line in $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{UA}}$. Hence, exactly as for $\mathcal{S}_{0,\mathsf{SA}}$ in Lemma 4.1, $\mathcal{S}_{0,\mathsf{UA}}$ is black-box and straight-line in $\mathcal{A}$.

Security. We reuse the hybrids $\mathsf{Hyb}_0, \mathsf{Hyb}_1, \mathsf{Hyb}_2$ from the proof of Lemma 4.1, now instantiated with ρ_+ and $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{UA}}$ in place of ρ and $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$. Since ρ_+ only appends a fixed, deterministic round of broadcasts to ρ, and $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{UA}}$ relays every corrupted message and delivery request of ρ's R rounds, and tracks every round-$(R{+}1)$ broadcast and its deliveries, exactly as $\mathcal{A}$ schedules them (with the honest parties' broadcasts and the receipt of round-$(R{+}1)$ broadcasts occurring exactly as prescribed by ρ_+ in both hybrids), the two arguments establishing $\mathsf{Hyb}_0 \equiv \mathsf{Hyb}_1$ and $\mathsf{Hyb}_1 \overset{c}{\equiv} \mathsf{Hyb}_2$ in the proof of Lemma 4.1 go through unchanged: in particular, J and D (hence $\mathsf{AllSent}$) are, in both Hyb_1 and Hyb_2, deterministic functions of $\mathcal{A}$'s queries alone, unaffected by whether $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{UA}}$'s standalone execution of ρ's R rounds is real or replaced by $\mathcal{S}_{\mathsf{SR}}$ and the trusted party for f.

We insert one further hybrid before concluding:

Hybrid $\mathsf{Hyb}_{2.5}$. Identical to Hyb_2, except that the output delivered to the honest parties is read off directly from J and $\mathsf{AllSent}$, as tracked by $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{UA}}$: every honest party $j \in [n] \setminus I$ is delivered y_j (its output from the trusted party for f) if $J = \emptyset$ and $\mathsf{AllSent} = 1$, and is otherwise delivered no output, and keeps waiting.

$\mathsf{Hyb}_2 \equiv \mathsf{Hyb}_{2.5}$. In Hyb_2, each honest party's output is, exactly as in $\pi_0^{\rho+}$, determined by the round-$(R{+}1)$ end-synchronisation: party j outputs y_j if and only if it receives a round-$(R+1)$ broadcast from all n parties, and otherwise keeps waiting. We argue that the same occurs identically, in $\mathsf{Hyb}_{2.5}$. Suppose $J = \emptyset$ and $\mathsf{AllSent} = 1$ (i.e., $D = [n]$): every honest party has then completed ρ's R rounds, and the round-$(R{+}1)$ broadcast of every party has been delivered to at least one honest party by the time $\mathcal{A}$ halts. Since the round-$(R{+}1)$ messages are sent over the *atomic* broadcast channel, which is consistent, each of these n broadcasts is eventually delivered to every honest party, even after $\mathcal{A}$ halts. Hence, every honest party receives a round-$(R{+}1)$ broadcast from all n parties and outputs y_j, exactly as $\mathsf{Hyb}_{2.5}$. Conversely, suppose $J \neq \emptyset$ or $\mathsf{AllSent} = 0$: then by the time $\mathcal{A}$ halts, the round-$(R{+}1)$ broadcast of some party $k^* \in [n]$ has been delivered to no honest party, either since k^* never broadcasts it (an honest $k^* \in J$ never completes ρ's R rounds and so, by the end-synchronisation instructions, never reaches the broadcast step at all; a corrupted

k^* may simply never be instructed by $\mathcal{A}$ to broadcast it), or since $\mathcal{A}$ never delivers it. By the consistency of the broadcast channel, this broadcast is then never delivered. Every honest party's output for ρ_+ requires receiving a round-$(R+1)$ broadcast from every one of the n parties, including k^*; hence no honest party ever satisfies this condition, and every honest party obtains no output and keeps waiting, exactly as in $\mathsf{Hyb}_{2.5}$. Hence $\mathsf{Hyb}_2 \equiv \mathsf{Hyb}_{2.5}$.

$\mathsf{Hyb}_{2.5} \equiv \mathsf{HYBRID}^{f,\mathsf{UA}}_{\pi_0,I,\mathcal{S}_{0,\mathsf{UA}}}$. By construction, $\mathcal{S}_{0,\mathsf{UA}}$ performs exactly the computation $\mathsf{Hyb}_{2.5}$ describes: it runs $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{UA}}$ against $\mathcal{S}_{\mathsf{SR}}$ for every query prior to $\mathcal{A}$ halting, and its single $(\mathtt{abort}, J')$ message, with $J' \in \{\emptyset, [n] \setminus I\}$ chosen according to whether $J = \emptyset \wedge \mathsf{AllSent} = 1$, induces exactly the output distribution of $\mathsf{Hyb}_{2.5}$, i.e., unanimous output. Hence $\mathsf{Hyb}_{2.5}$ and $\mathsf{HYBRID}^{f,\mathsf{UA}}_{\pi_0,I,\mathcal{S}_{0,\mathsf{UA}}}$ are identically distributed.

From the above, it must hold that $\mathsf{HYBRID}^{f,\mathsf{UA}}_{\pi_0,I,\mathcal{S}_{0,\mathsf{UA}}} \overset{c}{\equiv} \mathsf{REAL}_{\pi_0^{\rho_+},I,\mathcal{A}}$, as required. □

We now show that start synchronisation and broadcast end synchronisation together yield security under concurrent general composition with unanimous abort.

Theorem 4.4 (Unanimous Abort via Start and End Synchronisation)**.** *Let f be an n-party functionality and let ρ be an R-round synchronous protocol that t-securely computes f against super-rushing adversaries (Definition 3.4) with unanimous abort (Section 3.5), computational (resp. statistical, perfect) security and black-box straight-line simulation (Remark 3.5), for some $t < n$. Let $\widehat{\rho}$ be the $(R+2)$-round protocol obtained by adding start synchronisation and broadcast end synchronisation to ρ. Then for every n-party protocol π in the f-hybrid model that makes a single ideal call to f, every* PPT *real-model adversary $\mathcal{A}$ for $\pi^{\widehat{\rho}}$, and every $I \subset [n]$ with $|I| \leq t$, there exists a* PPT *hybrid-model adversary $\mathcal{S}_{\mathsf{UA}}$, black-box and straight-line in $\mathcal{A}$, such that*

$$\mathsf{HYBRID}^{f,\mathsf{UA}}_{\pi,I,\mathcal{S}_{\mathsf{UA}}} \overset{c}{\equiv} \mathsf{REAL}_{\pi^{\widehat{\rho}},I,\mathcal{A}} \ ,$$

and respectively, $\overset{s}{\equiv}$ for statistical security and $\equiv$ for perfect security.

Proof. The proof is identical to the proof of Theorem 4.2, where the round-$(R+1)$ messages are now sent over the broadcast channel (and are still handled internally by the simulator and the reduction, as in the proof of Theorem 4.2), except for the following differences.

The instant T. With broadcast end synchronisation, we redefine T as the first instant at which the round-$(R+1)$ broadcast of every party $k \in [n]$ has been delivered to at least one honest party. Since every honest party broadcasts its round-$(R+1)$ message only upon completing the R rounds of ρ, at T, every honest party has completed the R rounds of ρ, and hence at time T: (i) every message of the rounds $1, \ldots, R$ from a corrupted party to an honest party has been delivered; (ii) every message that an honest party sends in $\widehat{\rho}$ is already determined by its input, its random coins and its incoming messages; and (iii) the output of every honest party from ρ is determined. Moreover, since an honest party obtains its output from $\widehat{\rho}$ only upon receiving a round-$(R+1)$ broadcast from all n parties, which requires T to have occurred, prior to T, no honest party has an output from $\widehat{\rho}$. The simulator, the reduction and the procedure Φ below are defined with respect to this T.

Unanimity of the real execution. By the consistency of the atomic broadcast channel, if T occurs, then every honest party eventually receives the round-$(R+1)$ broadcasts of all n parties, and hence obtains its output from $\widehat{\rho}$, even if $\mathcal{A}$ halts right after T. If $\mathcal{A}$ halts before T, then the round-$(R+1)$ broadcast of some party has been delivered to no honest party, and hence is never delivered, so no honest party obtains its output from $\widehat{\rho}$. Hence, in $\pi^{\widehat{\rho}}$, either all honest parties obtain their output from $\widehat{\rho}$, or none does.

The abort decision of $\mathcal{S}_{\mathsf{UA}}$. Let $\mathcal{S}_{\mathsf{UA}}$ be identical to $\mathcal{S}_{\mathsf{SA}}$ (Figure 3) with the above T, except for the following. In Step 5(c), upon receiving the outputs of the corrupted parties from the trusted party, $\mathcal{S}_{\mathsf{UA}}$ withholds its response to the trusted party. At T, after completing the execution of $\mathcal{S}_{\mathsf{SR}}$, $\mathcal{S}_{\mathsf{UA}}$ forwards the message $(\mathtt{abort}, J_{\mathsf{SR}})$ of $\mathcal{S}_{\mathsf{SR}}$, where $J_{\mathsf{SR}} \in \{\emptyset, [n] \setminus I\}$; if $\mathcal{A}$ halts after T_0 and before T, $\mathcal{S}_{\mathsf{UA}}$ sends $(\mathtt{abort}, [n] \setminus I)$, and delivers no output. In the former case, $\mathcal{S}_{\mathsf{UA}}$ delivers the output of P_j from the trusted party to P_j once P_j has received a round-$(R+1)$ broadcast from all n parties, exactly as in Step 7(b) of $\mathcal{S}_{\mathsf{SA}}$; and additionally, when $\mathcal{A}$ halts after T, $\mathcal{S}_{\mathsf{UA}}$ delivers the output from the trusted party to every honest party

that has not yet received it, before outputting the output of $\mathcal{A}$, mirroring the consistency of the broadcast channel. Hence, if T occurs, every honest party obtains its output from f, which is y_j for every honest party if $J_{\mathsf{SR}} = \emptyset$, and $\perp$ for every honest party otherwise; and if T does not occur, no honest party obtains an output from f, exactly as in the real execution. Note that $\mathcal{S}_{\mathsf{UA}}$ only responds to the trusted party at T or when $\mathcal{A}$ halts; until then, no honest party obtains an output from f, exactly as no honest party obtains an output from $\widehat{\rho}$ prior to T.

With these modifications, let $\mathcal{A}_{\mathsf{UA}}^{\mathcal{A}}$ be the super-rushing adversary for ρ that is defined identically to the reduction $\mathcal{A}_{\mathsf{SA}}^{\mathcal{A}}$ (Figure 4) with the above T, where the round-$(R+1)$ broadcasts are handled internally by recording their deliveries. Let the completion procedure Φ be defined identically, except that when $\mathcal{A}$ halts after T, Φ delivers every round-$(R+1)$ broadcast to every emulated honest party that has not yet received it, as the broadcast channel does, and hands y_j to every emulated honest party P_j that has not yet obtained its output from $\widehat{\rho}$, after which the emulated honest parties of π perform whatever local computation π prescribes. In the real execution, by the argument for unanimity above, the honest parties that have not obtained their outputs from $\widehat{\rho}$ when $\mathcal{A}$ halts after T obtain them afterwards, and these are their outputs from ρ exactly as Φ hands them. If $\mathcal{A}$ halts before T, no honest party obtains its output from $\widehat{\rho}$, and Φ ignores the outputs of ρ. In the hybrid execution, if T occurs, $\mathcal{S}_{\mathsf{UA}}$ forwards the same message $(\mathtt{abort}, J_{\mathsf{SR}})$ as $\mathcal{S}_{\mathsf{SR}}$, and delivers the outputs from the trusted party exactly when Φ hands the outputs from the ideal execution to the emulated honest parties, including upon $\mathcal{A}$ halting after T. If $\mathcal{A}$ halts after T_0 and before T, the trusted party receives $(\mathtt{abort}, [n] \setminus I)$, possibly while $J_{\mathsf{SR}} = \emptyset$. However, in this case Φ ignores the outputs of the honest parties, and $\mathcal{S}_{\mathsf{UA}}$ delivers no output. The same argument as in the proof of Theorem 4.2 then yields that a distinguisher between $\mathsf{HYBRID}_{\pi,I,\mathcal{S}_{\mathsf{UA}}}^{f,\mathsf{UA}}$ and $\mathsf{REAL}_{\pi^{\widehat{\rho}},I,\mathcal{A}}$ yields a distinguisher between $\mathsf{REAL}_{\rho,I,\mathcal{A}_{\mathsf{UA}}^{\mathcal{A}}}$ and $\mathsf{IDEAL}_{f,I,\mathcal{S}_{\mathsf{SR}}^{\mathcal{A}_{\mathsf{UA}}^{\mathcal{A}}}}^{\mathsf{UA}}$, contradicting the super-rushing security of ρ. □

Remark 4.5 (Last round over broadcast)**.** *If every message of the final round R of ρ is sent over the atomic broadcast channel, then round R already serves as broadcast end synchronisation, and the additional round $R+1$ may be omitted, yielding an $(R+1)$-round protocol. Indeed, redefine T as the first instant at which the round-R broadcast of every party has been delivered to some honest party. Since every honest party broadcasts its round-R message only upon completing round $R-1$, at T, every honest party has completed the rounds $1, \ldots, R-1$, and every message of the corrupted parties to the honest parties is fixed: the messages of the rounds $1, \ldots, R-1$ have been delivered, and by the consistency of the broadcast channel, the round-R broadcasts are eventually delivered, identically, to every honest party. Moreover, prior to T, no honest party has an output. Hence, the proof of Theorem 4.4 goes through with this T.*

5 The Compiler with a Setup

In this section, we show how to extend Theorem 4.2 and Theorem 4.4 to protocols ρ, that make use of standard setup assumptions. Specifically, we show that protocols ρ that are secure with *adaptive inputs* in the $\mathcal{G}$-hybrid model (Definition 3.11) with black-box straight-line simulation (Remark 3.12) are secure under concurrent general composition with *localised setup* (Definition 3.13). We assume also that the setup is *stateless* (Definition 3.14), i.e., the interactions of the honest parties with $\mathcal{G}_{\mathsf{sid}}$ do not affect its responses via the interface of the adversary and of the corrupted parties.

Organisation. In Section 5.1, we extend the security under concurrent execution of Lemma 4.1 and Lemma 4.3 to protocols in the $\mathcal{G}$-hybrid model (Lemma 5.1). In Section 5.2, we show that adding start and end synchronisation to such protocols yields security under concurrent general composition, with selective abort for point-to-point end synchronisation (Theorem 5.2) and with unanimous abort for broadcast end synchronisation (Theorem 5.5). Finally, in Section 5.3, we show that security with adaptive inputs cannot be relaxed to the usual notion of security in the $\mathcal{G}$-hybrid model, even for localised and stateless setup (Theorem 5.7).

5.1 Concurrent Execution with Setup

We first extend the security under concurrent execution of Section 4.1 and Section 4.4 to protocols in the $\mathcal{G}$-hybrid model. Recall the dummy composition π_0 that forwards every party's entire input to f in a single ideal call, and outputs the response verbatim. In the composed protocol π_0^ρ, the single execution of ρ uses its own copy of $\mathcal{G}$, and the adversary may interact with it at any point in time.

Lemma 5.1. *Let f be an n-party functionality, let $\mathcal{G}$ be an ideal functionality, and let ρ be an R-round synchronous protocol in the $\mathcal{G}$-hybrid model such that the following holds:*

1. *ρ t-securely computes f against super-rushing adversaries with selective abort, adaptive inputs, and computational (resp. statistical) security (as per Definition 3.11); and*
2. *The underlying simulator for ρ's security is black-box and straight-line (as per Remark 3.12); and*
3. *The setup $\mathcal{G}$ is localised (as per Definition 3.13) and stateless (as per Definition 3.14).*

Then for every PPT *real-model adversary $\mathcal{A}$ for π_0^ρ and every $I \subset [n]$ with $|I| \leq t$, there exists a* PPT *hybrid-model adversary $\mathcal{S}_0$, that is black-box and straight-line in $\mathcal{A}$, such that for every* PPT *environment $\mathcal{Z}$, it holds that*

$$\left|\Pr\left[\mathsf{HYBRID}^{f,\mathsf{SA}}_{\pi_0,I,\mathcal{S}_0,\mathcal{Z}}(\lambda)=1\right]-\Pr\left[\mathsf{REAL}^{\mathcal{G}}_{\pi_0^\rho,I,\mathcal{A},\mathcal{Z}}(\lambda)=1\right]\right|\leq \mathsf{negl}(\lambda)\ ,$$

and respectively, the same holds for every (not necessarily PPT*) adversary $\mathcal{A}$ and environment $\mathcal{Z}$, with a hybrid-model adversary $\mathcal{S}_0$ that need not run in polynomial time, for statistical security. Moreover, if ρ is secure with unanimous abort (with adaptive inputs), and ρ_+ is the protocol obtained by appending broadcast end synchronisation to ρ, then the same holds with $\mathsf{HYBRID}^{f,\mathsf{UA}}_{\pi_0,I,\mathcal{S}_0,\mathcal{Z}}$ and $\mathsf{REAL}^{\mathcal{G}}_{\pi_0^{\rho_+},I,\mathcal{A},\mathcal{Z}}$ in place of $\mathsf{HYBRID}^{f,\mathsf{SA}}_{\pi_0,I,\mathcal{S}_0,\mathcal{Z}}$ and $\mathsf{REAL}^{\mathcal{G}}_{\pi_0^\rho,I,\mathcal{A},\mathcal{Z}}$, respectively.*

Proof. The proof follows the proof of Lemma 4.1 (resp. Lemma 4.3 for unanimous abort). We describe the differences.

The reduction. The super-rushing adversary $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$ (Figure 1) (resp. $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{UA}}$ (Figure 5) for unanimous abort) additionally proceeds as follows.

1. It forwards every interaction of $\mathcal{A}$ with $\mathcal{G}$, at any point in time, to the instance of $\mathcal{G}$ in the standalone execution of ρ.
2. The honest parties interact with $\mathcal{G}$ only via messages of ρ. The reduction schedules these messages exactly as $\mathcal{A}$ schedules them. When $\mathcal{A}$ halts, it completes them in a fixed manner, as every other message.
3. After $\mathcal{A}$ halts, it does not interact with $\mathcal{G}$ on behalf of the corrupted parties.

For unanimous abort, $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{UA}}$ sets $\mathsf{AllSent} \leftarrow 1$ if and only if, when $\mathcal{A}$ halts, the round-$(R+1)$ broadcast of every party $k \in [n]$ has been delivered to at least one honest party. Otherwise, it sets $\mathsf{AllSent} \leftarrow 0$. By the consistency of the atomic broadcast channel (Section 3.4), these are exactly the broadcasts that are eventually delivered to every honest party. Note that $\mathsf{AllSent} = 1$ implies $J = \emptyset$, since an honest party broadcasts its round-$(R+1)$ message only upon completing the R rounds of ρ.

Without loss of generality, $\mathcal{A}$ outputs its view, which determines the sequence of its queries. The reduction outputs whatever $\mathcal{A}$ outputs. The set J and the bit $\mathsf{AllSent}$ are deterministic functions of the queries of $\mathcal{A}$ alone, and hence of the output of the reduction.

Let $\mathcal{S}_{\mathsf{SR}} \stackrel{def}{=} \mathcal{S}^{\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}}$ (resp. $\mathcal{S}^{\mathcal{A}^{\mathcal{A}}_{0,\mathsf{UA}}}$) be the black-box straight-line simulator for ρ (Remark 3.12).

The environment. In both $\mathsf{REAL}^{\mathcal{G}}_{\pi_0^\rho,I,\mathcal{A},\mathcal{Z}}$ and $\mathsf{HYBRID}^{f,\mathsf{SA}}_{\pi_0,I,\mathcal{S}_0,\mathcal{Z}}$, the environment $\mathcal{Z}$ chooses the inputs $\boldsymbol{x}$ to π_0 and the auxiliary input z. It may interact with $\mathcal{G}$ before choosing them and after receiving the output vector (Definition 3.11). Since π_0 forwards the entire input of every party to f, the inputs of the honest parties to ρ are exactly the inputs chosen by $\mathcal{Z}$. Given $\mathcal{Z}$, consider the environment $\mathcal{Z}_0^{\mathcal{Z}}$ for ρ that proceeds as follows.

1. Invoke $\mathcal{Z}$, forwarding every interaction of $\mathcal{Z}$ with $\mathcal{G}$ to the interface of $\mathcal{Z}_0^{\mathcal{Z}}$ to $\mathcal{G}$. Output the inputs $\boldsymbol{x}$ and the auxiliary input z output by $\mathcal{Z}$.
2. Upon receiving the outputs $(y_j)_{j\notin I}$ of the honest parties and the output of the adversary, compute J (resp. J and $\mathsf{AllSent}$) from the output of the adversary, and modify the outputs of the honest parties as follows.
   - For selective abort, withhold y_j for every $j \in J$, i.e., record that P_j has obtained no output and keeps waiting (Section 3.5).
   - For unanimous abort, if $J = \emptyset$ and $\mathsf{AllSent} = 1$, keep every y_j. Otherwise, withhold y_j for every $j \notin I$.

   Hand the resulting output vector, with the output of the adversary unchanged, to $\mathcal{Z}$.
3. Forward every subsequent interaction of $\mathcal{Z}$ with $\mathcal{G}$ to the interface of $\mathcal{Z}_0^{\mathcal{Z}}$ to $\mathcal{G}$. Output the bit output by $\mathcal{Z}$.

Note that $\mathcal{Z}$, and hence $\mathcal{Z}_0^{\mathcal{Z}}$, may interact with $\mathcal{G}$ prior to choosing the inputs.

The hybrids. Every hybrid is the output of $\mathcal{Z}$ in the corresponding experiment.

Hybrid Hyb_0. The real execution $\mathsf{REAL}^{\mathcal{G}}_{\pi_0^\rho,I,\mathcal{A},\mathcal{Z}}$ (resp. $\mathsf{REAL}^{\mathcal{G}}_{\pi_0^{\rho+},I,\mathcal{A},\mathcal{Z}}$ for unanimous abort).

Hybrid Hyb_1. The real execution $\mathsf{REAL}^{\mathcal{G}}_{\rho,I,\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}},\mathcal{Z}_0^{\mathcal{Z}}}$ (resp. $\mathsf{REAL}^{\mathcal{G}}_{\rho,I,\mathcal{A}^{\mathcal{A}}_{0,\mathsf{UA}},\mathcal{Z}_0^{\mathcal{Z}}}$). That is, $\mathcal{A}$ interacts with the standalone execution of ρ, including its $\mathcal{G}$, through the reduction, and $\mathcal{Z}_0^{\mathcal{Z}}$ modifies the output vector as described above. The interactions of $\mathcal{Z}$ with $\mathcal{G}$ are with the same instance of $\mathcal{G}$.

Hybrid Hyb_2. The ideal execution $\mathsf{IDEAL}^{\mathsf{SA}}_{f,I,\mathcal{S}_{\mathsf{SR}},\mathcal{Z}_0^{\mathcal{Z}}}$ (resp. $\mathsf{IDEAL}^{\mathsf{UA}}_{f,I,\mathcal{S}_{\mathsf{SR}},\mathcal{Z}_0^{\mathcal{Z}}}$). That is, the standalone execution of ρ is replaced by an interaction with $\mathcal{S}_{\mathsf{SR}}$ and the trusted party for f. $\mathcal{S}_{\mathsf{SR}}$ responds to every interaction with $\mathcal{G}$, of the reduction and of $\mathcal{Z}$.

Hybrid Hyb_3. The hybrid execution $\mathsf{HYBRID}^{f,\mathsf{SA}}_{\pi_0,I,\mathcal{S}_0,\mathcal{Z}}$ (resp. $\mathsf{HYBRID}^{f,\mathsf{UA}}_{\pi_0,I,\mathcal{S}_0,\mathcal{Z}}$).

The simulator. $\mathcal{S}_0$ is identical to $\mathcal{S}_{0,\mathsf{SA}}$ (Figure 2), and additionally routes every interaction of $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$ and of $\mathcal{Z}$ with $\mathcal{G}$ to $\mathcal{S}_{\mathsf{SR}}$. It hands the inputs of the corrupted parties, which it receives in the hybrid execution, to $\mathcal{S}_{\mathsf{SR}}$. In particular, $\mathcal{S}_0$ computes J from the execution of $\mathcal{A}$ itself, and outputs whatever $\mathcal{A}$ outputs. For unanimous abort, $\mathcal{S}_0$ is identical to $\mathcal{S}_{0,\mathsf{UA}}$ (from the proof of Lemma 4.3), with the same modification, where $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{UA}}$ takes the place of $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$. Recall that once $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{UA}}$ halts and $\mathcal{S}_{\mathsf{SR}}$ has sent $(\mathtt{abort}, J_{\mathsf{SR}})$, $\mathcal{S}_0$ decides as follows.

- If $J = \emptyset$ and $\mathsf{AllSent} = 1$, it forwards $(\mathtt{abort}, J_{\mathsf{SR}})$ to the trusted party, and delivers the output from the trusted party to every honest party.
- Otherwise, it sends $(\mathtt{abort}, [n] \setminus I)$, and delivers no output. Every honest party keeps waiting.

Since $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{SA}}$ (resp. $\mathcal{A}^{\mathcal{A}}_{0,\mathsf{UA}}$) is black-box and straight-line in $\mathcal{A}$, and $\mathcal{S}_{\mathsf{SR}}$ is black-box and straight-line in its oracle, $\mathcal{S}_0$ is black-box and straight-line in $\mathcal{A}$.

Selective abort. Let $p_i \stackrel{def}{=} \mathsf{Pr}[\mathsf{Hyb}_i = 1]$ for $i \in \{0,1,2,3\}$.

- $\mathsf{Hyb}_0 \equiv \mathsf{Hyb}_1$. We compare the two hybrids phase by phase.
  - **Before the execution.** $\mathcal{Z}$ interacts with the same instance of $\mathcal{G}$ in both hybrids. It outputs the same $\boldsymbol{x}$ and z, which are exactly the inputs of the honest parties to ρ.
  - **Until $\mathcal{A}$ halts.** $\mathcal{A}_{0,\mathsf{SA}}^{\mathcal{A}}$ relays every message, delivery request and interaction of $\mathcal{A}$ with $\mathcal{G}$ verbatim. Hence, every message that any party sends or receives, and every interaction with $\mathcal{G}$, is identically distributed in both hybrids.
  - **The outputs.** As in the proof of Lemma 4.1, a party is in J if and only if it never completes ρ in Hyb_0, i.e., if and only if it obtains no output and keeps waiting in Hyb_0. Every other honest party obtains the same output in both hybrids, which is determined when $\mathcal{A}$ halts. Since π_0 sends no messages other than the ideal call, the view of $\mathcal{A}$ does not depend on when the honest parties obtain their outputs. The output of the adversary is the output of $\mathcal{A}$ in both hybrids. Hence, once the outputs of the parties in J are withheld, the output vector handed to $\mathcal{Z}$ is identically distributed in both hybrids.
  - **After the execution.** In Hyb_1, the completion of ρ after $\mathcal{A}$ halts may cause interactions of the honest parties with $\mathcal{G}$. These do not occur in Hyb_0, where these parties are stalled. The only subsequent interactions with $\mathcal{G}$ are those of $\mathcal{Z}$, with the same instance and via the interface of the adversary and of the corrupted parties. Since $\mathcal{G}$ is stateless (Definition 3.14), the responses to these interactions are identically distributed in both hybrids, given the history of interactions with $\mathcal{G}$ until $\mathcal{A}$ halts. By Item 1 of Definition 3.13, conditioning on this history is equivalent to conditioning on the transcript of the execution until $\mathcal{A}$ halts.

  Hence, $p_0 = p_1$.
- $\mathsf{Hyb}_1 \stackrel{c}{\equiv} \mathsf{Hyb}_2$. $\mathcal{A}_{0,\mathsf{SA}}^{\mathcal{A}}$ is a valid super-rushing adversary for ρ, exactly as in the proof of Lemma 4.1, and is PPT whenever $\mathcal{A}$ is. $\mathcal{Z}_0^{\mathcal{Z}}$ is PPT whenever $\mathcal{Z}$ is. By the security of ρ with adaptive inputs (Definition 3.11), with the simulator $\mathcal{S}_{\mathsf{SR}}$ and the environment $\mathcal{Z}_0^{\mathcal{Z}}$,

$$|p_1 - p_2| = \left|\mathsf{Pr}[\mathsf{REAL}_{\rho,I,\mathcal{A}_{0,\mathsf{SA}}^{\mathcal{A}},\mathcal{Z}_0^{\mathcal{Z}}}^{\mathcal{G}}(\lambda) = 1] - \mathsf{Pr}[\mathsf{IDEAL}_{f,I,\mathcal{S}_{\mathsf{SR}},\mathcal{Z}_0^{\mathcal{Z}}}^{\mathsf{SA}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda) \ .$$

  This step requires security with adaptive inputs, since $\mathcal{Z}_0^{\mathcal{Z}}$ may interact with $\mathcal{G}$ prior to choosing the inputs.
- $\mathsf{Hyb}_2 \equiv \mathsf{Hyb}_3$. By construction, $\mathcal{S}_0$ performs exactly the computation that Hyb_2 describes:
  1. It responds to every interaction of $\mathcal{Z}$ with $\mathcal{G}$, both before and after the execution, via $\mathcal{S}_{\mathsf{SR}}$.
  2. It runs $\mathcal{A}_{0,\mathsf{SA}}^{\mathcal{A}}$ with $\mathcal{S}_{\mathsf{SR}}$ until $\mathcal{A}$ halts, and forwards the same message $(\mathtt{abort}, J_{\mathsf{SR}})$ to the trusted party. Hence, the parties in J_{SR} obtain $\perp$, exactly as in Hyb_2.
  3. It delivers the output from the trusted party to every honest party not in the same set J, and withholds the outputs of the parties in J.
  4. It outputs whatever $\mathcal{A}$ outputs, which is the output of the adversary in Hyb_2.

  The trusted party receives the inputs $\boldsymbol{x}$ of the honest parties, and the inputs of the corrupted parties from $\mathcal{S}_{\mathsf{SR}}$, exactly as in Hyb_2. Hence, the outputs $(y_j)_{j \notin I}$ are identical in both hybrids, and $p_2 = p_3$.

Unanimous abort. The hybrids are instantiated as described above, where ρ is assumed to be secure with unanimous abort.

- $\mathsf{Hyb}_0 \equiv \mathsf{Hyb}_1$. Exactly as for selective abort, until $\mathcal{A}$ halts, every message of the first R rounds, and every interaction with $\mathcal{G}$, is identically distributed in both hybrids. Hence, J and AllSent are deterministic functions of the queries of $\mathcal{A}$ alone (as in the proof of Lemma 4.3). Recall that an honest party P_j obtains its output y_j of ρ in Hyb_0 if and only if it receives a round-$(R+1)$ broadcast from all n parties. We consider two cases.

– **Case I:** $\mathsf{AllSent} = 1$**.** Then $J = \emptyset$, i.e., every honest party has completed the R rounds of ρ when $\mathcal{A}$ halts. Moreover, the round-$(R+1)$ broadcast of every party has been delivered to at least one honest party. By the consistency of the atomic broadcast channel, each of these n broadcasts is eventually delivered to every honest party, even after $\mathcal{A}$ halts. Hence, every honest party P_j obtains y_j. This output is determined when $\mathcal{A}$ halts, and is identical to the output of P_j in Hyb_1, which $\mathcal{Z}_0^{\mathcal{Z}}$ keeps.

– **Case II:** $\mathsf{AllSent} = 0$**.** Then the round-$(R+1)$ broadcast of some party $k^* \in [n]$ has been delivered to no honest party when $\mathcal{A}$ halts. Either k^* never broadcasts it (an honest $k^* \in J$, or a corrupted k^*), or $\mathcal{A}$ never delivers it. By the consistency of the atomic broadcast channel, this broadcast is then never delivered. Hence, no honest party ever obtains its output, and every honest party keeps waiting, exactly as $\mathcal{Z}_0^{\mathcal{Z}}$ records in Hyb_1.

In both cases, the outputs of the honest parties in Hyb_0 are exactly the outputs that $\mathcal{Z}_0^{\mathcal{Z}}$ hands to $\mathcal{Z}$ in Hyb_1. The output of the adversary is the output of $\mathcal{A}$ in both. Finally, the completion of ρ after $\mathcal{A}$ halts may cause additional interactions of the honest parties with $\mathcal{G}$ in Hyb_1. Exactly as for selective abort, since $\mathcal{G}$ is stateless (Definition 3.14), and by Item 1 of Definition 3.13, the responses to the subsequent interactions of $\mathcal{Z}$ are identically distributed in both hybrids, given the transcript of the execution until $\mathcal{A}$ halts. Hence, $p_0 = p_1$.

- $\mathsf{Hyb}_1 \stackrel{c}{\equiv} \mathsf{Hyb}_2$. $\mathcal{A}_{0,\mathsf{UA}}^{\mathcal{A}}$ is a valid super-rushing adversary for ρ (as in the proof of Lemma 4.3), and is PPT whenever $\mathcal{A}$ is. $\mathcal{Z}_0^{\mathcal{Z}}$ is PPT whenever $\mathcal{Z}$ is. By the security of ρ with unanimous abort and adaptive inputs, with the simulator $\mathcal{S}_{\mathsf{SR}}$ and the environment $\mathcal{Z}_0^{\mathcal{Z}}$, it holds that $|p_1 - p_2| \leq \mathsf{negl}(\lambda)$.

- $\mathsf{Hyb}_2 \equiv \mathsf{Hyb}_3$. In Hyb_2, $\mathcal{S}_{\mathsf{SR}}$ sends $(\mathtt{abort}, J_{\mathsf{SR}})$ with $J_{\mathsf{SR}} \in \{\emptyset, [n] \setminus I\}$ to the trusted party. If $J = \emptyset$ and $\mathsf{AllSent} = 1$, then $\mathcal{Z}_0^{\mathcal{Z}}$ hands $\mathcal{Z}$ the output y_j of every honest party P_j from the trusted party; these are all $\perp$ if $J_{\mathsf{SR}} = [n] \setminus I$. Otherwise, it withholds every output. By construction, $\mathcal{S}_0$ performs exactly this computation:

  1. It responds to every interaction of $\mathcal{Z}$ with $\mathcal{G}$, both before and after the execution, via $\mathcal{S}_{\mathsf{SR}}$.
  2. It runs $\mathcal{A}_{0,\mathsf{UA}}^{\mathcal{A}}$ with $\mathcal{S}_{\mathsf{SR}}$ until $\mathcal{A}$ halts, and records $(\mathtt{abort}, J_{\mathsf{SR}})$.
  3. If $J = \emptyset$ and $\mathsf{AllSent} = 1$, it forwards $(\mathtt{abort}, J_{\mathsf{SR}})$ and delivers the outputs of all honest parties. Otherwise, it sends $(\mathtt{abort}, [n] \setminus I)$ and delivers no output.
  4. It outputs whatever $\mathcal{A}$ outputs.

  Hence, $p_2 = p_3$.

Conclusion. From the above,

$$\left|\Pr[\mathsf{HYBRID}^{f,\mathsf{SA}}_{\pi_0,I,\mathcal{S}_0,\mathcal{Z}}(\lambda) = 1] - \Pr[\mathsf{REAL}^{\mathcal{G}}_{\pi_0^{\rho},I,\mathcal{A},\mathcal{Z}}(\lambda) = 1]\right| = |p_3 - p_0| \leq \mathsf{negl}(\lambda) \ ,$$

(resp. with $\mathsf{HYBRID}^{f,\mathsf{UA}}_{\pi_0,I,\mathcal{S}_0,\mathcal{Z}}$ and $\mathsf{REAL}^{\mathcal{G}}_{\pi_0^{\rho+},I,\mathcal{A},\mathcal{Z}}$ for unanimous abort), as required. For statistical security, the same argument holds for every (not necessarily PPT) $\mathcal{A}$ and $\mathcal{Z}$, where $\mathcal{Z}_0^{\mathcal{Z}}$ need not run in polynomial time either. □

5.2 Start and End Synchronisation

We now extend the proofs of Theorem 4.2 and Theorem 4.4 to protocols with localised (Definition 3.13) and stateless (Definition 3.14) setup. Crucially, the honest parties interact with a copy of the setup instance, $\mathcal{G}_{\mathsf{sid}}$, only as part of the invocation to ρ, i.e., only after completing the start synchronisation of the instance. The adversary, however, may interact with it at any point in time, and in particular, prior to T_0. Hence, as in Lemma 5.1, the inputs of the honest parties to ρ may depend on the setup, now via the execution of $\pi^{\widehat{\rho}}$ prior to T_0, and we again require security with adaptive inputs.

Theorem 5.2 (Selective Abort with Setup)**.** *Let f be an n-party functionality, let $\mathcal{G}$ be an ideal functionality, and let ρ be an R-round synchronous protocol in the $\mathcal{G}$-hybrid model such that the following holds:*

1. *ρ t-securely computes f against super-rushing adversaries with selective abort, adaptive inputs, and computational (resp. statistical) security (as per Definition 3.11); and*

2. *The underlying simulator for ρ's security is black-box and straight-line (as per Remark 3.12); and*

3. *The setup $\mathcal{G}$ is localised (as per Definition 3.13) and stateless (as per Definition 3.14).*

Let $\widehat{\rho}$ be the $(R+2)$-round protocol obtained by adding start synchronisation and point-to-point end synchronisation to ρ. Then for every n-party protocol π in the f-hybrid model that makes a single ideal call to f, every PPT *real-model adversary $\mathcal{A}$ for $\pi^{\widehat{\rho}}$, and every $I \subset [n]$ with $|I| \leq t$, there exists a* PPT *hybrid-model adversary $\mathcal{S}_{\mathsf{SA}}$, black-box and straight-line in $\mathcal{A}$, such that for every* PPT *environment $\mathcal{Z}$, it holds that*

$$\left|\Pr\left[\mathsf{HYBRID}^{f,\mathsf{SA}}_{\pi,I,\mathcal{S}_{\mathsf{SA}},\mathcal{Z}}(\lambda)=1\right]-\Pr\left[\mathsf{REAL}^{\mathcal{G}}_{\pi^{\widehat{\rho}},I,\mathcal{A},\mathcal{Z}}(\lambda)=1\right]\right| \leq \mathsf{negl}(\lambda)\ ,$$

and respectively, the same holds for every (not necessarily PPT*) adversary $\mathcal{A}$ and environment $\mathcal{Z}$, with a hybrid-model adversary $\mathcal{S}_{\mathsf{SA}}$ that need not run in polynomial time, for statistical security.*

Proof. Fix a PPT real-model adversary $\mathcal{A}$ for $\pi^{\widehat{\rho}}$ and a subset $I \subset [n]$ with $|I| \leq t$. Let $\mathcal{S}$ be the black-box straight-line simulator for ρ against super-rushing adversaries with selective abort and adaptive inputs (Remark 3.12). We first fix conventions, make the two experiments of the theorem explicit, and identify two instants in the execution of $\pi^{\widehat{\rho}}$. We then define a procedure that completes an execution of ρ, the reduction $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$, the simulator $\mathcal{S}_{\mathsf{SA}}$, and for every environment $\mathcal{Z}$, an environment $\mathcal{Z}^{\mathcal{Z}}_{\rho}$ for ρ, and relate the four experiments.

Preliminaries. Since π makes a single ideal call to f, the execution of $\pi^{\widehat{\rho}}$ contains a single instance sid of $\widehat{\rho}$, and hence a single copy $\mathcal{G}_{\mathsf{sid}}$ of the setup, which we denote by $\mathcal{G}$ throughout.

- Since $\mathcal{G}$ is localised, by Item 1 of Definition 3.13, its random coins are independent of the random coins of $\mathcal{Z}$, of $\mathcal{A}$ and of the honest parties. By Item 2 of Definition 3.13, the honest parties never interact with $\mathcal{G}$ as part of π, nor as part of the start and end synchronisation (rounds 0 and $R+1$), which are not part of the code of ρ.

- An interaction of an honest party with $\mathcal{G}$ consists of messages of ρ between the party and $\mathcal{G}$, whose delivery is scheduled by the adversary, as for every other message of ρ. We assume that every such interaction is a query followed by a response, which the party awaits before completing the round.

- The adversary interacts with $\mathcal{G}$ on behalf of the corrupted parties and via the interface of $\mathcal{G}$ to the adversary, at any point in time. We refer to these two jointly as the *adversarial interface* of $\mathcal{G}$. The environment $\mathcal{Z}$, interacts with $\mathcal{G}$ via the same adversarial interface.

- Recall that $\mathcal{G}$ is stateless (Definition 3.14): for every sequence of past queries Q, all additional queries Q' of the honest parties and every party B, it holds that $\mathsf{View}_{\mathsf{B}}(Q) \equiv \mathsf{View}_{\mathsf{B}}(Q,Q')$. That is, given a transcript of the execution, additional interactions of the honest parties with $\mathcal{G}$ do not change the conditional distribution of the responses of $\mathcal{G}$ to subsequent interactions via the adversarial interface. Note that by Item 1 of Definition 3.13, conditioning on the interactions with $\mathcal{G}$ and its responses is equivalent to conditioning on the transcript of the execution.

Since the adversary schedules the delivery of every message, the execution ends when the adversary halts. An honest party that has not obtained its output from $\widehat{\rho}$ (resp. from f) by then never obtains it (except for broadcast end synchronisation, which we handle in the proof of Theorem 5.5), and the output of every honest party in π is whatever it has output by then. The honest parties use independent random coins in π and in $\widehat{\rho}$, and the round-0 and round-$(R+1)$ messages of the honest parties are deterministic. Finally, let x^{def} be a fixed default input to ρ.

The experiments. We make the two experiments of the theorem explicit, following Definition 3.11, with $\pi^{\widehat{\rho}}$ and π in place of ρ and f, respectively. The real experiment $\mathsf{REAL}^{\mathcal{G}}_{\pi^{\widehat{\rho}},I,\mathcal{A},\mathcal{Z}}(\lambda)$ proceeds as follows.

1. $\mathcal{Z}$ interacts with $\mathcal{G}$ via the adversarial interface, and outputs inputs $\boldsymbol{x} = (x_1, \ldots, x_n)$ to π and an auxiliary input z.
2. The parties run $\pi^{\widehat{\rho}}$ on $\boldsymbol{x}$, where $\mathcal{A}$ with auxiliary input z corrupts the parties in I. The honest parties interact with $\mathcal{G}$ only via the code of ρ (by Item 2 of Definition 3.13), and $\mathcal{A}$ interacts with the same instance of $\mathcal{G}$ via the adversarial interface.
3. $\mathcal{Z}$ receives the output vector, i.e., the outputs of the honest parties in π and the output of $\mathcal{A}$, may further interact with $\mathcal{G}$ via the adversarial interface, and outputs a bit, which is the output of the experiment.

The hybrid experiment $\mathsf{HYBRID}^{f,\mathsf{type}}_{\pi,I,\mathcal{S}',\mathcal{Z}}(\lambda)$ for a hybrid-model adversary $\mathcal{S}'$ and $\mathsf{type} \in \{\mathsf{SA}, \mathsf{UA}\}$ is identical, except that $\mathcal{S}'$ responds to every interaction of $\mathcal{Z}$ with $\mathcal{G}$, in both Step 1 and Step 3, above, and Step 2 is the hybrid execution of π on $\boldsymbol{x}$ with $\mathcal{S}'$ (with auxiliary input z) and the trusted party for f with abort of type type. Note that no honest party interacts with a setup in the hybrid execution.

The instants T_0 and T. In an execution of $\pi^{\widehat{\rho}}$, define the following two instants, which are determined by the scheduling of $\mathcal{A}$.

- T_0 is the first instant at which some honest party has received a round-0 message from all n parties.
- T is the first instant at which some honest party has received a round-$(R+1)$ message from all n parties.

We stress that T_0 and T can be determined based on the sequence of queries for messages that $\mathcal{A}$ makes, that is, $\mathcal{A}$'s scheduling. This applies to the reduction and the simulator below, which determine T_0 and T in their executions exactly as above.

Claim 5.3. *The following hold in every execution of $\pi^{\widehat{\rho}}$.*

1. *At T_0, every honest party has invoked $\widehat{\rho}$, and hence its input to ρ, which is its ideal message in π, is fixed. We denote these inputs by $\boldsymbol{x}_\rho = (x_{\rho,j})_{j \notin I}$.*
2. *Prior to T_0, no honest party has sent a message of ρ, nor interacted with $\mathcal{G}$, and $\mathcal{A}$ requests no message of ρ from an honest party. However, $\mathcal{A}$ may send messages of ρ on behalf of the corrupted parties, and may interact with $\mathcal{G}$ via the adversarial interface.*

Proof. For Item 1, an honest party proceeds past round 0 only upon receiving (HELLO, ρ) from all n parties, and an honest party sends (HELLO, ρ) upon invoking $\widehat{\rho}$. Hence, at T_0, every honest party has sent (HELLO, ρ), i.e., has invoked $\widehat{\rho}$. For Item 2, prior to T_0, no honest party has completed round 0. Hence, no honest party has begun ρ, and in particular, it has neither sent a message of ρ nor interacted with $\mathcal{G}$, by Item 2 of Definition 3.13. Since $\mathcal{A}$ is well behaved, it requests the round-r message of an honest party only once the party has received all its messages of the rounds prior to r, including round 0. Hence, $\mathcal{A}$ requests no message of ρ from an honest party prior to T_0. □

Claim 5.4. *The following hold in every execution of $\pi^{\widehat{\rho}}$.*

1. *At T, every message of rounds $1, \ldots, R$ from a corrupted party to an honest party, every message of rounds $1, \ldots, R$ between two honest parties, and every message between an honest party and $\mathcal{G}$ has been delivered. In particular, no honest party interacts with $\mathcal{G}$ after T.*
2. *At T, every message of $\widehat{\rho}$ from an honest party that $\mathcal{A}$ has not yet requested is determined by the input, the random coins and the incoming messages of the party.*
3. *At T, the output of every honest party from ρ is determined.*

4. *Prior to T, no honest party has obtained its output from $\widehat{\rho}$. Hence, prior to T, the messages of the honest parties in π do not depend on the execution of $\widehat{\rho}$, beyond the fact that it has been invoked.*

Proof. Every honest party sends its round-$(R+1)$ message only upon completing the R rounds of ρ. Hence, at T, *every* honest party has completed the R rounds of ρ, i.e., has received all its messages of rounds $1,\ldots,R$, including the responses of $\mathcal{G}$, which implies Item 1. Since the round-$(R+1)$ message of an honest party is $(\text{GOODBYE},\rho)$, and its messages of rounds $1,\ldots,R$ are computed from its incoming messages of the previous rounds, Item 2 and Item 3 follow. Finally, an honest party obtains its output from $\widehat{\rho}$ only upon receiving a round-$(R+1)$ message from all n parties, which first occurs at T, which implies Item 4. □

Completing an execution of ρ. Given an (external or simulated) execution of ρ in which some messages have not yet been sent, requested or delivered, the *completion* of the execution proceeds round by round, for $r=1,\ldots,R$:

(a) Send a fixed default message on behalf of a corrupted party, for every round-r message from a corrupted party to an honest party that has not yet been sent.

(b) Deliver every pending round-r message between two honest parties, or between an honest party and $\mathcal{G}$, until no such message is pending.

(c) Request every round-r message from an honest party to a corrupted party that has not yet been requested, and store the response in a buffer M.

Since the completion proceeds round by round, every request in (c) is made only after the honest party has received all its messages of the previous rounds, as required from a super-rushing adversary. The completion never interacts with $\mathcal{G}$ via the adversarial interface. Hence, the only interactions with $\mathcal{G}$ caused by the completion are the interactions of the honest parties in (b).

The reduction. The reduction $\mathcal{A}_{\mathsf{SA}}^{\mathcal{A}}$ (Figure 6) is a super-rushing adversary for a standalone execution of ρ in the $\mathcal{G}$-hybrid model, corrupting I. Its auxiliary input σ_0 is a record of an execution of $\pi^{\widehat{\rho}}$ until T_0: a flag $\mathsf{halt}\in\{0,1\}$ indicating whether $\mathcal{A}$ halted prior to T_0, the auxiliary input z of $\mathcal{A}$, the random coins of $\mathcal{A}$ and of the honest parties in π, and the sequence of all messages delivered prior to T_0, including the metadata of the ideal messages of π and the responses of $\mathcal{G}$ to $\mathcal{A}$. Such a record is produced by the environment $\mathcal{Z}_{\rho}^{\mathcal{Z}}$ below.

Reduction $\mathcal{A}_{\mathsf{SA}}^{\mathcal{A}}$

State: A buffer $M\leftarrow\emptyset$.

Execution:

1. **Halted.** If $\mathsf{halt}=1$ in σ_0, complete the external execution of ρ, output (σ_0,M) and halt.
2. **Replay.** Run $\mathcal{A}$ with the auxiliary input z and the random coins in σ_0, and internally emulate the honest parties of π with the random coins in σ_0, replaying the record σ_0 up to T_0. During the replay, every message delivered to $\mathcal{A}$, and every response of $\mathcal{G}$ to $\mathcal{A}$, is taken from σ_0; in particular, no interaction with $\mathcal{G}$ takes place.
3. **Messages prior to T_0.** Send every message of ρ that $\mathcal{A}$ has sent on behalf of a corrupted party prior to T_0 (as recorded in σ_0) to the external execution of ρ, in the order in which they were sent.
4. **Until T.** Continue the emulation of $\mathcal{A}$ and of the honest parties of π, and handle the queries of $\mathcal{A}$ as follows.
   (a) *Messages of π.* Handle every message and delivery request of π by the internal emulation of the honest parties of π. Respond to every request of $\mathcal{A}$ for the round-0 message of an honest party with (HELLO,ρ), and record the round-0 messages of the corrupted parties and their deliveries, so as to determine when every honest party completes round 0.

(b) *Messages of ρ.* Send every message of ρ that $\mathcal{A}$ sends on behalf of a corrupted party to the external execution of ρ. Forward every request of $\mathcal{A}$ for a message of ρ from an honest party to a corrupted party to the external execution of ρ, and the response back to $\mathcal{A}$. Upon $\mathcal{A}$ scheduling the delivery of a message of ρ between two honest parties, or between an honest party and $\mathcal{G}$, deliver the same message in the external execution of ρ, and relay to $\mathcal{A}$ whatever the external execution reveals about it.

(c) *Setup.* Forward every interaction of $\mathcal{A}$ with $\mathcal{G}$ via the adversarial interface to the adversarial interface of the $\mathcal{G}$ of the external execution of ρ, and every response back to $\mathcal{A}$.

(d) *Round-$(R+1)$ messages.* Respond to every request of $\mathcal{A}$ for the round-$(R+1)$ message of an honest party with $(\text{GOODBYE}, \rho)$, and record the round-$(R+1)$ messages of the corrupted parties and the deliveries of the round-$(R+1)$ messages between honest parties. These messages are never forwarded to the external execution.

5. **Completion.** At T, or when $\mathcal{A}$ halts before T, complete the external execution of ρ, storing the responses in M.
6. **Output.** Output (σ, M), where σ is the record of the emulation until T (or until $\mathcal{A}$ halts), defined as σ_0 and extended by all messages delivered to $\mathcal{A}$ and responses of $\mathcal{G}$ to $\mathcal{A}$ since T_0, together with an indication of whether $\mathcal{A}$ halted before T.

Figure 6: The super-rushing adversary $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$ for ρ in the $\mathcal{G}$-hybrid model corrupting I, given oracle access to the real-model adversary $\mathcal{A}$ for $\pi^{\widehat{\rho}}$, the auxiliary input σ_0, and the interface to a single standalone execution of ρ, including the adversarial interface of its $\mathcal{G}$.

$\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$ is a valid super-rushing adversary. In the external execution of ρ, an honest party begins round 1 immediately, whereas in $\widehat{\rho}$, it begins round 1 only upon completing round 0. This does not affect the external execution, since $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$ sends, requests and delivers messages of ρ only when $\mathcal{A}$ does in the emulation. If $\mathcal{A}$ requests the round-r message of an honest party P_j, then P_j has received all its messages of rounds $0, \ldots, r-1$ in the emulated execution of $\widehat{\rho}$, and since the messages of rounds $1, \ldots, r-1$ are delivered identically in the external execution, the request is valid in the external execution. Similarly, if $\mathcal{A}$ schedules the delivery of a message between two honest parties, or between an honest party and $\mathcal{G}$, then this message has been sent in the emulated execution, and hence also in the external execution. The round-0 and round-$(R+1)$ messages are handled internally, and do not involve the external execution. Finally, the completion sends every remaining message of the corrupted parties, delivers every remaining message between honest parties or between an honest party and $\mathcal{G}$, and requests every remaining message of the honest parties, in a valid order. Hence, $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$ is a valid super-rushing adversary for ρ. It is PPT, since $\mathcal{A}$ and π are, and its output has polynomial size.

The simulator. Recall the ideal execution of Definition 3.10. Given oracle access to a super-rushing adversary $\mathcal{A}'$ for ρ, the simulator $\mathcal{S}^{\mathcal{A}'}$ proceeds in three phases. In the *pre-execution phase*, it responds to the interactions of the environment with $\mathcal{G}$. In the *execution phase*, it receives the auxiliary input and the inputs of the corrupted parties. $\mathcal{S}^{\mathcal{A}'}$ then interacts with the trusted party for f, and interacts with its oracle, towards which it plays the roles of the honest parties of ρ and of $\mathcal{G}$. The execution phase ends with the output of $\mathcal{S}^{\mathcal{A}'}$. In the *post-execution phase*, it again responds to the queries of the environment $\mathcal{Z}$ to $\mathcal{G}$. We refer to these simulated interactions of the environment with $\mathcal{G}$ as the *environment interface* of $\mathcal{S}^{\mathcal{A}'}$, and to the simulated interactions of its oracle with $\mathcal{G}$ as its *oracle interface*. By Remark 3.12, $\mathcal{S}^{\mathcal{A}'}$ never rewinds its oracle or the environment, accesses the auxiliary input only through its oracle, and its output is the output of its oracle, which it does not otherwise use.

Let $\mathcal{S}_{\mathsf{SR}} \stackrel{def}{=} \mathcal{S}^{\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}}$. Note that $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$, and hence $\mathcal{S}_{\mathsf{SR}}$, depend only on $\mathcal{A}$, π and I, and not on the environment. The simulator $\mathcal{S}_{\mathsf{SA}}$ is given in Figure 7.

Simulator $\mathcal{S}_{\mathsf{SA}}$

State: A buffer $M \leftarrow \emptyset$, and a list $L \leftarrow \emptyset$ of messages of ρ sent by $\mathcal{A}$ prior to T_0.

Pre-execution: Invoke $\mathcal{S}_{\mathsf{SR}}$ in its pre-execution phase. Respond to every interaction of $\mathcal{Z}$ with $\mathcal{G}$ via the environment interface of $\mathcal{S}_{\mathsf{SR}}$.

Execution: Upon receiving the auxiliary input z, invoke $\mathcal{A}$ with auxiliary input z, and handle its queries as follows.

1. **Messages of π.**
   (a) Forward every message and delivery request of π between $\mathcal{A}$ and the hybrid execution, unmodified.
   (b) Once an honest party P_j sends its ideal message to the trusted party (of which $\mathcal{S}_{\mathsf{SA}}$ observes the metadata), deliver it to the trusted party immediately, and consider P_j to have sent (HELLO, ρ) to all parties.
   (c) Respond to every request of $\mathcal{A}$ for the round-0 message of an honest party with (HELLO, ρ), provided that it has been sent, and record the round-0 messages of the corrupted parties and their deliveries.
2. **Prior to T_0.** Respond to every interaction of $\mathcal{A}$ with $\mathcal{G}$ via the environment interface of $\mathcal{S}_{\mathsf{SR}}$. Append every message of ρ that $\mathcal{A}$ sends on behalf of a corrupted party to L.
3. **From T_0 until T.** At T_0, start the execution phase of $\mathcal{S}_{\mathsf{SR}}$, with the input x^{def} for every corrupted party, where the oracle of $\mathcal{S}_{\mathsf{SR}}$ is implemented by the running execution of $\mathcal{A}$, as follows.
   (a) Send the messages in L to $\mathcal{S}_{\mathsf{SR}}$, in order, as messages of the corrupted parties.
   (b) Forward every message of ρ that $\mathcal{A}$ sends on behalf of a corrupted party, every request of $\mathcal{A}$ for a message of ρ from an honest party to a corrupted party, and every delivery of a message of ρ between two honest parties, or between an honest party and $\mathcal{G}$, scheduled by $\mathcal{A}$, to $\mathcal{S}_{\mathsf{SR}}$, and the responses back to $\mathcal{A}$.
   (c) Forward every interaction of $\mathcal{A}$ with $\mathcal{G}$ to the oracle interface of $\mathcal{S}_{\mathsf{SR}}$, and the responses back to $\mathcal{A}$.
   (d) Respond to every request of $\mathcal{A}$ for the round-$(R+1)$ message of an honest party with $(\text{GOODBYE}, \rho)$, and record the round-$(R+1)$ messages of the corrupted parties and the deliveries of the round-$(R+1)$ messages between honest parties. These messages are never forwarded to $\mathcal{S}_{\mathsf{SR}}$.
   (e) Forward the query of $\mathcal{S}_{\mathsf{SR}}$ to the trusted party for f, the outputs of the corrupted parties back to $\mathcal{S}_{\mathsf{SR}}$, and the message $(\texttt{abort}, J_{\mathsf{SR}})$ of $\mathcal{S}_{\mathsf{SR}}$ to the trusted party.
4. **Completion.** At T, or when $\mathcal{A}$ halts after T_0 and before T, complete the execution of ρ simulated by $\mathcal{S}_{\mathsf{SR}}$, storing the responses in M, and let the execution phase of $\mathcal{S}_{\mathsf{SR}}$ end.
5. **After T.**
   (a) Respond to every request of $\mathcal{A}$ for a message of ρ from an honest party to a corrupted party from M.
   (b) Handle the round-$(R+1)$ messages as in Step 3(d). Once an honest party P_j has received a round-$(R+1)$ message from all n parties, deliver the output of P_j from the trusted party to P_j.
   (c) Respond to every interaction of $\mathcal{A}$ with $\mathcal{G}$ via the environment interface of $\mathcal{S}_{\mathsf{SR}}$.
6. **Halting prior to T_0.** If $\mathcal{A}$ halts before T_0, start the execution phase of $\mathcal{S}_{\mathsf{SR}}$ with the input x^{def} for every corrupted party, where its oracle is implemented by completing the execution of ρ simulated by $\mathcal{S}_{\mathsf{SR}}$. Respond to the query of $\mathcal{S}_{\mathsf{SR}}$ to the trusted party by computing f on the input x^{def} for every honest party and the inputs of the corrupted parties sent by $\mathcal{S}_{\mathsf{SR}}$, and handing $\mathcal{S}_{\mathsf{SR}}$ the outputs of the corrupted parties, and discard its message $(\texttt{abort}, J_{\mathsf{SR}})$.
7. **Output.** When $\mathcal{A}$ halts, output whatever $\mathcal{A}$ outputs.

Post-execution: Respond to every interaction of $\mathcal{Z}$ with $\mathcal{G}$ via the environment interface of $\mathcal{S}_{\mathsf{SR}}$.

Figure 7: The hybrid-model adversary $\mathcal{S}_{\mathsf{SA}}$ for π corrupting I, given oracle access to the real-model adversary $\mathcal{A}$, the simulator $\mathcal{S}_{\mathsf{SR}} = \mathcal{S}^{\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}}$ for ρ, and the trusted party for f.

Black-box, straight-line simulation. $\mathcal{S}_{\mathsf{SA}}$ invokes $\mathcal{A}$ once and never rewinds it: it relays the queries of $\mathcal{A}$ to the hybrid execution or to $\mathcal{S}_{\mathsf{SR}}$, responds to some of them from M or with (HELLO, ρ), $(\text{GOODBYE}, \rho)$, and performs deterministic computation. The oracle of $\mathcal{S}_{\mathsf{SR}}$ is implemented by the running execution of $\mathcal{A}$,

which is possible only since $\mathcal{S}_{\mathsf{SR}}$ is straight-line in its oracle. Moreover, $\mathcal{S}_{\mathsf{SA}}$ responds to the interactions of $\mathcal{Z}$ with $\mathcal{G}$, and to the interactions of $\mathcal{A}$ with $\mathcal{G}$ prior to T_0 and after T, via the environment interface of $\mathcal{S}_{\mathsf{SR}}$. Since $\mathcal{S}_{\mathsf{SA}}$ can rewind neither $\mathcal{Z}$ nor $\mathcal{A}$, this requires $\mathcal{S}_{\mathsf{SR}}$ to respond to the environment in a straight-line manner (Remark 3.12). Hence, $\mathcal{S}_{\mathsf{SA}}$ is black-box and straight-line in $\mathcal{A}$. It is PPT since $\mathcal{S}_{\mathsf{SR}}$ is PPT (as $\mathcal{A}_{\mathsf{SA}}^{\mathcal{A}}$ is PPT), where in Step 6, we assume as usual that f is computable in polynomial time. Finally, by Step 1, the trusted party for f has received the input of every honest party by T_0, and hence the trusted party always responds to the query of $\mathcal{S}_{\mathsf{SR}}$ in Step 3(e).

The environment. Fix an environment $\mathcal{Z}$. The environment $\mathcal{Z}_\rho^{\mathcal{Z}}$ for ρ is given in Figure 8. It runs the execution of $\pi^{\widehat{\rho}}$ prior to T_0 as part of its input selection, and the execution after T as part of its post-execution.

Algorithm $\mathcal{Z}_\rho^{\mathcal{Z}}$

Input selection:

1. Invoke $\mathcal{Z}$, forwarding every interaction of $\mathcal{Z}$ with $\mathcal{G}$ to the adversarial interface of $\mathcal{G}$, until $\mathcal{Z}$ outputs $(\boldsymbol{x}, z)$.
2. Sample random coins for $\mathcal{A}$ and for the honest parties of π, and internally emulate the execution of $\pi^{\widehat{\rho}}$ on $\boldsymbol{x}$ with $\mathcal{A}$ with auxiliary input z, until T_0 or until $\mathcal{A}$ halts. Here, every honest party sends (HELLO, ρ) upon invoking $\widehat{\rho}$, the messages of ρ that $\mathcal{A}$ sends on behalf of the corrupted parties are recorded, and every interaction of $\mathcal{A}$ with $\mathcal{G}$ is forwarded to the adversarial interface of $\mathcal{G}$.
3. If T_0 occurs, output the inputs $\boldsymbol{x}_\rho$ of the honest parties to ρ, the input x^{def} for every corrupted party, and the auxiliary input σ_0, the record of the emulation with $\mathsf{halt} = 0$. Otherwise, output the input x^{def} for every party, and the record σ_0 with $\mathsf{halt} = 1$.

Post-execution: Upon receiving the outputs $(y_j)_{j \notin I}$ of the honest parties and the output (σ, M) of the adversary:

4. **The procedure** Φ**.** If σ indicates that $\mathcal{A}$ halted (before T_0 or before T), let $\boldsymbol{v}$ be the outputs of the honest parties in π and the output of $\mathcal{A}$, as determined by σ. Otherwise, resume the emulation of $\mathcal{A}$ and of the honest parties of π by replaying σ, without interacting with $\mathcal{G}$, and continue it after T until $\mathcal{A}$ halts, as follows:
   (a) Respond to every request of $\mathcal{A}$ for a message of ρ from an honest party to a corrupted party from M.
   (b) Respond to every request of $\mathcal{A}$ for the round-$(R+1)$ message of an honest party with $(\text{GOODBYE}, \rho)$, and record the round-$(R+1)$ messages of the corrupted parties and the deliveries of the round-$(R+1)$ messages between honest parties. Once an honest party P_j has received a round-$(R+1)$ message from all n parties, hand y_j to the emulated P_j as its output from $\widehat{\rho}$.
   (c) Forward every interaction of $\mathcal{A}$ with $\mathcal{G}$ to the adversarial interface of $\mathcal{G}$.

   Let $\boldsymbol{v}$ be the outputs of the honest parties in π and the output of $\mathcal{A}$.
5. Hand $\boldsymbol{v}$ to $\mathcal{Z}$ as the output vector, forward every subsequent interaction of $\mathcal{Z}$ with $\mathcal{G}$ to the adversarial interface of $\mathcal{G}$, and output the bit output by $\mathcal{Z}$.

Figure 8: The environment $\mathcal{Z}_\rho^{\mathcal{Z}}$ for ρ, given oracle access to the environment $\mathcal{Z}$ for $\pi^{\widehat{\rho}}$, the real-model adversary $\mathcal{A}$ for $\pi^{\widehat{\rho}}$, and the adversarial interface of $\mathcal{G}$.

By Item 2 of Definition 3.13 and Item 2 of Claim 5.3, the emulated honest parties never interact with $\mathcal{G}$, neither in π nor in $\widehat{\rho}$ prior to T_0. This is crucial, since $\mathcal{Z}_\rho^{\mathcal{Z}}$ has access only to the adversarial interface of $\mathcal{G}$. By Item 2 of Claim 5.3, $\mathcal{A}$ requests no message of ρ from an honest party in Step 2, and hence $\mathcal{Z}_\rho^{\mathcal{Z}}$ need not emulate the honest parties in ρ. $\mathcal{Z}_\rho^{\mathcal{Z}}$ is PPT whenever $\mathcal{Z}$ is, since $\mathcal{A}$ and π are PPT, and the record σ_0 has polynomial size.

The real execution. We claim that

$$\Pr\left[\mathsf{REAL}^{\mathcal{G}}_{\rho, I, \mathcal{A}_{\mathsf{SA}}^{\mathcal{A}}, \mathcal{Z}_\rho^{\mathcal{Z}}}(\lambda) = 1\right] = \Pr\left[\mathsf{REAL}^{\mathcal{G}}_{\pi^{\widehat{\rho}}, I, \mathcal{A}, \mathcal{Z}}(\lambda) = 1\right] . \tag{1}$$

Observe that in $\pi^{\widehat{\rho}}$, the coins of $\mathcal{G}$ are independent of all other coins (Item 1 of Definition 3.13), as are the coins of the honest parties in $\widehat{\rho}$, exactly as the coins of $\mathcal{G}$ and of the honest parties in the standalone execution of ρ. We show that, except for the interactions of the honest parties with $\mathcal{G}$ caused by the completion, every message, every interaction with $\mathcal{G}$, and the output vector handed to $\mathcal{Z}$ are identical in the two experiments.

Prior to T_0. In both experiments, $\mathcal{Z}$ first interacts with the same $\mathcal{G}$ via the adversarial interface and outputs the same $(\boldsymbol{x}, z)$. In $\pi^{\widehat{\rho}}$, the execution prior to T_0 involves only $\mathcal{A}$, the honest parties of π and the round-0 messages, together with the messages of ρ that $\mathcal{A}$ sends on behalf of the corrupted parties and the interactions of $\mathcal{A}$ with $\mathcal{G}$ via the adversarial interface (by Item 2 of Definition 3.13 and Item 2 of Claim 5.3). This is exactly what $\mathcal{Z}_\rho^{\mathcal{Z}}$ emulates in Step 2, where the interactions of $\mathcal{A}$ with $\mathcal{G}$ are with the same instance via the same interface. Hence, the record σ_0 is the record of the execution of $\pi^{\widehat{\rho}}$ until T_0, and $\boldsymbol{x}_\rho$ are the inputs of the honest parties to ρ in $\pi^{\widehat{\rho}}$, by Item 1 of Claim 5.3.

From T_0 until T. By Step 2 of $\mathcal{A}_{\mathsf{SA}}^{\mathcal{A}}$, which uses the recorded responses of $\mathcal{G}$ and does not interact with $\mathcal{G}$, $\mathcal{A}$ and the emulated honest parties of π are in the same state at T_0 as in $\pi^{\widehat{\rho}}$. By Step 3, the external execution receives the messages of ρ that $\mathcal{A}$ sent prior to T_0, in the same order. In $\pi^{\widehat{\rho}}$, these messages are processed by the honest parties only after completing round 0, and hence it makes no difference that they are sent at the beginning of the external execution. From T_0 until T, the external execution of ρ runs on the inputs $\boldsymbol{x}_\rho$ with the same coins, and every message of ρ sent by $\mathcal{A}$, every request of $\mathcal{A}$, and every delivery scheduled by $\mathcal{A}$, including the delivery of the messages between the honest parties and $\mathcal{G}$, is applied to the external execution exactly when it is applied in $\pi^{\widehat{\rho}}$. Hence, the honest parties of ρ send the same messages, and interact with $\mathcal{G}$ at the same points, as the honest parties in $\pi^{\widehat{\rho}}$. The interactions of $\mathcal{A}$ with $\mathcal{G}$ are with the same instance, via the same interface. The round-0 and round-$(R+1)$ messages of an honest party are (HELLO, ρ) and $(\text{GOODBYE}, \rho)$, respectively, where $\mathcal{A}$ requests the latter only once the honest party has completed the R rounds of ρ (as $\mathcal{A}$ is well behaved). Hence, $\mathcal{A}_{\mathsf{SA}}^{\mathcal{A}}$ responds to them exactly as the honest party does. By Item 4 of Claim 5.4, prior to T, the honest parties of π behave exactly as their emulation by $\mathcal{A}_{\mathsf{SA}}^{\mathcal{A}}$. Consequently, the views of $\mathcal{A}$ are identical in the two experiments until T, or until $\mathcal{A}$ halts.

We now consider three cases based on when $\mathcal{A}$ halts: after T, before T but after T_0, and before T_0.

Case 1: T occurs. By Item 1 of Claim 5.4, at T, no message of rounds $1, \ldots, R$ from a corrupted party to an honest party and no message between honest parties or between an honest party and $\mathcal{G}$ is pending. Hence, the completion only requests the messages of the honest parties to the corrupted parties that $\mathcal{A}$ has not yet requested, and in particular, causes no interaction with $\mathcal{G}$. By Item 2 of Claim 5.4, these messages are determined at T, and hence M contains exactly the messages that $\mathcal{A}$ obtains upon requesting them in $\pi^{\widehat{\rho}}$, irrespective of the order of the requests. By Item 3 of Claim 5.4, the outputs $(y_j)_{j \notin I}$ of the honest parties in the external execution are exactly their outputs from ρ in $\pi^{\widehat{\rho}}$. After T, Φ continues the execution of $\pi^{\widehat{\rho}}$ exactly as it continues: it responds to the requests from M, handles the round-$(R+1)$ messages as the honest parties do, and hands y_j to P_j exactly when P_j obtains its output from $\widehat{\rho}$. By Item 1 of Claim 5.4, no honest party interacts with $\mathcal{G}$ after T in either experiment, and the interactions of $\mathcal{A}$ with $\mathcal{G}$ after T are with the same instance, via the same interface. Any further message of rounds $1, \ldots, R$ that $\mathcal{A}$ may send is ignored by the honest parties, who have completed these rounds. Hence, the output vector $\boldsymbol{v}$ handed to $\mathcal{Z}$ is identical in both experiments, and so are the subsequent interactions of $\mathcal{Z}$ with $\mathcal{G}$, and its output.

Case 2: T_0 occurs and $\mathcal{A}$ halts before T. The execution of $\pi^{\widehat{\rho}}$ ends when $\mathcal{A}$ halts, and no honest party obtains its output from $\widehat{\rho}$. Correspondingly, Φ ignores $(y_j)_{j \notin I}$ and M, and the output vector $\boldsymbol{v}$ is determined by σ, and is identical in both experiments. The only difference is that in $\mathsf{REAL}^{\mathcal{G}}_{\rho, I, \mathcal{A}_{\mathsf{SA}}^{\mathcal{A}}, \mathcal{Z}_\rho^{\mathcal{Z}}}$, the completion of the external execution causes interactions of the honest parties with $\mathcal{G}$ that do not occur in $\pi^{\widehat{\rho}}$, where these parties are stalled. The only subsequent interactions with $\mathcal{G}$ are those of $\mathcal{Z}$, via the adversarial interface. Since $\mathcal{G}$ is stateless (Definition 3.14), the responses to these interactions are identically distributed in both experiments, given the transcript of the execution until $\mathcal{A}$ halts.

Case 3: $\mathcal{A}$ halts before T_0. The execution of $\pi^{\widehat{\rho}}$ ends prior to T_0, and no honest party interacts with $\mathcal{G}$ (Item 2 of Claim 5.3). In $\mathsf{REAL}^{\mathcal{G}}_{\rho, I, \mathcal{A}_{\mathsf{SA}}^{\mathcal{A}}, \mathcal{Z}_\rho^{\mathcal{Z}}}$, the external execution of ρ runs on the default inputs

and is completed by $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$, and Φ outputs the output vector $\boldsymbol{v}$ determined by σ_0, which is identical to the output vector in $\pi^{\widehat{\rho}}$. As in Case 2, the interactions of the honest parties with $\mathcal{G}$ in the external execution do not change the distribution of the responses to the subsequent interactions of $\mathcal{Z}$, since $\mathcal{G}$ is stateless (Definition 3.14).

In all cases, the output of $\mathcal{Z}$ is identically distributed in both experiments, which proves Equation (1). Note that statelessness is used only in Cases 2 and 3.

The hybrid execution. We claim that

$$\Pr\left[\mathsf{IDEAL}^{\mathsf{SA}}_{f,I,\mathcal{S}_{\mathsf{SR}},\mathcal{Z}^{\mathcal{Z}}_{\rho}}(\lambda)=1\right]=\Pr\left[\mathsf{HYBRID}^{f,\mathsf{SA}}_{\pi,I,\mathcal{S}_{\mathsf{SA}},\mathcal{Z}}(\lambda)=1\right]. \tag{2}$$

For the sake of syntax, suppose that the two experiments use the same random coins for $\mathcal{Z}$, for $\mathcal{A}$, for the honest parties in π, and for $\mathcal{S}_{\mathsf{SR}}$.

Prior to T_0. In both experiments, the environment interface of $\mathcal{S}_{\mathsf{SR}}$ in its pre-execution phase responds to every interaction of $\mathcal{Z}$ with $\mathcal{G}$ prior to choosing the inputs, and $\mathcal{Z}$ outputs the same $(\boldsymbol{x},z)$. In the ideal experiment, $\mathcal{Z}^{\mathcal{Z}}_{\rho}$ then emulates $\mathcal{A}$ and the honest parties of π until T_0, and forwards every interaction of $\mathcal{A}$ with $\mathcal{G}$ to its adversarial interface, to which the environment interface of $\mathcal{S}_{\mathsf{SR}}$ responds, still in its pre-execution phase. In the hybrid experiment, $\mathcal{A}$ interacts with the honest parties of π in the hybrid execution, the environment interface of $\mathcal{S}_{\mathsf{SR}}$ responds to every interaction of $\mathcal{A}$ with $\mathcal{G}$ (Step 2), the round-0 message of an honest party is sent exactly when it invokes $\widehat{\rho}$ in the emulation, namely, when it sends its ideal message (Step 1), and no honest party obtains an output from f. Hence, the executions until T_0 are identical. Moreover, the trusted party for f in the hybrid experiment receives the ideal messages $\boldsymbol{x}_{\rho}$ of the honest parties by T_0, which are the inputs that $\mathcal{Z}^{\mathcal{Z}}_{\rho}$ outputs for the honest parties in the ideal experiment.

From T_0 until T. In the ideal experiment, $\mathcal{S}_{\mathsf{SR}}$ starts its execution phase with the inputs x^{def} of the corrupted parties and the auxiliary input σ_0, with oracle access to $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$, which first replays σ_0 and then sends the messages of ρ that $\mathcal{A}$ sent prior to T_0. In the hybrid experiment, $\mathcal{S}_{\mathsf{SR}}$ starts its execution phase at T_0 with the same inputs of the corrupted parties, and the first queries of its oracle are the same messages (Step 3(a)), while the running execution of $\mathcal{A}$ is in exactly the state that $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$ reaches by replaying σ_0. Since $\mathcal{S}_{\mathsf{SR}}$ accesses the auxiliary input only through its oracle, and never rewinds its oracle, it makes no difference that $\mathcal{S}_{\mathsf{SA}}$ never computes σ_0. From T_0 until T, $\mathcal{S}_{\mathsf{SA}}$ forwards between $\mathcal{A}$ and $\mathcal{S}_{\mathsf{SR}}$ exactly the queries that $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$ forwards between $\mathcal{A}$ and its external execution, including the interactions of $\mathcal{A}$ with $\mathcal{G}$, which are forwarded to the oracle interface of $\mathcal{S}_{\mathsf{SR}}$ in both experiments (Steps 3(b)–3(c) of $\mathcal{S}_{\mathsf{SA}}$ and Steps 4(b)–4(c) of $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$), and handles the messages of rounds 0 and $R+1$ exactly as $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$ does. By Item 4 of Claim 5.4, and since $\mathcal{S}_{\mathsf{SA}}$ delivers no output from f prior to T, the honest parties of π in the hybrid execution behave exactly as their emulation by $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$. The query of $\mathcal{S}_{\mathsf{SR}}$ to the trusted party (Step 3(e)), the outputs of the corrupted parties, and the message $(\mathtt{abort},J_{\mathsf{SR}})$ are identical in both experiments, since in both, the trusted party holds the inputs $\boldsymbol{x}_{\rho}$ of the honest parties (Step 1). Hence, the outputs $(y_j)_{j\notin I}$ of the honest parties from the trusted party are identical in both experiments, where $y_j=\perp$ for every $j\in J_{\mathsf{SR}}$.

Cases 1 and 2: T_0 occurs. At T, or when $\mathcal{A}$ halts after T_0 and before T, $\mathcal{S}_{\mathsf{SA}}$ completes the execution of ρ simulated by $\mathcal{S}_{\mathsf{SR}}$ exactly as $\mathcal{A}^{\mathcal{A}}_{\mathsf{SA}}$ completes its external execution, and hence the buffer M is identical in both experiments, and so is the remaining behaviour of $\mathcal{S}_{\mathsf{SR}}$ in its execution phase. In the ideal experiment, the output of $\mathcal{S}_{\mathsf{SR}}$ is the output (σ,M) of its oracle, which is handed to $\mathcal{Z}^{\mathcal{Z}}_{\rho}$ together with $(y_j)_{j\notin I}$, and Φ resumes the execution of $\mathcal{A}$ from σ. In the hybrid experiment, $\mathcal{S}_{\mathsf{SA}}$ continues the running execution of $\mathcal{A}$, which is in the state recorded by σ, and behaves exactly as Φ: it responds to the requests from M, handles the round-$(R+1)$ messages identically, and delivers the output y_j of P_j from the trusted party exactly when Φ hands y_j to the emulated P_j (Steps 5(a)–5(b)). In both experiments, the environment interface of $\mathcal{S}_{\mathsf{SR}}$ in its post-execution phase responds to the interactions of $\mathcal{A}$ with $\mathcal{G}$ after T (Step 5(c), and Step 4(c) of Φ in the ideal experiment). If $\mathcal{A}$ halts before T, then in both experiments no honest party obtains its output from f, and the output vector is determined by the execution until $\mathcal{A}$ halts. Hence, the output vector handed to $\mathcal{Z}$ is identical in both experiments, and in both, the environment interface of $\mathcal{S}_{\mathsf{SR}}$ responds to the subsequent interactions of $\mathcal{Z}$ with $\mathcal{G}$.

Case 3: $\mathcal{A}$ halts before T_0. If $\mathcal{A}$ halts before T_0, then in the ideal experiment, $\mathcal{Z}_\rho^{\mathcal{Z}}$ outputs the input x^{def} for every party and σ_0 with $\mathsf{halt} = 1$, and $\mathcal{S}_{\mathsf{SR}}$ runs its execution phase with the oracle $\mathcal{A}_{\mathsf{SA}}^{\mathcal{A}}$, which only completes the execution, with the trusted party computing f on the default inputs of the honest parties. In the hybrid experiment, $\mathcal{S}_{\mathsf{SA}}$ runs the execution phase of $\mathcal{S}_{\mathsf{SR}}$ in exactly the same manner, computing f itself on the same inputs (Step 6). Hence, the subsequent behaviour of $\mathcal{S}_{\mathsf{SR}}$, and in particular, its responses to the interactions of $\mathcal{Z}$ with $\mathcal{G}$, are identically distributed in both experiments. In both, no honest party obtains an output from f, and the output vector is determined by the execution until $\mathcal{A}$ halts.

In all cases, the output of $\mathcal{Z}$ is identically distributed in both experiments, which proves Equation (2).

Conclusion. $\mathcal{A}_{\mathsf{SA}}^{\mathcal{A}}$ is a PPT super-rushing adversary for ρ, and $\mathcal{Z}_\rho^{\mathcal{Z}}$ is a PPT environment. By the security of ρ with adaptive inputs (Definition 3.11), with the simulator $\mathcal{S}_{\mathsf{SR}} = \mathcal{S}^{\mathcal{A}_{\mathsf{SA}}^{\mathcal{A}}}$ and the environment $\mathcal{Z}_\rho^{\mathcal{Z}}$,

$$\left|\Pr\left[\mathsf{REAL}^{\mathcal{G}}_{\rho,I,\mathcal{A}_{\mathsf{SA}}^{\mathcal{A}},\mathcal{Z}_\rho^{\mathcal{Z}}}(\lambda)=1\right]-\Pr\left[\mathsf{IDEAL}^{\mathsf{SA}}_{f,I,\mathcal{S}_{\mathsf{SR}},\mathcal{Z}_\rho^{\mathcal{Z}}}(\lambda)=1\right]\right|\leq \mathsf{negl}(\lambda)\ .$$

Together with Equations (1) and (2),

$$\left|\Pr\left[\mathsf{HYBRID}^{f,\mathsf{SA}}_{\pi,I,\mathcal{S}_{\mathsf{SA}},\mathcal{Z}}(\lambda)=1\right]-\Pr\left[\mathsf{REAL}^{\mathcal{G}}_{\pi^{\widehat{\rho}},I,\mathcal{A},\mathcal{Z}}(\lambda)=1\right]\right|\leq \mathsf{negl}(\lambda)\ ,$$

for every PPT environment $\mathcal{Z}$, as required. Note that adaptive inputs are required, since $\mathcal{Z}_\rho^{\mathcal{Z}}$ interacts with $\mathcal{G}$, both directly via $\mathcal{Z}$ and on behalf of $\mathcal{A}$, prior to choosing the inputs $\boldsymbol{x}_\rho$, which may depend on these interactions.

Statistical security. For statistical security, the same argument holds for every (not necessarily PPT) $\mathcal{A}$ and $\mathcal{Z}$, where $\mathcal{A}_{\mathsf{SA}}^{\mathcal{A}}$, $\mathcal{Z}_\rho^{\mathcal{Z}}$, $\mathcal{S}_{\mathsf{SR}}$ and $\mathcal{S}_{\mathsf{SA}}$ need not run in polynomial time, and Equations (1) and (2) hold as is.

On private channels. If the channels between honest parties are ideally private, $\mathcal{A}$ observes only the metadata of the messages between honest parties. The reduction and the simulator relay to $\mathcal{A}$ exactly what the external, resp. simulated, execution of ρ reveals about such messages, and the above argument is unchanged. □

Theorem 5.5 (Unanimous Abort with Setup)**.** *Let f be an n-party functionality, let $\mathcal{G}$ be an ideal functionality, and let ρ be an R-round synchronous protocol in the $\mathcal{G}$-hybrid model such that the following holds:*

1. *ρ t-securely computes f against super-rushing adversaries with unanimous abort, adaptive inputs, and computational (resp. statistical) security (as per Definition 3.11); and*
2. *The underlying simulator for ρ's security is black-box and straight-line (as per Remark 3.12); and*
3. *The setup $\mathcal{G}$ is localised (as per Definition 3.13) and stateless (as per Definition 3.14).*

Let $\widehat{\rho}$ be the $(R+2)$-round protocol obtained by adding start synchronisation and broadcast end synchronisation to ρ. Then for every n-party protocol π in the f-hybrid model that makes a single ideal call to f, every PPT real-model adversary $\mathcal{A}$ for $\pi^{\widehat{\rho}}$, and every $I \subset [n]$ with $|I| \leq t$, there exists a PPT hybrid-model adversary $\mathcal{S}_{\mathsf{UA}}$, black-box and straight-line in $\mathcal{A}$, such that for every PPT environment $\mathcal{Z}$, it holds that

$$\left|\Pr\left[\mathsf{HYBRID}^{f,\mathsf{UA}}_{\pi,I,\mathcal{S}_{\mathsf{UA}},\mathcal{Z}}(\lambda)=1\right]-\Pr\left[\mathsf{REAL}^{\mathcal{G}}_{\pi^{\widehat{\rho}},I,\mathcal{A},\mathcal{Z}}(\lambda)=1\right]\right|\leq \mathsf{negl}(\lambda)\ ,$$

and respectively, the same holds for every (not necessarily PPT) adversary $\mathcal{A}$ and environment $\mathcal{Z}$, with a hybrid-model adversary $\mathcal{S}_{\mathsf{UA}}$ that need not run in polynomial time, for statistical security.

Proof. The proof follows the proof of Theorem 5.2, and we describe the differences. Recall the consistency of the atomic broadcast channel (Section 3.4): a broadcast message that has been delivered to some honest party is eventually delivered to every honest party, even if the adversary halts in the meantime, whereas a broadcast message that has been delivered to no honest party by the time the adversary halts is never delivered. In particular, the convention that an honest party never obtains its output from $\widehat{\rho}$ after the adversary halts no longer holds, and we adapt the instant T, the procedure Φ and the simulator accordingly.

The instant T. Redefine T as the first instant at which the round-$(R+1)$ broadcast of every party $k \in [n]$ has been delivered to at least one honest party.

Claim 5.6. *With broadcast end synchronisation and the above T, the following hold in every execution of $\pi^{\widehat{\rho}}$.*

1. *Claim 5.4 holds as is.*
2. *If T occurs, then every honest party eventually receives the round-$(R+1)$ broadcasts of all n parties, and hence obtains its output from $\widehat{\rho}$, even if $\mathcal{A}$ halts right after T. If $\mathcal{A}$ halts before T, then no honest party obtains its output from $\widehat{\rho}$.*

Proof. For Item 1, an honest party broadcasts its round-$(R+1)$ message only upon completing the R rounds of ρ. Hence, at T, every honest party has completed the R rounds of ρ, which implies Items 1–3 of Claim 5.4, exactly as in its proof. An honest party obtains its output from $\widehat{\rho}$ only upon receiving a round-$(R+1)$ broadcast from all n parties, which requires T to have occurred, which implies Item 4 of Claim 5.4. For Item 2, if T occurs, then each of the n round-$(R+1)$ broadcasts has been delivered to some honest party, and by the consistency of the broadcast channel, it is eventually delivered to every honest party. If $\mathcal{A}$ halts before T, then the round-$(R+1)$ broadcast of some party has been delivered to no honest party, and by the consistency of the broadcast channel, it is never delivered. □

Hence, in $\pi^{\widehat{\rho}}$, either every honest party obtains its output from $\widehat{\rho}$, or none does.

The reduction, the environment and the simulator. Let $\mathcal{A}_{\mathsf{UA}}^{\mathcal{A}}$ be identical to $\mathcal{A}_{\mathsf{SA}}^{\mathcal{A}}$ (Figure 6) with the above T, where the round-$(R+1)$ messages are now broadcast, and are still handled internally by recording their deliveries. Let $\mathcal{Z}_{\rho}^{\mathcal{Z}}$ be as above (Figure 8) with the above T, where Φ additionally proceeds as follows when $\mathcal{A}$ halts after T: it delivers every round-$(R+1)$ broadcast to every emulated honest party that has not yet received it, as the broadcast channel does, and hands y_j to every emulated honest party P_j that has not yet obtained its output from $\widehat{\rho}$, after which the emulated honest parties of π perform whatever local computation π prescribes. Let $\mathcal{S}_{\mathsf{SR}} \stackrel{def}{=} \mathcal{S}^{\mathcal{A}_{\mathsf{UA}}^{\mathcal{A}}}$ be the black-box straight-line simulator for ρ with unanimous abort and adaptive inputs. Let $\mathcal{S}_{\mathsf{UA}}$ be identical to $\mathcal{S}_{\mathsf{SA}}$ (Figure 7) with the above T, except for the following.

- In Step 3(e), $\mathcal{S}_{\mathsf{UA}}$ records the message $(\mathtt{abort}, J_{\mathsf{SR}})$ of $\mathcal{S}_{\mathsf{SR}}$, where $J_{\mathsf{SR}} \in \{\emptyset, [n] \setminus I\}$, and withholds its response to the trusted party. At T, after the completion, $\mathcal{S}_{\mathsf{UA}}$ forwards $(\mathtt{abort}, J_{\mathsf{SR}})$ to the trusted party. If $\mathcal{A}$ halts after T_0 and before T, $\mathcal{S}_{\mathsf{UA}}$ sends $(\mathtt{abort}, [n] \setminus I)$ to the trusted party and delivers no output.
- In Step 5(b), when $\mathcal{A}$ halts after T, $\mathcal{S}_{\mathsf{UA}}$ delivers the output from the trusted party to every honest party that has not yet received it, before outputting the output of $\mathcal{A}$.

Consequently, if T occurs, then every honest party obtains its output from f, which is y_j for every honest party if $J_{\mathsf{SR}} = \emptyset$, and $\perp$ for every honest party otherwise; and if T does not occur, then no honest party obtains an output from f. With point-to-point end synchronisation, the adversary may withhold the round-$(R+1)$ messages from some of the honest parties after T, and this unanimity fails.

The argument. $\mathcal{A}_{\mathsf{UA}}^{\mathcal{A}}$ is a valid PPT super-rushing adversary for ρ, $\mathcal{Z}_{\rho}^{\mathcal{Z}}$ is PPT, and $\mathcal{S}_{\mathsf{UA}}$ is PPT and black-box straight-line in $\mathcal{A}$, exactly as before. Equation (1) holds with $\mathcal{A}_{\mathsf{UA}}^{\mathcal{A}}$ in place of $\mathcal{A}_{\mathsf{SA}}^{\mathcal{A}}$, by the same case analysis with the above T. In Case 1, when $\mathcal{A}$ halts after T, by Item 2 of Claim 5.6, the honest parties that have not yet obtained their outputs from $\widehat{\rho}$ in $\pi^{\widehat{\rho}}$ obtain them after $\mathcal{A}$ halts, and these outputs are $(y_j)_{j \notin I}$ by Item 3 of Claim 5.4; this is exactly what Φ does upon $\mathcal{A}$ halting. In Case 2, by Item 2 of Claim 5.6, no honest party obtains its output from $\widehat{\rho}$, and Φ ignores $(y_j)_{j \notin I}$, exactly as before. Equation (2) holds with $\mathsf{IDEAL}_{f,I,\mathcal{S}_{\mathsf{SR}},\mathcal{Z}_{\rho}^{\mathcal{Z}}}^{\mathsf{UA}}$ and $\mathsf{HYBRID}_{\pi,I,\mathcal{S}_{\mathsf{UA}},\mathcal{Z}}^{f,\mathsf{UA}}$, by the same argument. If T occurs, the trusted party receives the same message $(\mathtt{abort}, J_{\mathsf{SR}})$ in both experiments, hence holds the same outputs $(y_j)_{j \notin I}$, and $\mathcal{S}_{\mathsf{UA}}$ delivers them exactly when Φ hands them to the emulated honest parties, including upon $\mathcal{A}$ halting after T. If $\mathcal{A}$ halts after T_0 and before T, the trusted party in the hybrid experiment receives $(\mathtt{abort}, [n] \setminus I)$, possibly while $J_{\mathsf{SR}} = \emptyset$; however, in this case Φ ignores the outputs of the honest parties, and $\mathcal{S}_{\mathsf{UA}}$ delivers

no output, so the output vector handed to $\mathcal{Z}$ is identical in both experiments. If $\mathcal{A}$ halts before T_0, the argument is unchanged. The conclusion follows as above, including the statistical case, by the security of ρ with unanimous abort and adaptive inputs. □

5.3 A Separation without Adaptive Inputs

We show that security with adaptive inputs in Theorem 5.2 cannot be relaxed to the usual notion of security in the $\mathcal{G}$-hybrid model, where the environment does not interact with $\mathcal{G}$ prior to choosing the inputs. The separation holds even though the start synchronisation fixes the inputs of the honest parties before any message of ρ is sent, and even though the setup is localised and stateless. The reason is that the adversary may interact with the setup prior to T_0, and influence the inputs of the honest parties to ρ via the calling protocol, as a function of the responses of the setup. Hence, in the composed execution, the inputs of the honest parties to ρ may depend on the setup, even if the environment chooses the inputs to π independently of it.

The setup. We say that $\mathcal{G}$ is an *unpredictable setup* if, upon its first request, it samples a value u from a distribution with min-entropy at least λ, and upon every request, from any party or from the adversary, it returns u. For example, $u = \mathsf{crs}$ for $\mathcal{F}_{\mathsf{crs}}$, $u = H(0)$ for $\mathcal{F}_{\mathsf{RO}}$, or the public key of P_2 for $\mathcal{F}_{\mathsf{PKI}}$. The copies of an unpredictable setup are localised (Definition 3.13) whenever each copy samples its own value u, and every copy is stateless (Definition 3.14), since its responses do not depend on whether, when or how the honest parties interact with it.

The functionality and the protocol. Consider $n = 2$ parties, the functionality f_{guess} (Figure 9), and the protocol ρ_{na} (Figure 10) in the $\mathcal{G}$-hybrid model. In ρ_{na}, the party P_2 reveals its secret s to P_1 if and only if its input v equals the value u of the setup. Intuitively, ρ_{na} is secure as long as the inputs are chosen independently of the setup, since then $v = u$ occurs with negligible probability.

Functionality f_{guess}

Inputs: P_1 has no input, and P_2 holds an input (v, s).

Outputs: P_2 outputs s, and P_1 outputs the empty string λ.

Figure 9: The two-party functionality f_{guess}.

Protocol ρ_{na}

Inputs: P_1 has no input, and P_2 holds (v, s).

Round 1:

1. P_2 requests the value u from $\mathcal{G}$.
2. If $v = u$, then P_2 sends s to P_1. Otherwise, P_2 sends 0 to P_1.
3. P_1 sends the dummy message 0 to P_2.

Output: P_2 outputs s, and P_1 outputs λ.

Figure 10: The protocol ρ_{na} for f_{guess} in the $\mathcal{G}$-hybrid model.

The calling protocol. The calling protocol π_{sep} (Figure 11) lets the corrupted party choose the value v that P_2 inputs to f_{guess}. It makes a single ideal call to f_{guess}.

Protocol π_{sep}

Inputs: P_1 has no input, and P_2 holds an input $s \in \{0,1\}^\lambda$.

Execution:

1. P_1 sends a value v to P_2.
2. Upon receiving v, the party P_2 invokes f_{guess} with input (v, s).

Output: P_2 outputs its output from f_{guess}, and P_1 outputs λ.

Figure 11: The protocol π_{sep} in the f_{guess}-hybrid model.

Theorem 5.7 (Separation without Adaptive Inputs)**.** *Let $\mathcal{G}$ be an unpredictable setup, and let $\widehat{\rho}_{\mathsf{na}}$ be the protocol obtained by adding start synchronisation and point-to-point end synchronisation to ρ_{na}. Then the following hold:*

1. *ρ_{na} 1-securely computes f_{guess} against super-rushing adversaries with guaranteed output delivery, statistical security and black-box straight-line simulation, under the usual notion of security in the $\mathcal{G}$-hybrid model, i.e., Definition 3.11 restricted to environments that do not interact with $\mathcal{G}$ prior to choosing the inputs; and*
2. *$\widehat{\rho}_{\mathsf{na}}$ does not 1-securely compute f_{guess} under 1-bounded concurrent general composition (Remark 3.8) with selective abort, neither with statistical nor with computational security. Specifically, there exists a PPT real-model adversary $\mathcal{A}_{\mathsf{sep}}$ for $\pi_{\mathsf{sep}}^{\widehat{\rho}_{\mathsf{na}}}$ corrupting P_1, and a PPT environment $\mathcal{Z}_{\mathsf{sep}}$ that does not interact with $\mathcal{G}$, such that for every (not necessarily PPT) hybrid-model adversary $\mathcal{S}_{\mathsf{SA}}$ corrupting P_1,*

$$\Pr\left[\mathsf{REAL}^{\mathcal{G}}_{\pi_{\mathsf{sep}}^{\widehat{\rho}_{\mathsf{na}}},\{1\},\mathcal{A}_{\mathsf{sep}},\mathcal{Z}_{\mathsf{sep}}}(\lambda)=1\right] - \Pr\left[\mathsf{HYBRID}^{f_{\mathsf{guess}},\mathsf{SA}}_{\pi_{\mathsf{sep}},\{1\},\mathcal{S}_{\mathsf{SA}},\mathcal{Z}_{\mathsf{sep}}}(\lambda)=1\right] \geq 1-2^{-\lambda}\ .$$

Proof. We show each item separately.

Standalone security. Fix an environment $\mathcal{Z}$ that does not interact with $\mathcal{G}$ prior to choosing the inputs, and a super-rushing adversary $\mathcal{A}$ corrupting a subset $I \subset [2]$ with $|I| \leq 1$. The simulator $\mathcal{S}^{\mathcal{A}}$ plays the role of $\mathcal{G}$ honestly towards $\mathcal{A}$ and $\mathcal{Z}$: it samples u as $\mathcal{G}$ does, and returns u upon every request, both of $\mathcal{A}$ and of $\mathcal{Z}$, before, during and after the execution. It sends $(\mathtt{abort}, \emptyset)$ to the trusted party, and outputs whatever $\mathcal{A}$ outputs. Its remaining behaviour depends on I.

- **P_1 is corrupted or $I = \emptyset$.** If P_1 is corrupted, $\mathcal{S}^{\mathcal{A}}$ sends the empty input on behalf of P_1 to the trusted party. It plays the role of the honest P_2 towards $\mathcal{A}$, schedules the request of P_2 to $\mathcal{G}$ and its response as $\mathcal{A}$ does, and responds to the request of $\mathcal{A}$ for the round-1 message of P_2 with 0. If $I = \emptyset$, it also plays the role of the honest P_1, whose round-1 message is 0.
- **P_2 is corrupted.** $\mathcal{S}^{\mathcal{A}}$ sends the input $(0,0)$ on behalf of P_2 to the trusted party, and responds to the request of $\mathcal{A}$ for the round-1 message of P_1 with 0.

In both cases, $\mathcal{S}^{\mathcal{A}}$ invokes $\mathcal{A}$ once and never rewinds it, and responds to the interactions of $\mathcal{Z}$ with $\mathcal{G}$ in a straight-line manner. Hence, $\mathcal{S}^{\mathcal{A}}$ is black-box and straight-line in $\mathcal{A}$ (Remark 3.12). Since $\mathcal{S}^{\mathcal{A}}$ responds to every request of $\mathcal{A}$ with a fixed message, independently of the order of the requests, it simulates super-rushing adversaries as well.

P_2 is corrupted. The only message of the honest P_1 is the dummy message 0, and its output is λ in both experiments. The responses of $\mathcal{G}$ are identically distributed in both experiments. Hence, the two experiments are identically distributed.

P_1 is corrupted or $I = \emptyset$. The output of P_2 is s in both experiments, the output of P_1 (if honest) is λ, and the responses of $\mathcal{G}$ are identically distributed. The two experiments differ only if $v = u$, in which case P_2 sends s instead of 0 in the real experiment. Since $\mathcal{Z}$ does not interact with $\mathcal{G}$ prior to choosing the inputs, and the random coins of $\mathcal{G}$ are independent of those of $\mathcal{Z}$ (Item 1 of Definition 3.13), the input v is independent of u. Since u has min-entropy at least λ, it holds that $\mathsf{Pr}[v = u] \leq 2^{-\lambda}$. Hence, the outputs of $\mathcal{Z}$ in the two experiments are $2^{-\lambda}$-close.

This proves Item 1.

Insecurity under concurrent general composition. Consider the adversary $\mathcal{A}_{\mathsf{sep}}$ (Figure 12) corrupting P_1 in an execution of $\pi_{\mathsf{sep}}^{\widehat{\rho}_{\mathsf{na}}}$, and the environment $\mathcal{Z}_{\mathsf{sep}}$ that proceeds as follows. It samples $s \leftarrow \{0,1\}^{\lambda}$ uniformly at random, outputs the input s for P_2 and the empty auxiliary input, and, upon receiving the output vector, outputs 1 if and only if the output of the adversary equals s. Note that $\mathcal{Z}_{\mathsf{sep}}$ never interacts with $\mathcal{G}$.

Adversary $\mathcal{A}_{\mathsf{sep}}$

Execution:

1. **Obtaining the setup.** Request the value u from the copy of $\mathcal{G}$ of the instance of $\widehat{\rho}_{\mathsf{na}}$.
2. **Choosing the input.** Send $v \stackrel{def}{=} u$ to P_2 as the message of P_1 in π_{sep}.
3. **Start synchronisation.** Once P_2 invokes $\widehat{\rho}_{\mathsf{na}}$, send $(\text{HELLO}, \rho_{\mathsf{na}})$ on behalf of P_1 to P_2, and request the message $(\text{HELLO}, \rho_{\mathsf{na}})$ of P_2.
4. **Round 1.** Deliver the request of P_2 to $\mathcal{G}$ and its response. Then, request the round-1 message m of P_2, and send the dummy message 0 on behalf of P_1 to P_2.
5. **End synchronisation.** Send $(\text{GOODBYE}, \rho_{\mathsf{na}})$ on behalf of P_1 to P_2, and request the message $(\text{GOODBYE}, \rho_{\mathsf{na}})$ of P_2.
6. **Output.** Output m and halt.

Figure 12: The real-model adversary $\mathcal{A}_{\mathsf{sep}}$ for $\pi_{\mathsf{sep}}^{\widehat{\rho}_{\mathsf{na}}}$ corrupting P_1.

$\mathcal{A}_{\mathsf{sep}}$ is a valid adversary. $\mathcal{A}_{\mathsf{sep}}$ requests every message of P_2 only once P_2 can compute it: the round-0 message once P_2 has invoked $\widehat{\rho}_{\mathsf{na}}$, the round-1 message once P_2 has completed round 0 and has received the response of $\mathcal{G}$, and the round-2 message once P_2 has completed round 1. Hence, $\mathcal{A}_{\mathsf{sep}}$ is well behaved. It is clearly PPT, and so is $\mathcal{Z}_{\mathsf{sep}}$.

The real execution. Since $\mathcal{G}$ returns the same value u upon every request, P_2 invokes $\widehat{\rho}_{\mathsf{na}}$ with input (u, s), and its request to $\mathcal{G}$ in round 1 returns u. Hence, P_2 sends $m = s$ in round 1 of ρ_{na}, and $\mathcal{A}_{\mathsf{sep}}$ outputs s. Therefore, $\mathcal{Z}_{\mathsf{sep}}$ outputs 1 with probability 1.

The hybrid execution. Fix a hybrid-model adversary $\mathcal{S}_{\mathsf{SA}}$ corrupting P_1 in an execution of π_{sep} in the f_{guess}-hybrid model. The only message that depends on s is the ideal message of P_2 to the trusted party, of which $\mathcal{S}_{\mathsf{SA}}$ observes only the metadata. The output of P_1 from f_{guess} is λ, and the output of P_2 is not revealed to $\mathcal{S}_{\mathsf{SA}}$. Hence, the view of $\mathcal{S}_{\mathsf{SA}}$, and in particular its output, is independent of s. Since s is uniformly distributed over $\{0,1\}^{\lambda}$, the output of $\mathcal{S}_{\mathsf{SA}}$ equals s with probability at most $2^{-\lambda}$, and therefore $\mathcal{Z}_{\mathsf{sep}}$ outputs 1 with probability at most $2^{-\lambda}$.

Conclusion. Clearly, the environment succeeds in distinguishing between the two executions with probability $\geq 1 - 2^{-\lambda}$, for every hybrid-model adversary $\mathcal{S}_{\mathsf{SA}}$, including one that does not run in polynomial time. Since $\mathcal{A}_{\mathsf{sep}}$ and $\mathcal{Z}_{\mathsf{sep}}$ are PPT, $\widehat{\rho}_{\mathsf{na}}$ is neither statistically nor computationally secure. □

Tightness of the separation. By Theorem 5.7, the protocol ρ_{na} satisfies every hypothesis of Theorem 5.2 except security with adaptive inputs: it is secure against super-rushing adversaries with guaranteed output delivery, and hence with selective abort, its simulator is black-box and straight-line, and the setup is localised and stateless. Indeed, ρ_{na} is not secure with adaptive inputs: an environment that obtains u from $\mathcal{G}$ and then chooses $v \stackrel{def}{=} u$ distinguishes the real and the ideal experiments, exactly as $\mathcal{A}_{\mathsf{sep}}$ does. In that sense, the hypothesis of security with adaptive inputs in Theorem 5.2 is tight. Note also that $\mathcal{Z}_{\mathsf{sep}}$ never interacts with $\mathcal{G}$, and the dependence of the input of P_2 to ρ_{na} on the setup is created by $\mathcal{A}_{\mathsf{sep}}$ alone, via the calling protocol.

Acknowledgements

The author would like to thank Yuval Ishai for explicitly pointing out the round-doubling of the KLR result. The author would also like to thank Rahul Satish for the many fruitful discussions that led to this work.

References

[ACCI25] Gilad Asharov, Anirudh Chandramouli, Ran Cohen, and Yuval Ishai. Peeking into the future: MPC resilient to super-rushing adversaries. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings, Part V*, volume 15605 of *Lecture Notes in Computer Science*, pages 390–420. Springer, 2025.

[BCG93] Michael Ben-Or, Ran Canetti, and Oded Goldreich. Asynchronous secure computation. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 52–61, 1993.

[BCNP04] Boaz Barak, Ran Canetti, Jesper Buus Nielsen, and Rafael Pass. Universally composable protocols with relaxed set-up assumptions. In *45th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2004*, pages 186–195. IEEE Computer Society, 2004.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of Annual ACM Symposium on Theory of Computing*, 1988.

[BKR94] Michael Ben-Or, Boaz Kelmer, and Tal Rabin. Asynchronous secure computations with optimal resilience. In *Proceedings of the 13th Annual ACM Symposium on Principles of Distributed Computing*, pages 183–192, 1994.

[BL18] Fabrice Benhamouda and Huijia Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*. Springer, 2018.

[BLZL20] Erica Blum, Chen-Da Liu-Zhang, and Julian Loss. Always have a backup plan: Fully secure synchronous MPC with asynchronous fallback. In *Advances in Cryptology – CRYPTO 2020, Part II*, volume 12171 of *Lecture Notes in Computer Science*. Springer, 2020.

[BMU07] Michael Backes, Jörn Müller-Quade, and Dominique Unruh. On the necessity of rewinding in secure multiparty computation. In *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007*, volume 4392 of *Lecture Notes in Computer Science*, pages 157–173. Springer, 2007.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BS05] Boaz Barak and Amit Sahai. How to play almost any mental game over the net – concurrent composition via super-polynomial simulation. In *46th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2005*, pages 543–552. IEEE Computer Society, 2005.

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

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *FOCS '01*, pages 136–145, 2001.

[Can07] Ran Canetti. Obtaining universally composable security: Towards the bare bones of trust. In *Advances in Cryptology – ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 88–112. Springer, 2007.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, pages 11–19. ACM, 1988.

[CCGZ21] Ran Cohen, Sandro Coretti, Juan A. Garay, and Vassilis Zikas. Round-preserving parallel composition of probabilistic-termination cryptographic protocols. *Journal of Cryptology*, 34(2):12, 2021.

[CDG+18] Jan Camenisch, Manu Drijvers, Tommaso Gagliardoni, Anja Lehmann, and Gregory Neven. The wonderful world of global random oracles. In *Advances in Cryptology – EUROCRYPT 2018, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 280–312. Springer, 2018.

[CDPW07] Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. Universally composable security with global setup. In *Theory of Cryptography Conference – TCC 2007*, volume 4392 of *Lecture Notes in Computer Science*, pages 61–85. Springer, 2007.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 19–40. Springer, 2001.

[CFG+25] Ran Cohen, Pouyan Forghani, Juan Garay, Rutvik Patel, and Vassilis Zikas. Is it even possible? on the parallel composition of asynchronous MPC protocols. Cryptology ePrint Archive, Paper 2025/1707, 2025.

[CGHZ16] Sandro Coretti, Juan A. Garay, Martin Hirt, and Vassilis Zikas. Constant-round asynchronous multi-party computation based on one-way functions. In *Advances in Cryptology – ASIACRYPT 2016, Part II*, volume 10032 of *Lecture Notes in Computer Science*. Springer, 2016.

[CGZ20] Ran Cohen, Juan A. Garay, and Vassilis Zikas. Broadcast-optimal two-round MPC. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Proceedings, Part II*, volume 12106 of *Lecture Notes in Computer Science*, pages 828–858. Springer, 2020.

[CJS14] Ran Canetti, Abhishek Jain, and Alessandra Scafuro. Practical UC security with a global random oracle. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, CCS 2014*, pages 597–608. ACM, 2014.

[CKL06] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. *Journal of Cryptology*, 19(2):135–167, 2006.

[Cle86] Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In Juris Hartmanis, editor, *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, pages 364–369. ACM, 1986.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pages 494–503, 2002.

[CRZ26] Michele Ciampi, Divya Ravi, and Mingrui Zou. Round-preserving compilers for super-rushing secure MPC. Cryptology ePrint Archive, Paper 2026/1745, 2026.

[CsHL+14] Yi-Ting Chiang, Tsan sheng Hsu, Churn-Jung Liau, Yun-Ching Liu, Chih-Hao Shen, Da-Wei Wang, and Justin Zhan. An information-theoretic approach for secure protocol composition. In *International Conference on Security and Privacy in Communication Networks, SecureComm 2014, Part I*, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pages 405–423. Springer, 2014.

[DM00] Yevgeniy Dodis and Silvio Micali. Parallel reducibility for information-theoretically secure computation. In *CRYPTO '00*, volume 1880, pages 74–92, 2000.

[DMQN20] Rafael Dowsley, Jörn Müller-Quade, and Anderson C. A. Nascimento. On the composability of statistically secure random oblivious transfer. *Entropy*, 22(1):107, 2020.

[DS83] Danny Dolev and H. Raymond Strong. Authenticated algorithms for Byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.

[DvdGMQ13] Rafael Dowsley, Jeroen van de Graaf, and Jörn Müller-Quade. On the composability of statistically secure bit commitments. *Journal of Internet Technology*, 14:509–516, 2013.

[FLS99] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM Journal on Computing*, 29(1):1–28, 1999.

[GGJS11] Sanjam Garg, Vipul Goyal, Abhishek Jain, and Amit Sahai. Bringing people of different beliefs together to do UC. In *Theory of Cryptography, 8th Theory of Cryptography Conference, TCC 2011*, volume 6597 of *Lecture Notes in Computer Science*, pages 311–328. Springer, 2011.

[GL05] Shafi Goldwasser and Yehuda Lindell. Secure multi-party computation without agreement. *J. Cryptol.*, 18(3):247–287, 2005.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, STOC 1987*, pages 218–229. ACM, 1987.

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1: Basic Techniques*. Cambridge University Press, 2001.

[Gol04] Oded Goldreich. *The Foundations of Cryptography - Volume 2: Basic Applications*. Cambridge University Press, 2004.

[GS14] Divya Gupta and Amit Sahai. On constant-round concurrent zero-knowledge from a knowledge assumption. In *Progress in Cryptology – INDOCRYPT 2014*, pages 71–88. Springer, 2014.

[GS18] Sanjam Garg and Akshayaram Srinivasan. Two-round multiparty secure computation from minimal assumptions. In *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*. Springer, 2018.

[Her24] Maya Herrscher. *Secure Composition of Public Key Infrastructures and Protocols in Computational Cryptographic Models*. PhD thesis, Universität Hamburg, 2024.

[HU06] Dennis Hofheinz and Dominique Unruh. Simulatable security and polynomially bounded concurrent composability. In *2006 IEEE Symposium on Security and Privacy (S&P 2006)*, pages 169–183. IEEE Computer Society, 2006.

[KL11] Dafna Kidron and Yehuda Lindell. Impossibility results for universal composability in public-key models and with fixed inputs. *Journal of Cryptology*, 24:517–544, 2011.

[KLP05] Yael Tauman Kalai, Yehuda Lindell, and Manoj Prabhakaran. Concurrent general composition of secure protocols in the timing model. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, STOC 2005*, pages 644–653. ACM, 2005.

[KLR06] Eyal Kushilevitz, Yehuda Lindell, and Tal Rabin. Information-theoretically secure protocols and security under composition. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, 2006.

[KLR10] Eyal Kushilevitz, Yehuda Lindell, and Tal Rabin. Information-theoretically secure protocols and security under composition. *SIAM Journal on Computing*, 2010.

[KMTZ13] Jonathan Katz, Ueli Maurer, Björn Tackmann, and Vassilis Zikas. Universally composable synchronous computation. In *Theory of Cryptography, 10th Theory of Cryptography Conference, TCC 2013*, volume 7785 of *Lecture Notes in Computer Science*, pages 477–498. Springer, 2013.

[Lin03a] Yehuda Lindell. Bounded-concurrent secure two-party computation without setup assumptions. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, STOC 2003*, pages 683–692. ACM, 2003.

[Lin03b] Yehuda Lindell. General composition and universal composability in secure multiparty computation. In *44th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2003*, pages 394–403. IEEE Computer Society, 2003.

[Lin04] Yehuda Lindell. Lower bounds for concurrent self composition. In *Theory of Cryptography, 1st Theory of Cryptography Conference, TCC 2004*, volume 2951 of *Lecture Notes in Computer Science*, pages 203–222. Springer, 2004.

[LLR06] Yehuda Lindell, Anna Lysyanskaya, and Tal Rabin. On the composition of authenticated byzantine agreement. *Journal of the ACM*, 53(6):881–917, 2006.

[LZLM+20] Chen-Da Liu-Zhang, Julian Loss, Ueli Maurer, Tal Moran, and Daniel Tschudi. MPC with synchronous security and asynchronous responsiveness. In *Advances in Cryptology – ASIACRYPT 2020, Part III*, Lecture Notes in Computer Science, pages 92–119. Springer, 2020.

[LZM20] Chen-Da Liu-Zhang and Ueli Maurer. Synchronous constructive cryptography. In *Theory of Cryptography, 18th Theory of Cryptography Conference, TCC 2020, Part II*, volume 12552 of *Lecture Notes in Computer Science*, pages 439–472. Springer, 2020.

[MMY06] Tal Malkin, Ryan Moriarty, and Nikolai Yakovenko. Generalized environmental security from number theoretic assumptions. In *Theory of Cryptography, 3rd Theory of Cryptography Conference, TCC 2006*, volume 3876 of *Lecture Notes in Computer Science*, pages 343–359. Springer, 2006.

[MPR06] Silvio Micali, Rafael Pass, and Alon Rosen. Input-indistinguishable computation. In *47th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2006*, pages 367–378. IEEE Computer Society, 2006.

[MR91] Silvio Micali and Phillip Rogaway. Secure computation (abstract). In *CRYPTO '91*, volume 576, pages 392–404, 1991.

[Pas03] Rafael Pass. Simulation in quasi-polynomial time, and its application to protocol composition. In *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 160–176. Springer, 2003.

[Pas04] Rafael Pass. Bounded-concurrent secure multi-party computation with a dishonest majority. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, STOC 2004*, pages 232–241. ACM, 2004.

[PR03] Rafael Pass and Alon Rosen. Bounded-concurrent secure two-party computation in a constant number of rounds. In *44th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2003*, pages 404–413. IEEE Computer Society, 2003.

[PS04] Manoj Prabhakaran and Amit Sahai. New notions of security: Universal composability without trusted setup. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, STOC 2004*, pages 242–251. ACM, 2004.

[PW00] Birgit Pfitzmann and Michael Waidner. Composition and integrity preservation of secure reactive systems. In *Proceedings of the 7th ACM Conference on Computer and Communications Security, CCS 2000*, pages 245–254. ACM, 2000.

[RB89] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *Proceedings of ACM Symposium on Theory of Computing*, 1989.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th Annual Symposium on Foundations of Computer Science, FOCS 1999*, pages 543–553, 1999.

[VW26] Himanshu Vashishth and Mor Weiss. Composition theorems for zero-knowledge IOPs. Cryptology ePrint Archive, Paper 2026/307, 2026.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, FOCS 1986*, pages 162–167. IEEE Computer Society, 1986.